This file is a merged representation of a subset of the codebase, containing specifically included files and files not matching ignore patterns, combined into a single document by Repomix. The content has been processed where comments have been removed, empty lines have been removed.

# File Summary

## Purpose

This file contains a packed representation of the entire repository's contents. It is designed to be easily consumable by AI systems for analysis, code review, or other automated processes.

## File Format

The content is organized as follows:

1. This summary section
2. Repository information
3. Directory structure
4. Multiple file entries, each consisting of:

a. A header with the file path (## File: path/to/file)

b. The full contents of the file in a code block

## Usage Guidelines

* This file should be treated as read-only. Any changes should be made to the original repository files, not this packed version.
* When processing this file, use the file path to distinguish between different files in the repository.
* Be aware that this file may contain sensitive information. Handle it with the same level of security as you would the original repository.
* Pay special attention to the Repository Description. These contain important context and guidelines specific to this project.

## Notes

* Some files may have been excluded based on .gitignore rules and Repomix's configuration
* Binary files are not included in this packed representation. Please refer to the Repository Structure section for a complete list of file paths, including binary files
* Only files matching these patterns are included: **/routes/**/*.js,* ***/models/****/*.js, **/src/routes/**/*.jsx,* ***/src/components/ApplicationForm/****/*.jsx, **/src/components/Case/**/*.jsx,* ***/src/components/Form/****/*.jsx, **/src/apis/**/*.js,* ***/utils/****/*.js, \*\*/config/\*.js, **/app.js,** /src/App.jsx, **/package.json,** /.env.example, \*\*/README.md
* Files matching these patterns are excluded: **/node\_modules/**, **/.git/**, **/dist/**, **/build/**, **/public/**, **/*.lock, /tests/, /test/, /spec/, /tests/, /scripts/, /migrations/, /db\_snapshot/, /letter\_templates/, /.vscode/, /.idea/, /transactions/, /bd-scs-pdf-worker/, /bd-scs-react-frontend-frontend/, /src/components/basic/, /src/components/Dialog/, /src/components/FormDetail/, /src/components/ListItem/, /src/components/Login/, /src/constants/, /src/hooks/, /src/context/, /test, /*Test*, /spec, /*Spec*, /setupTests.js, /reportWebVitals.js, /i18n.js, /*.css,** /\*.scss, \*\*/\*.less
* Files matching patterns in .gitignore are excluded
* Files matching default ignore patterns are excluded
* Code comments have been removed from supported file types
* Empty lines have been removed from all files

## Additional Info

### User Provided Header

# AIDOC Codebase Summary (Compact)

Optimized for <800k tokens - Core functionality only

# Directory Structure

bd-scs-backend-backend/  
 config/  
 application.js  
 cat.js  
 collections.js  
 letterTemplates.js  
 replyDays.js  
 task.js  
 user.js  
 models/  
 AdrBlk\_T.js  
 AdrBlkFileRef.js  
 Application\_old.js  
 Application.js  
 Attachment.js  
 BsBlock.js  
 Case.js  
 Eminute.js  
 index.js  
 Notification.js  
 OAuthToken.js  
 Submission.js  
 Sys\_Meta\_Data\_T.js  
 SysFileRef.js  
 Task.js  
 User.js  
 routes/  
 applications.js  
 attachments.js  
 auth.js  
 cases.js  
 fileReferences.js  
 index.js  
 OAuthModel.js  
 submissions.js  
 tasks.js  
 users.js  
 utils/  
 addressUtils.js  
 application.js  
 hkpostUtils.js  
 letter.js  
 MongoDBHelper.js  
 sendEmail.js  
 SQLDBHelper.js  
 app.js  
 package.json  
 README.md  
bd-scs-backend-web/  
 src/  
 apis/  
 application.js  
 auth.js  
 case.js  
 letterTemplate.js  
 task.js  
 user.js  
 components/  
 ApplicationForm/  
 CommonForm.jsx  
 constants.jsx  
 index.jsx  
 OldCommonForm.jsx  
 PremisesTable.jsx  
 RelatedPremisesTable.jsx  
 Case/  
 BuildingInformation/  
 FloorLevel/  
 Form.jsx  
 index.jsx  
 form.jsx  
 index.jsx  
 DV/  
 AdequacyOfExitsFromStoreysTable.jsx  
 AdequacyOfStaircasesTable.jsx  
 OccupantCapacityOfRoomsTable.jsx  
 P3/  
 index.jsx  
 Q1.jsx  
 Q4.jsx  
 P3ForCCC/  
 Q6/  
 q1.jsx  
 q2.jsx  
 q3.jsx  
 q4.jsx  
 q5.jsx  
 q6a.jsx  
 q6b.jsx  
 index.jsx  
 Q1.jsx  
 Q3.jsx  
 Q4.jsx  
 Q5.jsx  
 P4/  
 Q1/  
 Q3.jsx  
 Q4.jsx  
 Q5\_2.jsx  
 Q9.jsx  
 TextQuestion.jsx  
 Q2/  
 2a.jsx  
 2b.jsx  
 2c.jsx  
 index.jsx  
 Q3.jsx  
 Q4.jsx  
 P5/  
 index.jsx  
 Q1.jsx  
 Q2.jsx  
 Q3.jsx  
 Q4.jsx  
 Q5.jsx  
 Q6.jsx  
 P6/  
 Q1/  
 index.jsx  
 Q1.jsx  
 Q2.jsx  
 Q3.jsx  
 Q2/  
 Q1.jsx  
 Q2.jsx  
 index.jsx  
 Q3.jsx  
 Q4.jsx  
 SCH/  
 Info.jsx  
 TableA.jsx  
 TableB.jsx  
 TableC.jsx  
 Eminutes.jsx  
 FolioHistoryTable.jsx  
 FormHeader.jsx  
 P1.jsx  
 P2.jsx  
 P7.jsx  
 P8.jsx  
 QuestionHeader.jsx  
 RadioCheckbox.jsx  
 ReminderReceivedTable.jsx  
 SubmissionTable.jsx  
 Form/  
 Type/  
 a2.jsx  
 appointAp.jsx  
 b2.jsx  
 c2.jsx  
 cccaic2.jsx  
 cccaimh3.jsx  
 e2.jsx  
 f2.jsx  
 g2.jsx  
 jokc02.jsx  
 submitDoc.jsx  
 updateInfo.jsx  
 withdrawApp.jsx  
 addressForm.jsx  
 agencyForm.jsx  
 applicantForm.jsx  
 basicApplicantForm.jsx  
 contactForm.jsx  
 contactForm2.jsx  
 delegateForm.jsx  
 proposedCenterInfoForm.jsx  
 schoolForm.jsx  
 config/  
 cat.js  
 routes/  
 advance-search/  
 index.jsx  
 result.jsx  
 application/  
 add.jsx  
 detail.jsx  
 index.jsx  
 applicationDetails/  
 index.jsx  
 old\_index.jsx  
 building-search/  
 index.jsx  
 case/  
 BsRecommendation/  
 index.jsx  
 CCC/  
 index.jsx  
 DeskStudy/  
 detail.jsx  
 DvTable/  
 index.jsx  
 FRC/  
 detail.jsx  
 MISC/  
 detail.jsx  
 MOE/  
 detail.jsx  
 P1/  
 detail.jsx  
 P8/  
 detail.jsx  
 P9/  
 detail.jsx  
 SCH/  
 index.jsx  
 Structural/  
 bs/  
 cccdetail.jsx  
 detail.jsx  
 se/  
 cccdetail.jsx  
 detail.jsx  
 UBW/  
 detail.jsx  
 add.jsx  
 detail.jsx  
 helper.jsx  
 efolio-search/  
 index.jsx  
 forms/  
 a2.jsx  
 appointAp.jsx  
 b2.jsx  
 c2.jsx  
 cccaic2.jsx  
 cccaimh3.jsx  
 e2.jsx  
 f2.jsx  
 g2.jsx  
 index.jsx  
 jokc02.jsx  
 submitDoc.jsx  
 updateInfo.jsx  
 withdrawApp.jsx  
 legacy\_formDetail/  
 index.jsx  
 submissionDetails/  
 index.jsx  
 task/  
 index.jsx  
 user-management/  
 add.jsx  
 detail.jsx  
 list.jsx  
 dashboard.jsx  
 home.jsx  
 index.jsx  
 login.jsx  
 profile.jsx  
 report.jsx  
 App.jsx  
 package.json  
 README.md  
bd-scs-nodejs-frontend/  
 src/  
 config/  
 database.js  
 models/  
 AdrBlk\_T.js  
 AdrBlk.js  
 ApplicationCase.js  
 ApplicationFile.js  
 ApRse.js  
 Attachment.js  
 BackendUpdate.js  
 GenOtp.js  
 IamSmart.js  
 LogEvents.js  
 SchoolAppInfo.js  
 SchoolAppSubmission.js  
 ScsMasterTable.js  
 Staff.js  
 Sys\_Meta\_Data\_T.js  
 Sys\_Meta\_Data.js  
 routes/  
 ApplicationController.js  
 AuthController.js  
 ESignController.js  
 utils/  
 aes256gcm.js  
 applicationUtils.js  
 ExternalSigner.js  
 hkpostUtils.js  
 iamSmartUtils.js  
 loginUtils.js  
 on9Dotnet.js  
 signConfig.js  
 Signer.js  
 signUtils.js  
 app.js  
 package.json

# Files

## File: bd-scs-backend-backend/config/application.js

const APPLICATION\_TYPES = [  
 "NEWSCH",  
 "EXTSCH",  
 "ALTSCH",  
 "NEWCCC",  
 "NEWJOKC",  
 "NEWMUT",  
 "ALTCCC",  
 "NLHE",  
];  
const APPLICATION\_NO\_TYPES = {  
 NEWSCH: 12,  
 EXTSCH: 12,  
 NLHE: 12,  
 NEWCCC: 13,  
 NEWJOKC: 13,  
 NEWMUT: 13,  
};  
const BCIS\_CAT\_NATURE\_MAP = [  
 {  
 ApplicationType: "NEWSCH",  
 FormName: "A2",  
 category: "SCH",  
 nature: "NEW",  
 },  
 {  
 ApplicationType: "NEWSCH",  
 FormName: "B2",  
 category: "SCH",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWSCH",  
 FormName: "C2",  
 category: "SCH",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWSCH",  
 FormName: "submit-doc",  
 category: "SCH",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWSCH",  
 FormName: "update-info",  
 category: "SCH",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWSCH",  
 FormName: "appoint-ap",  
 category: "SCH",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWSCH",  
 FormName: "inspect-req",  
 category: "SCH",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWSCH",  
 FormName: "withdraw-app",  
 category: "SCH",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWSCH",  
 FormName: "SCS-1",  
 category: "SCH",  
 nature: "REV",  
 scs: true,  
 },  
 {  
 ApplicationType: "EXTSCH",  
 FormName: "E2",  
 category: "SCH",  
 nature: "NEW",  
 },  
 {  
 ApplicationType: "EXTSCH",  
 FormName: "F2",  
 category: "SCH",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "EXTSCH",  
 FormName: "G2",  
 category: "SCH",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "EXTSCH",  
 FormName: "submit-doc",  
 category: "SCH",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "EXTSCH",  
 FormName: "update-info",  
 category: "SCH",  
 nature: null,  
 },  
 {  
 ApplicationType: "EXTSCH",  
 FormName: "appoint-ap",  
 category: "SCH",  
 nature: null,  
 },  
 {  
 ApplicationType: "EXTSCH",  
 FormName: "inspect-req",  
 category: "SCH",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "EXTSCH",  
 FormName: "withdraw-app",  
 category: "SCH",  
 nature: null,  
 },  
 {  
 ApplicationType: "EXTSCH",  
 FormName: "SCS-1",  
 category: "SCH",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWCCC",  
 FormName: "CCCAI-C2",  
 category: "CCC",  
 nature: "NEW",  
 },  
 {  
 ApplicationType: "NEWCCC",  
 FormName: "submit-doc",  
 category: "CCC",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWCCC",  
 FormName: "update-info",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWCCC",  
 FormName: "appoint-ap",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWCCC",  
 FormName: "inspect-req",  
 category: "CCC",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWCCC",  
 FormName: "withdraw-app",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWCCC",  
 FormName: "SCC-1",  
 category: "CCC",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWMUT",  
 FormName: "CCCAI-MH3",  
 category: "CCC",  
 nature: "NEW",  
 },  
 {  
 ApplicationType: "NEWMUT",  
 FormName: "submit-doc",  
 category: "CCC",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWMUT",  
 FormName: "update-info",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWMUT",  
 FormName: "appoint-ap",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWMUT",  
 FormName: "inspect-req",  
 category: "CCC",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWMUT",  
 FormName: "withdraw-app",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWMUT",  
 FormName: "SCC-1",  
 category: "CCC",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWJOKC",  
 FormName: "EDB-JOKC02",  
 category: "CCC",  
 nature: "NEW",  
 },  
 {  
 ApplicationType: "NEWJOKC",  
 FormName: "submit-doc",  
 category: "CCC",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWJOKC",  
 FormName: "update-info",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWJOKC",  
 FormName: "appoint-ap",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWJOKC",  
 FormName: "inspect-req",  
 category: "CCC",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWJOKC",  
 FormName: "withdraw-app",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWJOKC",  
 FormName: "SCC-1",  
 category: "CCC",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "ALTSCH",  
 FormName: "ALTSCH",  
 category: "SCH",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTSCH",  
 FormName: "submit-doc",  
 category: "SCH",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTSCH",  
 FormName: "update-info",  
 category: "SCH",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTSCH",  
 FormName: "appoint-ap",  
 category: "SCH",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTSCH",  
 FormName: "inspect-req",  
 category: "SCH",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTSCH",  
 FormName: "withdraw-app",  
 category: "SCH",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTSCH",  
 FormName: "SCS-2",  
 category: "SCH",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "NEWNLHE",  
 FormName: "NLHE",  
 category: "NLHE",  
 nature: "NEW",  
 },  
 {  
 ApplicationType: "NEWNLHE",  
 FormName: "submit-doc",  
 category: "NLHE",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWNLHE",  
 FormName: "update-info",  
 category: "NLHE",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWNLHE",  
 FormName: "appoint-ap",  
 category: "NLHE",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWNLHE",  
 FormName: "inspect-req",  
 category: "NLHE",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "NEWNLHE",  
 FormName: "withdraw-app",  
 category: "NLHE",  
 nature: null,  
 },  
 {  
 ApplicationType: "NEWNLHE",  
 FormName: "SCN-1",  
 category: "NLHE",  
 nature: "REV",  
 },  
 {  
 ApplicationType: "ALTNLHE",  
 FormName: "NEWNHLE / ALTNHLE",  
 category: "NLHE",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTNLHE",  
 FormName: "submit-doc",  
 category: "NLHE",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTNLHE",  
 FormName: "update-info",  
 category: "NLHE",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTNLHE",  
 FormName: "appoint-ap",  
 category: "NLHE",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTNLHE",  
 FormName: "inspect-req",  
 category: "NLHE",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTNLHE",  
 FormName: "withdraw-app",  
 category: "NLHE",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTNLHE",  
 FormName: "SCN-1",  
 category: "NLHE",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTCCC",  
 FormName: "ALTCCC",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTCCC",  
 FormName: "submit-doc",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTCCC",  
 FormName: "update-info",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTCCC",  
 FormName: "appoint-ap",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTCCC",  
 FormName: "inspect-req",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTCCC",  
 FormName: "withdraw-app",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTCCC",  
 FormName: "SCC-2",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTMUT",  
 FormName: "ALTMUT",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTMUT",  
 FormName: "submit-doc",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTMUT",  
 FormName: "update-info",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTMUT",  
 FormName: "appoint-ap",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTMUT",  
 FormName: "inspect-req",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTMUT",  
 FormName: "withdraw-app",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTMUT",  
 FormName: "SCC-2",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTJOKC",  
 FormName: "ALTJOKC",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTJOKC",  
 FormName: "submit-doc",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTJOKC",  
 FormName: "update-info",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTJOKC",  
 FormName: "appoint-ap",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTJOKC",  
 FormName: "inspect-req",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "ALTJOKC",  
 FormName: "withdraw-app",  
 category: "CCC",  
 nature: null,  
 },  
 {  
 ApplicationType: "ALTJOKC",  
 FormName: "SCC-2",  
 category: "CCC",  
 nature: "ALT",  
 },  
 {  
 ApplicationType: "RENEWCCC",  
 FormName: null,  
 category: "CCC",  
 nature: "RNL",  
 },  
 {  
 ApplicationType: "RENEWMUT",  
 FormName: null,  
 category: "CCC",  
 nature: "RNL",  
 },  
 {  
 ApplicationType: "RENEWJOKC",  
 FormName: null,  
 category: "CCC",  
 nature: "RNL",  
 },  
];  
module.exports = {  
 APPLICATION\_TYPES,  
 APPLICATION\_NO\_TYPES,  
 BCIS\_CAT\_NATURE\_MAP,  
};

## File: bd-scs-backend-backend/config/cat.js

const CAT\_DESCRIPTIONS = {  
 moe\_q1\_q9\_cat2\_4: {  
 a: "The opening of the required exit at the licensed premise should have a minimum clear width of \_\_\_mm (when the glass sliding door thereat is in an open position, if any).",  
 b: "Minimum \_\_\_\_ exits should be provided to the proposed licensed premises. The \_\_\_\_ exits should have a minimum clear width of \_\_\_\_mm each and a minimum total clear width of \_\_\_mm. For double-leaf door, each leaf should have a clear width of not less than 600 mm. The lines of the travel distance between the two exits and any point in the premises should form an angle of not less than 30°. The exit doors should also open in the direction of exit and should not obstruct the exit route at any part of their swings."  
 },  
 moe\_q1\_q9\_cat3: {  
 a: "The required staircases serving the storeys of the subject building above the ground storey should have a total discharge value of not less than the total occupant capacity of those storeys.",  
 b: "The no. and width of exit doors and exit routes of \_\_\_\_/F should be not less than those shown in Table B2 of FS Code according to the occupant capacity of the \_\_\_\_/F."  
 },  
 moe\_q2\_q1\_cat2\_4: {  
 a: "The clear width of door openings for Classroom \_\_\_ should not be less than \_\_\_ mm each.",  
 b: "Minimum \_\_\_\_ exits should be provided to Classroom \_\_\_\_. The \_\_\_\_ exits should have a minimum clear width of \_\_\_\_mm each and a minimum total clear width of \_\_\_\_mm. For double-leaf door, each leaf should have a clear width of not less than 600 mm. The lines of the travel distance between the two exits and any point in Classroom\_\_\_\_ should form an angle of not less than 30°. The exit doors should also open in the direction of exit and should not obstruct the exit route at any part of their swings."  
 },  
 moe\_q2\_q3\_cat2\_4: {  
 a: "All passages leading to the exits with the licensed premises should have a minimum clear width of \_\_\_\_mm and minimum headroom of 2000mm.",  
 b: "The level difference across the required exit / exit route should be overcome by step(s) with a riser not higher than 175mm and treads of not less than 225mm wide.",  
 c: "The exit door constructed to open both ways should have a transparent upper view panel.",  
 d: "The gradient of the new ramp(s) forming part of exit route(s) should not be steeper than 1:12."  
 },  
 ubw\_q1\_options: {  
 a: "Unauthorised rooftop/flat roof/lane/yard/open area structures forming part of the licensed premises.",  
 b: "Unauthorised structures on/suspended from balcony/verandah/canopy.",  
 c: "Unauthorised canopies/retractable awnings/projections projecting from the external wall of building erected after the implementation of such MW items under the Minor Works Control System (MWCS).",  
 d: "Unauthorised signboards (including shopfront signboards) not eligible for joining the Signboard Validation Scheme.",  
 e: "Unauthorised supporting structures/frames for air-conditioning units/light fittings/mechanical plants (e.g. water cooling towers and associated supporting structures) projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof erected after the implementation of such MW items under the MWCS.",  
 f: "Unauthorised external metal ventilation ducts and associated supporting frames projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof erected after the implementation of such MW items under the MWCS.",  
 g: "Unauthorised obstructions to smoke vents.",  
 h: "Unauthorised alteration/removal of compartment walls/fire resisting walls and doors.",  
 i: "Unauthorised openings/slab over existing floors for pipe ducts.",  
 j: "Unauthorised installation of glass panels on external walls on upper floors.",  
 k: "Unauthorised removal of protective barrier/external wall without providing a permanent protective barrier at a height of 1.1 m minimum resulting in a danger of falling persons or objects.",  
 l: "Unauthorised alteration/removal of approved barrier free facilities for persons with a disability.",  
 m: "Unauthorised hollow raised platforms within premises hindering the access for persons with a disability.",  
 n: "Unauthorised removal of internal staircases not having been certified by AP/RSE as being structurally safe and accompanied with supporting calculations.",  
 o: "Unauthorised supporting frames for suspending an air-conditioning/mechanical ventilation plants of weight more than 150 kg inside the licensed premises not having been certified by AP/RSE as being structurally safe and accompanied with supporting calculations.",  
 p: "Unauthorised external solid fence walls/mesh fences/railings/poles on roof erected after the implementation of such MW items under the MWCS.",  
 q: "Unauthorised slabs filling up approved cockloft/staircase voids.",  
 r: "Unauthorised cocklofts, intermediate floors and floor extensions.",  
 s: "Unauthorised staircases.",  
 t: "Unauthorised openings through structural slabs and walls.",  
 u: "Unauthorised removal, partial removal/major alteration of structural members.",  
 v: "UBWs in common areas resulting in obstruction of means of escape from the licensed premises/the building.",  
 },  
 ubw\_q2\_q1\_option: {  
 a: "Supporting structures for building services installation (BSI) located on-grade/on roof (other than cantilevered slab/inaccessible roof) with the height of the supporting structures not more than 1.5 m.",  
 b: "Shopfront side structures (SSS), projecting not more than 150 mm from the original building line. The wall area covered by any one side of SSS is not more than 5 m2, the distance between any part of the SSS and the ground is not more than 6 m and the SSS do not obstruct access to any public utilities.",  
 c: "Shopfront overhead projections, which do not consist of wooden structural elements, projecting not more than 600 mm from the external wall of building having minimum vertical clearance of 2.5 m and not supporting other structures or equipment such as air-conditioning units. If the overhead projections consist of stone, the difference between any part of the projection and the ground should not be more than 6 m.",  
 d: "Canopies, which do not consist of stone, tile, glass/cement mortar and are not constructed of concrete, projecting not more than 500 mm from the external wall of building. The distance between the highest point of the canopy and the roof/ground is more than 3 m.",  
 e: "Retractable awnings at a door opening not serving as an exit for an escape staircase/leading to a balcony/verandah or a window opening for uses other than plant room/lavatory/bathroom/kitchen, projecting not more than 2 m (over the roof)/2.5 m (any other cases) from the external wall of building when fully extended and not more than 500 mm when retracted. The distance between the highest point of the awning and the roof/ground is not more than 5.5 m. The width of awning is at most 500 mm wider than both sides of such opening.",  
 f: "Existing signboards validated through the Signboard Validation Scheme (SVS).",  
 g: "Supporting frames for air-conditioning units/lighting fittings projecting not more than 600 mm from the external wall of building. The distance between the highest point of the frame and the roof/ground is more than 3 m.",  
 h: "External metal ventilation ducts and the associated supporting frames located on-grade/on roof. The distance between the highest point of the duct/frame and the roof/ground is not more than 1.5 m.",  
 i: "External metal ventilation ducts and associated supporting frames projecting not more than 600 mm from the external wall of building. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m.",  
 j: "External metal ventilation ducts and the associated supporting frames hung underneath the soffit of a balcony/verandah/canopy (other than cantilevered slab) with largest cross-sectional dimension of the duct not more than 600 mm. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m.",  
 k: "External metal ventilation ducts and the associated supporting frames located on a balcony/verandah/canopy (other than cantilevered slab) with largest cross-sectional dimension of the duct not more than 600 mm. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m. The distance between the highest point of the duct/frame and the balcony/verandah/canopy is not more than 1.5 m.",  
 l: "External poles located on roof including any feature at its top with the height not more than 1.5 m.",  
 m: "External solid fence walls located on roof with the height not more than 1.1 m and thickness not more than 100 mm.",  
 n: "External mesh fences/railings located on roof including any feature at its top with the height not more than 1.5 m and is not used as a protective barrier. If lower part is a solid fence wall, the height and thickness are not more than 300 mm and 100 mm respectively.",  
 o: "Unauthorised installation of glass panels on external walls on upper floors having area of each glass panel not exceeding 6 m and the least dimension of such glass panel not exceeding 1.8 m and not overloading to cantilevered slab type balconies. The remaining external wall can provide adequate protection for openings to prevent the spread of fire between floors concerned.",  
 p: "Supporting frames for suspending an air-conditioning/mechanical ventilation plant of more than 150 kg inside a building certified by AP/RSE as being structurally safe with supporting calculations.",  
 q: "Unauthorised openings/slabs over existing floors for food hoists and pipe ducts having been certified by AP/RSE as being structurally safe with supporting calculations and should not result in additional gross floor area under the Building (Planning) Regulations.",  
 r: "Unauthorised alteration/removal of approved access ramp for persons with a disability / accessible toilet for persons with a disability / accessible urinal subject to re-provision of access ramp for persons with a disability/ accessible toilet for persons with a disability / accessible urinal in compliance with Design Manual: Barrier Free Access.",  
 s: "Hollow raised platforms with a height not greater than 600 mm within premises and not hindering the access for persons with a disability.",  
 t: "Hollow raised platforms with a height between 600 mm to 2000 mm within premises if such platforms having been certified by AP/RSE as being structurally safe with supporting calculations and not hindering the access for persons with a disability.",  
 u: "Removal of internal staircase having been certified by AP/RSE as being structurally safe with supporting calculations and the staircase void should be provided by protective barriers in accordance with Building (Construction) Regulation.",  
 v: "Kitchens and toilets within approved premises and drainage works certified as being in proper function and properly connected.",  
 w: "Small storage chamber for LPG cylinders with an aggregate capacity not exceeding 130 L and complying with 'LPG Installation for Catering Purposes in Commercial Premises' issued by the Electrical and Mechanical Services Department.",  
 },  
 desk\_study\_q5\_cat1:  
 "A revised application form with correct address should be submitted.",  
 structural\_schnlhkinds\_q4\_options\_cat1\_1: {  
 a: "Construction/Alteration/Repair/Replacement/Removal of window or window wall. (建造/改動/修葺/更換/拆除窗或玻璃外牆)",  
 b: "Erection/Alteration/Removal of supporting frames for air-conditioning unit projecting from an external wall of a building. (豎設/改動/拆除自建築物外牆伸出的，用於支承空調機的支架)",  
 c: `Erection/Alteration/Removal of external Ventilation duct works and associated supporting frames projecting from an external wall of a building. (豎設/改動/拆除自建築物外牆伸出的室外金屬通風管道或相關的承托支架)`,  
 d: "Repair of structural elements. (修葺結構構件)",  
 e: "Erection/Alteration/Removal of wall signboard. (豎設/改動/拆除靠牆招牌)",  
 f: "Erection, repair, alteration or removal of aboveground / underground drains (豎設/修葺/改動/拆除地底以上 / 地下的排水渠)",  
 },  
 structural\_schnlhkinds\_q4\_options\_cat1\_2: {  
 a: "Removal of unauthorised structures projecting from the external wall of a building. (拆除自建築物外牆伸出的違例構築物)",  
 b: "Removal of unauthorised floor slab. (拆除違例樓板)",  
 c: "Erection/Alteration/Removal of protective barrier. (豎設/改動/拆除防護欄障)",  
 },  
 structural\_schnlhkinds\_q4\_options\_cat1\_3: {  
 a: "Erection/Alteration/Repair/Removal of retractable awnings. (豎設/改動/修葺/拆除可收合遮篷)",  
 b: "Erection/Alteration/Removal of external non-load bearing reinforced concrete wall of a building. (豎設/改動/拆除用鋼筋混凝土建造的非承重外牆)",  
 c: "Erection/Alteration/Removal of external non-load bearing block wall of a building. (豎設/改動/拆除用磚建造的非承重外牆)",  
 d: "Erection or alteration of any solid fence wall on-grade (豎設/改動於地面上的實心圍牆)",  
 e: "Erection or alteration of any external pole on-grade (豎設/改動於地面上的室外支柱)",  
 f: `Erection/Alteration/Repair/ Removal of mesh fences or metal railings on a roof (豎設／改動／ 修葺／拆除建築物屋頂上的網欄或金屬欄杆)`,  
 },  
 structural\_schnlhkinds\_q4\_options\_cat1\_4: {  
 a: "Alteration of any opening to the enclosure of the staircase that is used as a means of access for firefighting and rescue or its protected lobby. (改動用作消防和救援進出途徑的樓梯或其防護門廊的圍封部分上的洞口)",  
 b: "Erection of non-load bearing block walls in a flat. (於樓宇單位內豎設用磚建造的非承重牆)",  
 c: "Thickening of floor slab in a flat by laying solid screeding. (舖設實心地台，以加厚樓宇單位內的樓板)",  
 d: "Erection/Alteration of supporting frames for suspending an air-conditioning plant or a mechanical ventilation plant inside a building. (豎設/改動於建築物內用作懸掛空調裝置或機械通風裝置的承托支)",  
 e: "Erection/Alteration of ventilation duct works or associated supporting frames inside a building. (豎設/改動 於建築物內豎設用作懸掛空調裝置或機械通風裝置的承托支架)",  
 f: "Erection/Alteration of fire damper in a ventilation system. (豎設/改動於通風系統中的防火閘)",  
 },  
 structural\_schnlhkinds\_q5\_options\_cat1: {  
 a: "Any false ceiling of the proposed licensed premises should be constructed of lightweight material and the void above false ceiling should be kept vacant and not to be used for any other purpose.",  
 b: "Only lightweight partitions of gypsum board or similar construction should be erected at the licensed premises.",  
 c: "No plant and equipment or the like i.e. Building Services Installation suspended from the structural ceiling with the supporting frames having a vertical clearance of less than 2000mm within the licensed premises weights more than 150kg.",  
 },  
 structural\_schnlhkinds\_q6\_options\_cat1: {  
 a: `There is no information as to the design and construction of the subject premises, and I am unable to certify that the premises are structural suitable for use as school. You should therefore required to appoint an Authorized Person/ Registered Structural Engineer (""AP/RSE"") in respect of the followings :-  
(a) to check that the premises are in sound structural condition;  
(b) to submit justification that the premises are suitable for the intended use as school, having regard to the loading for which the premises was designed and constructed; and  
(c) to check that the resistance of fire of the structural elements of the premises are satisfactory.  
The above structural justification should be submitted for BD's acceptance. Upon the receipt of the revised plan, additional building safety requirement may be imposed.`,  
 },  
 frc\_q1\_q2b\_cat1: {  
 a: `An undertaking letter from the owner/management company of the building confirming the following aspects should be submitted:  
(a) The management company is to ensure that with the endorsement of an AP/RSE, the common internal corridors are constructed with materials having an FRR of not less than \_\_\_\_ minutes and the doors thereat are having an FRR of not less than \_\_\_\_ minutes and be self-closing. An AP/RSE is required to check and certify the fire resistance rating (FRR) and submit duly completed Appendix A to Practice Note for Authorized Persons and Registered Structural Engineers APP-13 by an AP/RSE with supporting test/assessment reports. (Such reports are not required for submission if Practice Note for Authorized Persons and Registered Structural Engineers ADM-20 is applicable.)`,  
 },  
 frc\_q2\_q1\_cat1: {  
 a: `An undertaking letter from the applicant confirming the following aspects should be submitted:  
The adjoining non-licence area is under same management and control by the applicant and the operation hours of the licensed premises should be the same or longer than the non-licence area. Revised plan with an annotation "Licensed premsies and adjoining non-licence area are under the same control and management of the applicant" should be submitted.`,  
 },  
 ubw\_q1\_q3\_cat1\_1: {  
 a: "Construction/Alteration/Repair/Replacement/Removal of window or window wall. (建造/改動/修葺/更換/拆除窗或玻璃外牆)",  
 b: "Erection/Alteration/Removal of supporting frames for air-conditioning unit projecting from an external wall of a building. (豎設/改動/拆除自建築物外牆伸出的，用於支承空調機的支架)",  
 c: "Erection/Alteration/Removal of external Ventilation duct works and associated supporting frames projecting from an external wall of a building. (豎設/改動/拆除自建築物外牆伸出的室外金屬通風管道或相關的承托支架)",  
 d: "Erection/Alteration/Removal of wall signboard. (豎設/改動/拆除 靠牆招牌)",  
 e: "Removal of unauthorised structures projecting from the external wall of a building. (拆除自建築物外牆伸出的違例構築物)",  
 f: "Removal of unauthorised suspended slab/unauthorised structure hung underneath the soffit of a balcony or canopy (拆除懸掛式違例樓板／懸掛或固定於露台或簷篷部分的違例構築物)",  
 g: "Removal of any unauthorized structure located on-grade or on a slab (other than a cantilevered slab) ((拆除位於地面或平板 ( 不包括懸臂式平板 ) 上的違例構築物)",  
 h: "Erection/Alteration/Removal of protective barrier. (豎設/改動/拆除防護欄障)",  
 i: "Erection/Alteration/Repair/Removal of retractable awnings. (豎設/改動/修葺/拆除可收合遮篷)",  
 j: "Erection/Alteration/Removal of external non-load bearing reinforced concrete wall of a building. (豎設/改動/拆除用鋼筋混凝土建造的非承重外牆)",  
 k: "Erection/Alteration/Removal of external non-load bearing block wall of a building. (豎設/改動/拆除用磚建造的非承重外牆)",  
 l: "Erection/Alteration/Repair/ Removal of mesh fences or metal railings on a roof (豎設／改動／ 修葺／拆除建築物屋頂上的網欄或金屬欄杆)",  
 },  
 ubw\_q1\_q3\_cat1: {  
 a: "Photo records showing the completion of removal/ rectification/ reinstatement works of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ should be submitted for BD’s acceptance.",  
 },  
 ubw\_q4\_cat1: {  
 a: "Revised plan showing the removal/ deleteion of \_\_\_\_\_\_\_ should be submitted for BD's acceptance. Upon the receipt of the revised plan, additional building safety requirement may be imposed.",  
 },  
 misc\_q1\_cat1: {  
 a: "The application date shown on the plan should be DATE’ in accordance with the submitted Form A2.",  
 },  
 desk\_study\_q4\_cat2\_3a: {  
 a: `Alteration and Addition Works under BD ref.: bd\_ref`,  
 },  
 ubw\_q2\_q2\_cat2\_1a: {  
 a: "Supporting frames for suspending an air-conditioning/mechanical ventilation plant of more than 150 kg inside a building",  
 b: "Unauthorised openings/slabs over existing floors for pipe ducts",  
 c: "Hollow raised platforms with a height between 600 mm to 2 000 mm within premises",  
 d: "Removal of internal staircase",  
 },  
 ubw\_q2\_q1\_cat2\_1b: {  
 a: "Supporting structures for building services installation (BSI) located on-grade/on roof (other than cantilevered slab/inaccessible roof) with the height of the supporting structures not more than 1.5 m.",  
 b: "Shopfront side structures (SSS), projecting not more than 150 mm from the original building line. The wall area covered by any one side of SSS is not more than 5 m2, the distance between any part of the SSS and the ground is not more than 6 m and the SSS do not obstruct access to any public utilities.",  
 c: "Shopfront overhead projections, which do not consist of wooden structural elements, projecting not more than 600 mm from the external wall of building having minimum vertical clearance of 2.5 m and not supporting other structures or equipment such as air-conditioning units. If the overhead projections consist of stone, the difference between any part of the projection and the ground should not be more than 6 m.",  
 d: "Canopies, which do not consist of stone, tile, glass/cement mortar and are not constructed of concrete, projecting not more than 500 mm from the external wall of building. The distance between the highest point of the canopy and the roof/ground is more than 3 m.",  
 e: "Retractable awnings at a door opening not serving as an exit for an escape staircase/ leading to a balcony/verandah or a window opening for uses other than plant room/ lavatory/bathroom/kitchen, projecting not more than 2 m (over the roof)/2.5 m (any other cases) from the external wall of building when fully extended and not more than 500 mm when retracted. The distance between the highest point of the awning and the roof/ground is not more than 5.5 m. The width of awning is at most 500 mm wider than both sides of such opening.",  
 f: "Existing signboards validated through the Signboard Validation Scheme (SVS).",  
 g: "Supporting frames for air-conditioning units/lighting fittings projecting not more than 600 mm from the external wall of building. The distance between the highest point of the frame and the roof/ground is more than 3 m.",  
 h: "External metal ventilation ducts and the associated supporting frames located on-grade/on roof. The distance between the highest point of the duct/frame and the roof/ground is not more than 1.5 m.",  
 i: "External metal ventilation ducts and associated supporting frames projecting not more than 600 mm from the external wall of building. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m.",  
 j: "External metal ventilation ducts and the associated supporting frames hung underneath the soffit of a balcony/verandah/canopy (other than cantilevered slab) with largest cross-sectional dimension of the duct not more than 600 mm. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m.",  
 k: "External metal ventilation ducts and the associated supporting frames located on a balcony/verandah/canopy (other than cantilevered slab) with largest cross-sectional dimension of the duct not more than 600 mm. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m. The distance between the highest point of the duct/frame and the balcony/verandah/canopy is not more than 1.5 m.",  
 l: "External poles located on roof including any feature at its top with the height not more than 1.5 m.",  
 m: "External solid fence walls located on roof with the height not more than 1.1 m and thickness not more than 100 mm.",  
 n: "External mesh fences/railings located on roof including any feature at its top with the height not more than 1.5 m and is not used as a protective barrier. If lower part is a solid fence wall, the height and thickness are not more than 300 mm and 100 mm respectively.",  
 o: "Unauthorised installation of glass panels on external walls on upper floors having area of each glass panel not exceeding 6 m and the least dimension of such glass panel not exceeding 1.8 m and not overloading to cantilevered slab type balconies. The remaining external wall can provide adequate protection for openings to prevent the spread of fire between floors concerned.",  
 p: "Supporting frames for suspending an air-conditioning/mechanical ventilation plant of more than 150 kg inside a building certified by AP/RSE as being structurally safe with supporting calculations.",  
 q: "Unauthorised openings/slabs over existing floors for food hoists and pipe ducts having been certified by AP/RSE as being structurally safe with supporting calculations and should not result in additional gross floor area under the Building (Planning) Regulations.",  
 r: "Unauthorised alteration/removal of approved access ramp for persons with a disability/accessible toilet for persons with a disability/accessible urinal subject to re-provision of access ramp for persons with a disability/accessible toilet for persons with a disability/accessible urinal in compliance with Design Manual: Barrier Free Access.",  
 s: "Hollow raised platforms with a height not greater than 600 mm within premises and not hindering the access for persons with a disability.",  
 t: "Hollow raised platforms with a height between 600 mm to 2,000 mm within premises if such platforms having been certified by AP/RSE as being structurally safe with supporting calculations and not hindering the access for persons with a disability.",  
 u: "Removal of internal staircase having been certified by AP/RSE as being structurally safe with supporting calculations and the staircase void should be provided by protective barriers in accordance with Building (Construction) Regulation.",  
 v: "Kitchens and toilets within approved premises and drainage works certified as being in proper function and properly connected.",  
 w: "Small storage chamber for LPG cylinders with an aggregate capacity not exceeding 130 L and complying with 'LPG Installation for Catering Purposes in Commercial Premises' issued by the Electrical and Mechanical Services Department.",  
 },  
 frc\_q2\_q2\_cat2\_3b: {  
 a: "The enclosure walls separating the licensed premises from adjoining units which is non-licence area, as highlighted on the attached plan, should be constructed up to structural ceiling and have an FRR of not less than \_\_\_\_\_ minutes.",  
 },  
 frc\_q3\_cat2\_3b: {  
 a: "The altered walls, as highlighted on the attached plan, should be constructed up to structural ceiling and have an FRR of not less than \_\_\_\_\_ minutes.",  
 b: "The entrance door of the licensed premises should have an FRR of not less than \_\_\_\_\_ minutes. Such door should be self-closing and provided with smoke seals.",  
 c: "Fire stop for sealing up openings for passage of building services through the the enclosure wall of the licensed premises facing the common internal corridor having an FRR of not less than \_\_\_\_\_ minutes.",  
 d: "Fire dampers for protecting openings for passage of ventilation ducts through the the enclosure wall of the licensed premises facing the common internal corridor having an FRR of not less than \_\_\_\_\_ minutes.",  
 },  
 frc\_q4\_cat2\_3b: {  
 a: "The new walls and doors for the fireman's lift lobby/ protected lobby of exit staircase should have an FRR of not less than \_\_\_ minutes. The walls should be constructed up to the structural ceiling and the doors should be self-closing and provided with vision panels and smoke seals.",  
 b: "The new doors of Male/ Female LAV. within the protected lobby of the exit staircase, as highlighted on the attached plan, should have an FRR of not less than \_\_\_ minutes. Such door should be self-closing and provided with smoke seals.",  
 c: "The solid return having a minimum width of 450mm up to structural floor should be provided along the shopfront of the premises adjacent to the exit staircase of the building and should have an FRR of not less than 60 minutes.",  
 },  
 frc\_q6\_cat1: {  
 a: "The existing approved self-closing fire resisting door should not be altered and should be maintained in good condition and working in order.",  
 b: "The approved fire shutters for compartment purpose within the licensed premises should not be obstructed by furniture and fixtures.",  
 c: "The approved smoke vents / smoke extraction ducts should not be altered and obstructed at all times."  
 },  
 ubw\_q1\_q3\_cat2\_4: {  
 a: "Unauthorised rooftop/flat roof/lane/yard/open area structures should be removed.",  
 b: "Unauthorised structures on/suspended from balcony/verandah/canopy should be removed.",  
 c: "Unauthorised canopies/retractable awnings/projections projecting from the external wall of building should be removed.",  
 d: "Unauthorised signboards should be removed.",  
 e: "Unauthorised supporting structures/frames for air-conditioning units/light fittings/mechanical plants projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof should be removed.",  
 f: "Unauthorised external metal ventilation ducts and associated supporting frames projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof should be removed.",  
 g: "Any obstruction to smoke vents is removed and smoke vents reinstated in accordance with the approved plans. The AP/RSE is required to certify the fire resistance rating (FRR) for the material used for reinstatement of the smoke vents and submit duly completed Appendix A to Practice Note for Authorized Persons and Registered Structural Engineers and Registered Geotechnical Engineers (PNAP) APP-13 with supporting test/assessment reports to BD (Such reports are not required for submission if PNAP ADM-20 is applicable).",  
 h: "Unauthorised alteration/removal of compartment walls/fire resisting walls and doors should be reinstated in accordance with approved plan.",  
 i: "Unauthorised openings/slab over existing floors for pipe ducts should be reinstated in accordance with approved plan.",  
 j: "Unauthorised installation of glass panels on external walls on upper floors should be removed and the configuration of external walls should be reinstated in accordance with approved plan.",  
 k: "Protective barrier/external wall should be reinstated in accordance with approved plan.",  
 l: "The access ramp for persons with a disability/ The accessible toilet for persons with a disability / The accessible urinal should be reinstated in accordance with approved plan.",  
 m: "Unauthorised hollow raised platforms within premises hindering the access for persons with a disability should be removed.",  
 n: "Internal staircase should be reinstated in accordance with approved plan.",  
 o: "Unauthorised supporting frames for suspending an air-conditioning/mechanical ventilation plants of weight more than 150 kg inside the premises should be removed.",  
 p: "Unauthorised external solid fence walls/mesh fences/railings/poles on roof should be removed.",  
 q: "Unauthorised slabs filling up approved cockloft/staircase voids should be removed and reinstated in accordance with approved plan.",  
 r: "Unauthorised cocklofts/intermediate floors/floor extensions should be removed.",  
 s: "Unauthorised staircases should be removed.",  
 t: "Unauthorised openings through structural slabs/walls should be rectified and reinstated in accordance with approved plan.",  
 u: "Unauthorised removal/partial removal/major alteration of structural members should be reinstated in accordance with approved plan.",  
 v: "UBWs in common areas resulting in obstruction of means of escape from the licensed premises/the building should be removed.",  
 },  
 ubw\_q3\_cat2\_4: {  
 a: "The accessible ramp/ accessible toilet for persons with a disability should be reinstated in accordance with the approved plan or constructed in accordance with “Design Manual Barrier Free Access 2008”.",  
 },  
 structural\_schnlhkinds\_q3\_cat2\_2a: {  
 a: "The AP/RSE is required to check and confirm the structural adequacy of the existing floor in respect of the additional loads due to the following :",  
 b: "Existing raised screed",  
 c: "Additional solid partition",  
 },  
 structural\_schnlhkinds\_q1\_cat3: {  
 a: "A structural justification should be submitted by AP/RSE to demonstrate that the the concerned premises on xx/F are structurally capable of withstanding the imposed load of 3 kPa for School use.",  
 },  
 frc\_q1\_q2a\_cat3: {  
 a: "Common internal corridor serving rooms or flats should be separated from rooms or flats by fire barriers having an fire resistance rating (FRR) of not less than that of \_\_\_\_\_ minutes.",  
 },  
 ubw\_q4\_cat3: {  
 a: "Structure within dedicated area / public passage which were excluded from GFA calculation in the original building design should be removed.",  
 },  
};  
module.exports = { CAT\_DESCRIPTIONS };

## File: bd-scs-backend-backend/config/collections.js

module.exports = {  
 Application: "Application",  
 Submission: "Submission",  
 Attachment: "Attachment",  
 BsBlock: "BsBlock",  
 Task: "Task",  
 Case: "Case",  
 Eminute: "Eminute",  
 OAuthToken: "OAuthToken",  
 User: "User",  
 Notification: "Notification",  
 SysFileRef: "SysFileRef",  
 AdrBlkFileRef: "AdrBlkFileRef",  
};

## File: bd-scs-backend-backend/config/letterTemplates.js

const LETTER\_TEMPLATES = {  
 appendix\_x: {  
 name: "Appendix X Template.docx",  
 path: "letter\_templates/lu\_appendix\_x\_template.docx",  
 fields: [  
 "mws",  
 "cat1",  
 "cat2\_1a",  
 "cat2\_1b",  
 "cat2\_2",  
 "cat2\_3a",  
 "cat2\_3b",  
 "cat2\_4",  
 "cat3",  
 "school\_name",  
 "school\_address",  
 "applicant\_name",  
 "letter\_date",  
 ],  
 },  
 appendix\_ii: {  
 name: "Appendix II Template.docx",  
 path: "letter\_templates/lu\_appendix\_ii\_template.docx",  
 fields: [],  
 },  
 appendix\_xi: {  
 name: "Appendix XI Template.docx",  
 path: "letter\_templates/lu\_appendix\_xi\_template.docx",  
 fields: [],  
 },  
 "8bl\_no\_objection": {  
 name: "8BL No Objection Template.docx",  
 path: "letter\_templates/8bl\_no\_objection\_template.docx",  
 fields: [  
 "ubws",  
 "app\_no",  
 "surname",  
 "letter\_date",  
 "bs\_bdgis",  
 "bs\_letter\_name",  
 "bs\_email",  
 ],  
 },  
 "8al": {  
 name: "8AL Template.docx",  
 path: "letter\_templates/8al\_template.docx",  
 },  
 "certificate1": {  
 name: "Certificate 1 Template.docx",  
 path: "letter\_templates/Certificate 1.docx",  
 },  
 "certificate2": {  
 name: "Certificate 2 Template.docx",  
 path: "letter\_templates/Certificate 2.docx",  
 },  
 "certificate3a": {  
 name: "Certificate 3a Template.docx",  
 path: "letter\_templates/Certificate 3a.docx",  
 },  
 "certificate3b": {  
 name: "Certificate 3b Template.docx",  
 path: "letter\_templates/Certificate 3b.docx",  
 },  
 "certificate4": {  
 name: "Certificate 4 Template.docx",  
 path: "letter\_templates/Certificate 4.docx",  
 }  
};  
module.exports = LETTER\_TEMPLATES;

## File: bd-scs-backend-backend/config/replyDays.js

const REPLY\_DAYS = {  
 CLACCC: 24,  
 RNLCCC: 22,  
 NEWCCC: 12,  
 REVCCC: 12,  
 ENQCCC: 10,  
 CMPLCCC: 10,  
 AUDCCC: 60,  
 ALTCCC: 22,  
 SCSAUDCCC: 60,  
 NEWKIND: 12,  
 REVKIND: 12,  
 CMPLKIND: 10,  
 ENQKIND: 10,  
 ALTKIND: 24,  
 CLAKIND: 24,  
 AUDKIND: 60,  
 SCSAUDKIND: 60,  
 REVNLHE: 22,  
 NEWNLHE: 22,  
 ENQNLHE: 10,  
 CMPLNLHE: 10,  
 ALTNLHE: 22,  
 CLANLHE: 24,  
 AUDNLHE: 60,  
 SCERTNLHE: 60,  
 ALTSCH: 24,  
 AUDSCH: 60,  
 CLASCH: 24,  
 CMPLSCH: 10,  
 ENQSCH: 10,  
 NEWSCH: 22,  
 REVSCH: 20,  
 SCSAUDSCH: 60,  
};  
module.exports = REPLY\_DAYS;

## File: bd-scs-backend-backend/config/task.js

const TASKS = [  
 {  
 type: "INITIAL\_SITE\_INSPECTION",  
 doneBy: "SO",  
 progressType: "BS",  
 zIndex: 1,  
 },  
 {  
 type: "ASSIGN\_GR",  
 doneBy: "GR",  
 progressType: "REG",  
 zIndex: 1,  
 },  
 {  
 type: "GR\_HANDLE",  
 doneBy: "GR",  
 progressType: "REG",  
 zIndex: 2,  
 },  
 {  
 type: "PREPARE\_STRUCTURAL\_ADVICE",  
 doneBy: "SE",  
 progressType: "SE",  
 zIndex: 1,  
 },  
 {  
 type: "CHECK\_PLAN\_AND\_SITE\_CONDITION",  
 doneBy: "BS",  
 progressType: "BS",  
 zIndex: 2,  
 },  
 {  
 type: "ENDORSE\_LETTER\_CERT",  
 doneBy: "BS",  
 progressType: "BS",  
 zIndex: 3,  
 },  
 {  
 type: "ENDORSE\_OBJECTION\_SPECIAL\_CASE",  
 doneBy: "BS",  
 progressType: "BS",  
 zIndex: 4,  
 },  
 {  
 type: "PREPARE\_LETTER\_CERT\_D",  
 doneBy: "BS",  
 progressType: "BS",  
 zIndex: 5,  
 },  
 {  
 type: "ENDORSE\_LETTER\_CERT",  
 doneBy: "BS",  
 progressType: "BS",  
 zIndex: 6,  
 },  
 {  
 type: "ENDORSE\_BS\_EMINUTES",  
 doneBy: "SBS",  
 progressType: "BS",  
 zIndex: 7,  
 },  
 {  
 type: "ENDORSE\_SE\_EMINUTES",  
 doneBy: "SSE",  
 progressType: "SE",  
 zIndex: 2,  
 },  
 {  
 type: "ENDORSE\_SO\_EMINUTES",  
 doneBy: "BS",  
 progressType: "BS",  
 zIndex: 8,  
 },  
 {  
 type: "ENDORSE\_TO\_EMINUTES",  
 doneBy: "SE",  
 progressType: "SE",  
 zIndex: 3,  
 },  
 {  
 type: "DESK\_STUDY",  
 name: "Desk Study",  
 doneBy: "SO",  
 progressType: "BS",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "REVSCH",  
 "REVKIND",  
 "SCSAUDSCH",  
 "SCSAUDKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "REVCCC",  
 "SCSAUDCCC",  
 "ALTCCC",  
 "NEWNLHE",  
 "REVNLHE",  
 "ALTNLHE",  
 "SCSAUDNLHE",  
 "CCCRNL",  
 "ALTSCHVIASCS",  
 "REVKINDVIASCS",  
 "ALTKINDVIASCS",  
 "REVCCCVIASCS",  
 "ALTCCCVIASCS",  
 "REVNLHEVIASCS",  
 "ALTNLHEVIASCS",  
 ],  
 zIndex: 9,  
 defaultStatus: "ACTIVE",  
 eminuteAction: ["SO", "BS", "ACTIVE", "COMPLETED"],  
 },  
 {  
 type: "INSPECTION\_REPORT",  
 name: "Inspection Report",  
 doneBy: "SO",  
 progressType: "BS",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "REVSCH",  
 "REVKIND",  
 "SCSAUDSCH",  
 "SCSAUDKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "REVCCC",  
 "SCSAUDCCC",  
 "ALTCCC",  
 "NEWNLHE",  
 "REVNLHE",  
 "ALTNLHE",  
 "SCSAUDNLHE",  
 "CCCRNL",  
 ],  
 zIndex: 10,  
 defaultStatus: "ACTIVE",  
 eminuteAction: ["SO", "BS", "ACTIVE", "COMPLETED"],  
 },  
 {  
 type: "ENDORSE\_SO\_EMINUTE",  
 name: "SO E-minute Endorsement",  
 doneBy: "BS",  
 progressType: "BS",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "REVSCH",  
 "REVKIND",  
 "SCSAUDSCH",  
 "SCSAUDKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "REVCCC",  
 "SCSAUDCCC",  
 "ALTCCC",  
 "NEWNLHE",  
 "REVNLHE",  
 "ALTNLHE",  
 "SCSAUDNLHE",  
 "CCCRNL",  
 "REVSCHVIASCS",  
 "ALTSCHVIASCS",  
 "REVKINDVIASCS",  
 "ALTKINDVIASCS",  
 "REVCCCVIASCS",  
 "ALTCCCVIASCS",  
 "REVNLHEVIASCS",  
 "ALTNLHEVIASCS",  
 ],  
 zIndex: 11,  
 defaultStatus: "INACTIVE",  
 eminuteAction: ["SO", "BS", "INACTIVE", "ACTIVE"],  
 endorseAction: ["SO", "BS", "ACTIVE", "COMPLETED"],  
 },  
 {  
 type: "CHECK\_PLAN\_AND\_PROFORMA",  
 name: "Plan Checking & Proforma",  
 doneBy: "BS",  
 progressType: "BS",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "REVSCH",  
 "REVKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "REVCCC",  
 "ALTCCC",  
 "NEWNLHE",  
 "REVNLHE",  
 "ALTNLHE",  
 ],  
 zIndex: 13,  
 defaultStatus: "INACTIVE",  
 eminuteAction: ["BS", "SBS", "ACTIVE", "COMPLETED"],  
 endorseAction: ["SO", "BS", "INACTIVE", "ACTIVE"],  
 },  
 {  
 type: "PREPARE\_LETTER\_CERT",  
 name: "Letter/Cert. Preparation",  
 doneBy: "BS",  
 progressType: "BS",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "REVSCH",  
 "REVKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "REVCCC",  
 "ALTCCC",  
 "REVSCHVIASCS",  
 "ALTSCHVIASCS",  
 "REVKINDVIASCS",  
 "ALTKINDVIASCS",  
 "REVCCCVIASCS",  
 "ALTCCCVIASCS",  
 "REVNLHEVIASCS",  
 "ALTNLHEVIASCS",  
 ],  
 zIndex: 14,  
 defaultStatus: "INACTIVE",  
 eminuteAction: ["BS", "SBS", "ACTIVE", "COMPLETED"],  
 endorseAction: ["SO", "BS", "INACTIVE", "ACTIVE"],  
 },  
 {  
 type: "PREPARE\_LETTER",  
 name: "Letter Preparation",  
 doneBy: "BS",  
 progressType: "BS",  
 catNature: ["SCSAUDSCH", "SCSAUDKIND", "SCSAUDCCC", "SCSAUDNLHE", "CCCRNL"],  
 zIndex: 15,  
 defaultStatus: "INACTIVE",  
 },  
 {  
 type: "PREPARE\_MEMO",  
 name: "Memo Preparation",  
 doneBy: "BS",  
 progressType: "BS",  
 catNature: ["NEWNLHE", "REVNLHE", "ALTNLHE"],  
 zIndex: 16,  
 defaultStatus: "INACTIVE",  
 },  
 {  
 type: "ENDORSE\_BS\_EMINUTE",  
 name: "BS E-minute Endorsement",  
 doneBy: "SBS",  
 progressType: "BS",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "REVSCH",  
 "REVKIND",  
 "SCSAUDSCH",  
 "SCSAUDKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "REVCCC",  
 "SCSAUDCCC",  
 "ALTCCC",  
 "NEWNLHE",  
 "REVNLHE",  
 "ALTNLHE",  
 "SCSAUDNLHE",  
 "CCCRNL",  
 "REVSCHVIASCS",  
 "ALTSCHVIASCS",  
 "REVKINDVIASCS",  
 "ALTKINDVIASCS",  
 "REVCCCVIASCS",  
 "ALTCCCVIASCS",  
 "REVNLHEVIASCS",  
 "ALTNLHEVIASCS",  
 ],  
 zIndex: 18,  
 defaultStatus: "INACTIVE",  
 eminuteAction: ["BS", "SBS", "INACTIVE", "ACTIVE"],  
 endorseAction: ["BS", "SBS", "ACTIVE", "COMPLETED"],  
 },  
 {  
 type: "LETTER\_SIGNATURE",  
 name: "Letter Signature",  
 doneBy: "SBS",  
 progressType: "BS",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "REVSCH",  
 "REVKIND",  
 "SCSAUDSCH",  
 "SCSAUDKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "REVCCC",  
 "SCSAUDCCC",  
 "ALTCCC",  
 "NEWNLHE",  
 "REVNLHE",  
 "ALTNLHE",  
 "SCSAUDNLHE",  
 "CCCRNL",  
 "REVSCHVIASCS",  
 "ALTSCHVIASCS",  
 "REVKINDVIASCS",  
 "ALTKINDVIASCS",  
 "REVCCCVIASCS",  
 "ALTCCCVIASCS",  
 "REVNLHEVIASCS",  
 "ALTNLHEVIASCS",  
 ],  
 zIndex: 18,  
 defaultStatus: "INACTIVE",  
 endorseAction: ["BS", "SBS", "INACTIVE", "ACTIVE"],  
 },  
 {  
 type: "ENDORSE\_SBS\_EMINUTE",  
 name: "SBS E-minute Endorsement",  
 doneBy: "CBS",  
 progressType: "BS",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "REVSCH",  
 "REVKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "REVCCC",  
 "ALTCCC",  
 "NEWNLHE",  
 "REVNLHE",  
 "ALTNLHE",  
 "SCSAUDNLHE",  
 "REVNLHEVIASCS",  
 "ALTNLHEVIASCS",  
 ],  
 zIndex: 19,  
 defaultStatus: "INACTIVE",  
 eminuteAction: ["SBS", "CBS", "INACTIVE", "ACTIVE"],  
 endorseAction: ["SBS", "CBS", "ACTIVE", "COMPLETED"],  
 },  
 {  
 type: "SEARCH\_STRUCTURAL\_DATA",  
 name: "Structural Data Search",  
 doneBy: "TO",  
 progressType: "SE",  
 catNature: ["NEWSCH", "NEWKIND", "NEWCCC"],  
 zIndex: 4,  
 defaultStatus: "ACTIVE",  
 eminuteAction: ["TO", "SE", "ACTIVE", "COMPLETED"],  
 },  
 {  
 type: "ENDORSE\_TO\_EMINUTE",  
 name: "TO E-minute Endorsement",  
 doneBy: "SE",  
 progressType: "SE",  
 catNature: ["NEWSCH", "NEWKIND", "NEWCCC"],  
 zIndex: 5,  
 defaultStatus: "INACTIVE",  
 eminuteAction: ["TO", "SE", "INACTIVE", "ACTIVE"],  
 endorseAction: ["TO", "SE", "ACTIVE", "COMPLETED"],  
 },  
 {  
 type: "STRUCTURAL\_ADVICE\_AND\_PROFORMA",  
 name: "Structural Advice & Proforma",  
 doneBy: "SE",  
 progressType: "SE",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "SCSAUDSCH",  
 "SCSAUDKIND",  
 "NEWCCC",  
 "REVCCC",  
 "SCSAUDCCC",  
 "SCSAUDNLHE",  
 ],  
 zIndex: 6,  
 defaultStatus: "INACTIVE",  
 eminuteAction: ["SE", "SSE", "ACTIVE", "COMPLETED"],  
 endorseAction: ["TO", "SE", "INACTIVE", "ACTIVE"],  
 },  
 {  
 type: "STRUCTURAL\_ADVICE",  
 name: "Structural Advice Preparation",  
 doneBy: "SE",  
 progressType: "SE",  
 catNature: [  
 "REVSCH",  
 "REVKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "ALTCCC",  
 "NEWNLHE",  
 "REVNLHE",  
 "ALTNLHE",  
 ],  
 zIndex: 7,  
 defaultStatus: "INACTIVE",  
 },  
 {  
 type: "ENDORSE\_SE\_EMINUTE",  
 name: "SE E-minute Endorsement",  
 doneBy: "SSE",  
 progressType: "SE",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "REVSCH",  
 "REVKIND",  
 "SCSAUDSCH",  
 "SCSAUDKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "REVCCC",  
 "SCSAUDCCC",  
 "ALTCCC",  
 "NEWNLHE",  
 "REVNLHE",  
 "ALTNLHE",  
 "SCSAUDNLHE",  
 ],  
 zIndex: 8,  
 defaultStatus: "INACTIVE",  
 eminuteAction: ["SE", "SSE", "INACTIVE", "ACTIVE"],  
 endorseAction: ["SE", "SSE", "ACTIVE", "COMPLETED"],  
 },  
 {  
 type: "CREATE\_APPLICATION\_CASE",  
 name: "Application / Case Creation",  
 doneBy: "GR",  
 progressType: "REG",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "ALTCCC",  
 "NEWNLHE",  
 "REVNLHE",  
 "ALTNLHE",  
 ],  
 zIndex: 3,  
 defaultStatus: "COMPLETED",  
 },  
 {  
 type: "CREATE\_CASE",  
 name: "Case Creation",  
 doneBy: "GR",  
 progressType: "REG",  
 catNature: [  
 "REVCCC",  
 "CCCRNL",  
 "REVCCCVIASCS",  
 "ALTCCCVIASCS",  
 "REVNLHEVIASCS",  
 "ALTNLHEVIASCS",  
 ],  
 zIndex: 4,  
 defaultStatus: "COMPLETED",  
 },  
 {  
 type: "ISSUE\_ACKNOWLEDGEMENT\_LETTER",  
 name: "Issue Acknowledgement Letter",  
 doneBy: "GR",  
 progressType: "REG",  
 catNature: ["NEWSCH", "NEWKIND"],  
 zIndex: 5,  
 defaultStatus: "COMPLETED",  
 },  
 {  
 type: "DISPATCH",  
 name: "Dispatch",  
 doneBy: "GR",  
 progressType: "REG",  
 catNature: [  
 "NEWSCH",  
 "NEWKIND",  
 "REVSCH",  
 "REVKIND",  
 "SCSAUDSCH",  
 "SCSAUDKIND",  
 "ALTSCH",  
 "ALTKIND",  
 "NEWCCC",  
 "REVCCC",  
 "SCSAUDCCC",  
 "ALTCCC",  
 "SCSAUDNLHE",  
 "CCCRNL",  
 "REVSCHVIASCS",  
 "ALTSCHVIASCS",  
 "REVKINDVIASCS",  
 "ALTKINDVIASCS",  
 "REVCCCVIASCS",  
 "ALTCCCVIASCS",  
 "REVNLHEVIASCS",  
 "ALTNLHEVIASCS",  
 ],  
 zIndex: 6,  
 defaultStatus: "INACTIVE",  
 },  
 {  
 type: "AUDIT\_SELECTION",  
 name: "Audit Selection",  
 doneBy: "GR",  
 progressType: "REG",  
 catNature: [  
 "REVSCHVIASCS",  
 "ALTSCHVIASCS",  
 "REVKINDVIASCS",  
 "ALTKINDVIASCS",  
 "REVCCCVIASCS",  
 "ALTCCCVIASCS",  
 "REVNLHEVIASCS",  
 "ALTNLHEVIASCS",  
 ],  
 zIndex: 7,  
 defaultStatus: "INACTIVE",  
 },  
 {  
 type: "NOTIFICATION\_LETTER",  
 name: "Notification Letter",  
 doneBy: "GR",  
 progressType: "REG",  
 catNature: [  
 "REVSCHVIASCS",  
 "ALTSCHVIASCS",  
 "REVKINDVIASCS",  
 "ALTKINDVIASCS",  
 "REVCCCVIASCS",  
 "ALTCCCVIASCS",  
 "REVNLHEVIASCS",  
 "ALTNLHEVIASCS",  
 ],  
 zIndex: 8,  
 defaultStatus: "INACTIVE",  
 },  
 {  
 type: "DESK\_STUDY\_DOCUMENT\_CHECKLIST",  
 name: "Desk Study & Document Checklist",  
 doneBy: "SO",  
 progressType: "BS",  
 catNature: [  
 "REVSCHVIASCS",  
 ],  
 zIndex: 9,  
 defaultStatus: "ACTIVE",  
 eminuteAction: ["SO", "BS", "ACTIVE", "COMPLETED"],  
 },  
 {  
 type: "BS\_RECOMMENDATION",  
 name: "BS Recommendation",  
 doneBy: "BS",  
 progressType: "BS",  
 catNature: [  
 "REVSCHVIASCS",  
 ],  
 zIndex: 12,  
 defaultStatus: "INACTIVE",  
 eminuteAction: ["BS", "SBS", "ACTIVE", "COMPLETED"],  
 eminuteAction: ["SO", "BS", "INACTIVE", "ACTIVE"],  
 },  
];  
module.exports = {  
 TASKS,  
};

## File: bd-scs-backend-backend/config/user.js

const BS\_TEAM\_MAPPINGS = {  
 "BS/Lic1": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic1",  
 SO: "SO/Lic1",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic2",  
 TO: "TO/Lic3",  
 },  
 "BS/Lic2": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic1",  
 SO: "SO/Lic2",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic3",  
 TO: "TO/Lic2",  
 },  
 "BS/Lic4": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic1",  
 SO: "SO/Lic4",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic(TP)",  
 TO: "TO/Lic(TP)",  
 },  
 "BS/Lic5": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic1",  
 SO: "SO/Lic5",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic1",  
 TO: "TO/Lic1",  
 },  
 "BS/Lic10": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic1",  
 SO: "SO/Lic10",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic(TP)",  
 TO: "TO/Lic(TP)",  
 },  
 "BS/Lic12": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic1",  
 SO: "SO/Lic12",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic1",  
 TO: "TO/Lic1",  
 },  
 "BS/Lic(TP)": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic1",  
 SO: "SO/Lic(TP)",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic(TP)",  
 TO: "TO/Lic(TP)",  
 },  
 "BS/Lic(SD)": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic1",  
 SO: "SSO/Lic",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic1",  
 TO: "TO/Lic1",  
 },  
 "BS/Lic7": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic3(Atg)",  
 SO: "SO/Lic7",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic(TP)",  
 TO: "TO/Lic(TP)",  
 },  
 "BS/Lic8": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic3(Atg)",  
 SO: "SO/Lic8",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic2",  
 TO: "TO/Lic3",  
 },  
 "BS/Lic9": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic3(Atg)",  
 SO: "SO/Lic9",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic(TP)",  
 TO: "TO/Lic(TP)",  
 },  
 "BS/Lic6": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic3(Atg)",  
 SO: "SO/Lic6",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic3",  
 TO: "TO/Lic2",  
 },  
 "BS/Lic11": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic3(Atg)",  
 SO: "SO/Lic11",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic1",  
 TO: "TO/Lic1",  
 },  
 "BS/Lic3": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic2",  
 SO: "SO/Lic15",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic2",  
 TO: "TO/Lic3",  
 },  
 "BS/Lic13": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic2",  
 SO: "SO/Lic13",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic3",  
 TO: "TO/Lic2",  
 },  
 "BS/Lic14": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic2",  
 SO: "SO/Lic16",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic2",  
 TO: "TO/Lic3",  
 },  
 "BS/Lic15": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic2",  
 SO: "SO/Lic14",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic1",  
 TO: "TO/Lic1",  
 },  
 "BS/Lic16": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic2",  
 SO: "",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic(TP)",  
 TO: "TO/Lic(TP)",  
 },  
 "BS/Lic17": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic2",  
 SO: "SO/Lic17",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic3",  
 TO: "TO/Lic2",  
 },  
 "BS/Lic18": {  
 CBS: "CBS/NTE1&L",  
 SBS: "SBS/Lic2",  
 SO: "SO/Lic18",  
 SSE: "SSE/Lic",  
 SE: "SE/Lic1",  
 TO: "TO/Lic1",  
 },  
};  
module.exports = {  
 BS\_TEAM\_MAPPINGS,  
};

## File: bd-scs-backend-backend/models/AdrBlk\_T.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../utils/SQLDBHelper");  
class AdrBlkModel extends Model {}  
AdrBlkModel.init(  
 {  
 ADR\_BLK\_ID: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 BLK\_TYPE\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 BLDG\_CAT\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 BLDG\_USAGE\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 SYS\_REGION\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 SYS\_DISTRICT\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 AREA\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 BLK\_DESC\_E\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 BLK\_NO\_NUM: {  
 type: DataTypes.INTEGER,  
 allowNull: true,  
 },  
 BLK\_NO\_ALPHA: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_E3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_C1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_C2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_C3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E5: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_ST\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_ST\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_ST\_C: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C5: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OBSOLETE: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "AdrBlkModel\_T",  
 tableName: "AdrBlk\_T",  
 timestamps: true,  
 }  
);  
module.exports = AdrBlkModel;

## File: bd-scs-backend-backend/models/AdrBlkFileRef.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const AdrBlkFileRefSchema = new Schema({  
 adrBlkFileRefId: { type: String, required: true },  
 adrBlkId: { type: String, required: true },  
 sysFileRefId: { type: String, required: true },  
 createdDt: { type: Date, default: Date.now },  
 createdPost: { type: String, required: false },  
 createdName: { type: String, required: false },  
 createdSection: { type: String, required: false },  
 lastModifiedDt: { type: Date, default: Date.now },  
 lastModifiedPost: { type: String, required: false },  
 lastModifiedName: { type: String, required: false },  
 lastModifiedSection: { type: String, required: false },  
});  
module.exports = AdrBlkFileRefSchema;

## File: bd-scs-backend-backend/models/Application\_old.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const APPLICATION\_TYPES = [  
 "NEWSCH",  
 "EXTSCH",  
 "ALTSCH",  
 "NEWCCC",  
 "NEWJOKC",  
 "NEWMUT",  
 "ALTCCC",  
 "NLHE",  
];  
const APPLICATION\_NO\_TYPES = {  
 NEWSCH: 12,  
 EXTSCH: 12,  
 NLHE: 12,  
 NEWCCC: 13,  
 NEWJOKC: 13,  
 NEWMUT: 13,  
};  
const ApplicationStatus = {  
 DRAFT: "DRAFT",  
};  
const ApplicationSchema = new Schema({  
 ApplicationNo: String,  
 ApplicationType: {  
 type: String,  
 enum: APPLICATION\_TYPES,  
 },  
 EApplication: Boolean,  
 FormName: String,  
 ApplicantName: String,  
 ApplicantTitle: String,  
 ApplicantNameEN: String,  
 ApplicantEmail: String,  
 ApplicantMobile: String,  
 ApplicantHKIC: String,  
 ApplicantPosition: String,  
 ApplicationDate: String,  
 ApplicantPostTitle: String,  
 ApplicantRepOrgEN: String,  
 ApplicantRepOrg: String,  
 ApplicantAddressEN: String,  
 ApplicantTel: String,  
 AddressOfPremiseEN: String,  
 AddressOfPremiseENFloor: String,  
 AddressOfPremiseENUnit: String,  
 AddressOfPremiseCN: String,  
 AddressOfPremiseCNFloor: String,  
 AddressOfPremiseCNUnit: String,  
 BlockId: String,  
 CentreNameEnglish: String,  
 CentreNameChinese: String,  
 PhoneOfSchool: String,  
 ContactPTitle: String,  
 ContactIdentity: String,  
 ContactPName: String,  
 ContactNameEnglish: String,  
 ContactNameChinese: String,  
 ApplicantAddress: String,  
 ContactPAddress: String,  
 ContactAddressLine1: String,  
 ContactAddressLine2: String,  
 ContactPEmail: String,  
 ContactPTel: String,  
 ContactPMobile: String,  
 ContactPFax: String,  
 ContactCertRegNo: String,  
 ContactRegExpiryDate: String,  
 NameOfSchoolEN: String,  
 NameOfSchoolCN: String,  
 DescriptionOfSchool: String,  
 SchoolPremisesAddressLine1: String,  
 SchoolPremisesAddressLine2: String,  
 OtherSchoolDesc: String,  
 BdRefNo: String,  
 AgeOfStudent: String,  
 EstimatedNoOfStudent: Number,  
 BDLetterRefDate: String,  
 ChineseNameOfAp1: String,  
 EnglishNameOfAp1: String,  
 EmailOfAp1: String,  
 MobileOfAp1: String,  
 RegTypeOfAp1: String,  
 CertRegNoOfAp1: String,  
 ChineseNameOfAp2: String,  
 EnglishNameOfAp2: String,  
 EmailOfAp2: String,  
 MobileOfAp2: String,  
 RegTypeOfAp2: String,  
 CertRegNoOfAp2: String,  
 ChineseNameOfRse: String,  
 EnglishNameOfRse: String,  
 EmailOfRse: String,  
 MobileOfRse: String,  
 RegTypeOfRse: String,  
 CertRegNoOfRse: String,  
 AmendmentOfLayoutPlan: String,  
 Status: {  
 type: String,  
 enum: Object.values(ApplicationStatus),  
 },  
 updatedAt: { type: Date },  
 createdAt: { type: Date, default: Date.now },  
});  
module.exports = ApplicationSchema;  
module.exports.APPLICATION\_TYPES = APPLICATION\_TYPES;  
module.exports.APPLICATION\_NO\_TYPES = APPLICATION\_NO\_TYPES;

## File: bd-scs-backend-backend/models/Application.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const APPLICATION\_TYPES = {  
 SCH: "SCH",  
 CCC: "CCC",  
 NEWSCH: "NEWSCH",  
 NEWCCC: "NEWCCC",  
 EXTSCH: "EXTSCH",  
 EXTCCC: "EXTCCC",  
 ALTCCC: "ALTCCC",  
 NEWJOKC: "NEWJOKC",  
 NEWMUT: "NEWMUT",  
 ALTCCC: "ALTCCC",  
 NLHE: "NLHE",  
};  
const APPLICATION\_NO\_TYPES = {  
 SCH: 12,  
 CCC: 13,  
 NLHE: 12,  
};  
const ApplicationStatus = {  
 DRAFT: "DRAFT",  
};  
const SubmissionType = {  
 PAPER: "PAPER",  
 EAPPLICATION: "EAPPLICATION",  
};  
const ApplicationSchema = new Schema({  
 ApplicationNo: String,  
 ApplicationType: {  
 type: String,  
 enum: Object.values(APPLICATION\_TYPES),  
 },  
 SubmissionType: {  
 type: String,  
 enum: Object.values(SubmissionType),  
 },  
 FileReference: String,  
 Region: String,  
 District: String,  
 Area: String,  
 NameOfSchoolEN: String,  
 NameOfSchoolCN: String,  
 DescriptionOfSchool: String,  
 AgeOfStudent: String,  
 EstimatedNoOfStudent: Number,  
 assignedBS: String,  
 assignedSBS: String,  
 assignedGR: String,  
 address: Object,  
 BlockID: String,  
 AddressOfPremiseEN: String,  
 AddressOfPremiseCN: String,  
 AddressOfPremiseENFloor: String,  
 AddressOfPremiseCNFloor: String,  
 AddressOfPremiseENUnit: String,  
 AddressOfPremiseCNUnit: String,  
 ApplicantNameEN: String,  
 ApplicantNameCN: String,  
 ApplicantAddress: String,  
 ApplicantTel: String,  
 ApplicantFax: String,  
 ContactPersonEN: String,  
 ContactPersonCN: String,  
 ApplicantEmail: String,  
 ContactPersonEmail: String,  
 ApplicantMobile: String,  
 ContactPersonTel: String,  
 RelatedPremise: String,  
 RelatedPremises: [  
 {  
 ID: String,  
 Type: String,  
 AdditionalInfo: String,  
 OPYear: String,  
 Address: String,  
 AddressObject: Object,  
 },  
 ],  
 APP13: [  
 {  
 RegistrationNumberPrefix: String,  
 RegistrationNumber: String,  
 NameEN: String,  
 NameCN: String,  
 Email: String,  
 Mobile: String,  
 },  
 ],  
 SelfCertification: {  
 RegistrationNumberPrefix: String,  
 RegistrationNumber: String,  
 NameEN: String,  
 NameCN: String,  
 Email: String,  
 Mobile: String,  
 },  
 StructuralCalculation: {  
 RegistrationNumberPrefix: String,  
 RegistrationNumber: String,  
 NameEN: String,  
 NameCN: String,  
 Email: String,  
 Mobile: String,  
 },  
 Status: {  
 type: String,  
 enum: Object.values(ApplicationStatus),  
 },  
 updatedAt: { type: Date },  
 createdAt: { type: Date, default: Date.now },  
});  
module.exports = ApplicationSchema;  
module.exports.APPLICATION\_TYPES = APPLICATION\_TYPES;  
module.exports.APPLICATION\_NO\_TYPES = APPLICATION\_NO\_TYPES;  
module.exports.SubmissionType = SubmissionType;

## File: bd-scs-backend-backend/models/Attachment.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const AttachemntType = {  
 APPLICATION: "APPLICATION",  
 CASE: "CASE",  
 CREATE\_CASE: "CREATE\_CASE",  
 INSPECTION\_REPORT: "INSPECTION\_REPORT",  
 FIRE\_ENGINEERING\_PLAN: "FIRE\_ENGINEERING\_PLAN",  
 STRUCTURAL\_DATA\_SEARCH: "STRUCTURAL\_DATA\_SEARCH",  
 PREPARE\_LETTER: "PREPARE\_LETTER",  
 ISSUE\_LETTER: "ISSUE\_LETTER",  
};  
const AttachemntSchema = new Schema({  
 application: { type: Schema.Types.ObjectId, ref: "Application" },  
 submissionCase: { type: Schema.Types.ObjectId, ref: "Case" },  
 sysFileRefId: String,  
 efolio: String,  
 filePartNo: String,  
 remarks: String,  
 receivedDate: { type: Date },  
 type: {  
 type: String,  
 enum: Object.values(AttachemntType),  
 },  
 subType: String,  
 file: Object,  
 updatedAt: { type: Date },  
 createdAt: { type: Date, default: Date.now },  
});  
module.exports = AttachemntSchema;  
module.exports.AttachemntType = AttachemntType;

## File: bd-scs-backend-backend/models/BsBlock.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const BsBlockSchema = new Schema({  
 blockId: String,  
 bdgis: String,  
});  
module.exports = BsBlockSchema;

## File: bd-scs-backend-backend/models/Case.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const SubmissionType = {  
 PAPER: "PAPER",  
 ELECTRONIC: "ELECTRONIC",  
};  
const ReminderSchema = new Schema({  
 ReceivedDate: Date,  
 Remarks: String,  
 No: String,  
});  
const CaseSchema = new Schema({  
 application: { type: Schema.Types.ObjectId, ref: "Application" },  
 assignedGR: String,  
 caseOfficerReceive: String,  
 caseOfficerReply: String,  
 seniorCaseOfficerReceive: String,  
 seniorCaseOfficerReply: String,  
 caseNumber: String,  
 licensingCaseID: String,  
 SubmissionType: {  
 type: String,  
 enum: Object.values(SubmissionType),  
 },  
 FileReference: String,  
 LAFileReference: {  
 a: String,  
 b: String,  
 },  
 ViaSCS: Boolean,  
 Category: String,  
 Nature: String,  
 Region: String,  
 District: String,  
 Area: String,  
 ReceivedDate: Date,  
 SubstantialReplyDate: Date,  
 ActualReplyDate: Date,  
 CaseOfficer: String,  
 TargetReplyDate: Date,  
 ThreeTierReqt: String,  
 ObjectiontoLR: String,  
 AuditResult: String,  
 Remarks: String,  
 Referrer: {  
 GovDepartment: String,  
 FaxNumber: String,  
 Email: String,  
 },  
 Reminders: [ReminderSchema],  
 caseDescription: {  
 efolio: String,  
 filePartNo: String,  
 description: String,  
 remarks: String,  
 },  
 deck\_study: {  
 q1: [  
 {  
 submission: String,  
 luReply: String,  
 nature: String,  
 inspectionReport: String,  
 licensePlan: String,  
 },  
 ],  
 q2: {  
 answer: Boolean,  
 },  
 q3: [  
 {  
 fileRef: String,  
 approvedDate: Date,  
 acknowledgementDate: Date,  
 },  
 ],  
 q4: {  
 answer: Boolean,  
 options\_cat2\_3a: [String],  
 options\_cat2\_3a\_text: {  
 a: String,  
 },  
 value\_cat2\_3a: String,  
 },  
 q5: {  
 answer: Boolean,  
 answer\_cat1: Boolean,  
 answer\_cat1\_text: String,  
 },  
 q6: {  
 answer: Boolean,  
 },  
 q7: {  
 answer: Boolean,  
 },  
 q8: {  
 answer: String,  
 },  
 },  
 building\_information: {  
 address: String,  
 lotNo: String,  
 siteID: String,  
 notionalSiteID: String,  
 blockID: String,  
 typeOfBuilding: String,  
 occupationPermit: String,  
 },  
 structural\_schnlhkinds: {  
 q1: {  
 answer: Boolean,  
 kpa: Number,  
 options\_cat3: [String],  
 options\_cat3\_text: {  
 a: String,  
 },  
 other\_cat3: String,  
 },  
 q2: {  
 answer: Boolean,  
 },  
 q3: {  
 answer: Boolean,  
 options\_cat2\_2a: [String],  
 options\_cat2\_2a\_text: {  
 a: String,  
 b: String,  
 c: String,  
 },  
 other\_cat2\_2a: String,  
 },  
 q4: {  
 answer: Boolean,  
 options\_cat1\_1: [String],  
 options\_cat1\_1\_text: {  
 a: String,  
 b: String,  
 c: String,  
 d: String,  
 e: String,  
 f: String,  
 },  
 options\_cat1\_2: [String],  
 options\_cat1\_2\_text: {  
 a: String,  
 b: String,  
 c: String,  
 },  
 options\_cat1\_3: [String],  
 options\_cat1\_3\_text: {  
 a: String,  
 b: String,  
 c: String,  
 d: String,  
 e: String,  
 f: String,  
 },  
 options\_cat1\_4: [String],  
 options\_cat1\_4\_text: {  
 a: String,  
 b: String,  
 c: String,  
 d: String,  
 e: String,  
 f: String,  
 other: String,  
 },  
 other\_cat1\_4: [String],  
 },  
 q5: {  
 options\_cat1: [String],  
 options\_cat1\_text: {  
 a: String,  
 b: String,  
 c: String,  
 },  
 other\_cat1: String,  
 },  
 q6: {  
 options\_cat1: [String],  
 options\_cat1\_text: {  
 a: String,  
 },  
 other\_cat1: String,  
 },  
 },  
 structural\_ccc\_se: {  
 q1\_a\_playground: Boolean,  
 q1\_a\_childcarecentres: Boolean,  
 q1\_b\_playground: Boolean,  
 q1\_b\_childcarecentres: Boolean,  
 q1\_c: String,  
 },  
 structural\_ccc\_bs: {  
 q1: {  
 answer: Boolean,  
 other\_cat1: String,  
 other\_cat2\_1a: String,  
 other\_cat2\_1b: String,  
 other\_cat2\_2a: String,  
 other\_cat2\_4: String,  
 other\_cat3: String,  
 },  
 q2: {  
 answer: Boolean,  
 },  
 q3: {  
 answer: Boolean,  
 options\_cat3: [String],  
 other\_cat3: String,  
 },  
 q4: {  
 answer: Boolean,  
 options\_cat2\_2a: [String],  
 other\_cat2\_2a: String,  
 },  
 q5: {  
 answer: Boolean,  
 common\_minor\_works: [String],  
 structural\_alteration\_or\_removal\_works: [String],  
 roof\_or\_external\_works: [String],  
 subdivied\_flat\_or\_interior\_works: [String],  
 subdivied\_flat\_or\_interior\_works\_other: String,  
 other\_general\_conditions: [String],  
 },  
 q6: {  
 answer: Boolean,  
 q6\_1: {  
 answer: Boolean,  
 options\_cat1: [String],  
 other\_cat1: String,  
 options\_cat2: [String],  
 other\_cat2: String,  
 options\_cat3: [String],  
 other\_cat3: String,  
 },  
 q6\_2: {  
 answer: Boolean,  
 options\_cat1: [String],  
 other\_cat1: String,  
 options\_cat2: [String],  
 other\_cat2: String,  
 options\_cat3: [String],  
 other\_cat3: String,  
 },  
 q6\_3: {  
 answer: Boolean,  
 options\_cat1: [String],  
 other\_cat1: String,  
 options\_cat2: [String],  
 other\_cat2: String,  
 options\_cat3: [String],  
 other\_cat3: String,  
 },  
 q6\_4: {  
 answer: Boolean,  
 options\_cat1: [String],  
 other\_cat1: String,  
 options\_cat2: [String],  
 other\_cat2: String,  
 options\_cat3: [String],  
 other\_cat3: String,  
 },  
 q6\_5: {  
 answer: Boolean,  
 options\_cat1: [String],  
 other\_cat1: String,  
 options\_cat2: [String],  
 other\_cat2: String,  
 options\_cat3: [String],  
 other\_cat3: String,  
 },  
 q6\_6a: {  
 answer: Boolean,  
 options\_cat1: [String],  
 other\_cat1: String,  
 options\_cat2: [String],  
 other\_cat2: String,  
 options\_cat3: [String],  
 other\_cat3: String,  
 },  
 q6\_6b: {  
 answer: Boolean,  
 options\_cat1: [String],  
 other\_cat1: String,  
 options\_cat2: [String],  
 other\_cat2: String,  
 options\_cat3: [String],  
 other\_cat3: String,  
 },  
 q7: {  
 number: Number,  
 },  
 },  
 },  
 moe: {  
 q1: {  
 q3: {  
 answer: Boolean,  
 cat3: [String],  
 cat3\_other: String,  
 },  
 q4: {  
 numberOfKindergartenFloor: Number,  
 numberOfCommercialFloor: Number,  
 answer: Boolean,  
 cat3: [String],  
 cat3\_other: String,  
 },  
 q5: {  
 q1: {  
 answer: Boolean,  
 },  
 q2: {  
 answer: Boolean,  
 cat3: [String],  
 cat3\_other: String,  
 },  
 },  
 q9: {  
 cat1\_field\_1: String,  
 cat3: [String],  
 cat3\_option2\_field\_1: String,  
 cat3\_option2\_field\_2: String,  
 cat3\_other: String,  
 cat2\_4: [String],  
 cat2\_4\_option1\_field\_1: String,  
 cat2\_4\_option1\_field\_2: String,  
 cat2\_4\_option1\_field\_3: String,  
 cat2\_4\_option1\_field\_4: String,  
 cat2\_4\_option2\_field\_1: String,  
 cat2\_4\_other: String,  
 },  
 },  
 q2: {  
 q1: {  
 answer: Boolean,  
 cat2\_4: [String],  
 cat2\_4\_other: String,  
 },  
 q2: {  
 answer: Boolean,  
 cat1: [String],  
 cat1\_other: String,  
 },  
 q3: {  
 cat1: [String],  
 cat1\_option5\_field\_1: String,  
 cat1\_other: String,  
 cat2\_4: [String],  
 cat2\_4\_option1\_field\_1: String,  
 cat2\_4\_other: String,  
 },  
 },  
 q3: {  
 answer: Boolean,  
 cat1: [String],  
 cat1\_other: String,  
 },  
 q4: {  
 answer: Boolean,  
 cat1: [String],  
 cat1\_other: String,  
 },  
 },  
 frc: {  
 q1: {  
 answer: Boolean,  
 q1: {  
 answer: Boolean,  
 },  
 q2a: {  
 cat3: [String],  
 cat3\_text: {  
 a: String,  
 },  
 cat3\_other: String,  
 },  
 q2b: {  
 answer: Boolean,  
 cat1: [String],  
 cat1\_text: {  
 a: String,  
 },  
 cat1\_option1\_field\_1: String,  
 cat1\_option1\_field\_2: String,  
 cat1\_other: String,  
 },  
 },  
 q2: {  
 answer: Boolean,  
 q1: {  
 answer: Boolean,  
 cat1: [String],  
 cat1\_text: {  
 a: String,  
 },  
 cat1\_other: String,  
 },  
 q2: {  
 answer: Boolean,  
 cat2\_3b: [String],  
 cat2\_3b\_text: {  
 a: String,  
 },  
 cat2\_3b\_other: String,  
 },  
 },  
 q3: {  
 answer: Boolean,  
 cat2\_3b: [String],  
 cat2\_3b\_text: {  
 a: String,  
 b: String,  
 c: String,  
 d: String,  
 },  
 cat2\_3b\_other: String,  
 },  
 q4: {  
 answer: Boolean,  
 cat2\_3b: [String],  
 cat2\_3b\_text: {  
 a: String,  
 b: String,  
 c: String,  
 },  
 cat2\_3b\_other: String,  
 },  
 q5: {  
 answer: Boolean,  
 cat1: [String],  
 cat1\_other: String,  
 },  
 q6: {  
 cat1: [String],  
 cat1\_other: String,  
 },  
 },  
 ubw: {  
 q1: {  
 answer: Boolean,  
 options: [String],  
 options\_text: {  
 a: String,  
 b: String,  
 c: String,  
 d: String,  
 e: String,  
 f: String,  
 g: String,  
 h: String,  
 i: String,  
 j: String,  
 k: String,  
 l: String,  
 m: String,  
 n: String,  
 o: String,  
 p: String,  
 q: String,  
 r: String,  
 s: String,  
 t: String,  
 u: String,  
 v: String,  
 },  
 other: String,  
 q1: {  
 answer: Boolean,  
 option\_yes: [String],  
 option\_no: [String],  
 },  
 q3: {  
 cat1\_1: [String],  
 cat1\_1\_text: {  
 a: String,  
 b: String,  
 c: String,  
 d: String,  
 e: String,  
 f: String,  
 g: String,  
 h: String,  
 i: String,  
 j: String,  
 k: String,  
 l: String,  
 },  
 cat1\_1\_other: String,  
 cat1: [String],  
 cat1\_text: {  
 a: String,  
 },  
 cat1\_other: String,  
 cat2\_4: [String],  
 cat2\_4\_text: {  
 a: String,  
 b: String,  
 c: String,  
 d: String,  
 e: String,  
 f: String,  
 g: String,  
 h: String,  
 i: String,  
 j: String,  
 k: String,  
 l: String,  
 m: String,  
 n: String,  
 o: String,  
 p: String,  
 q: String,  
 r: String,  
 s: String,  
 t: String,  
 u: String,  
 v: String,  
 },  
 cat2\_4\_other: String,  
 },  
 },  
 q2: {  
 answer: Boolean,  
 q1: {  
 option: [String],  
 option\_text: {  
 a: String,  
 b: String,  
 c: String,  
 d: String,  
 e: String,  
 f: String,  
 g: String,  
 h: String,  
 i: String,  
 j: String,  
 k: String,  
 l: String,  
 m: String,  
 n: String,  
 o: String,  
 p: String,  
 q: String,  
 r: String,  
 s: String,  
 t: String,  
 u: String,  
 v: String,  
 w: String,  
 },  
 other: String,  
 },  
 q2: {  
 answer: Boolean,  
 cat2\_1a: [String],  
 cat2\_1a\_text: {  
 a: String,  
 b: String,  
 c: String,  
 d: String,  
 },  
 cat2\_1a\_other: String,  
 },  
 },  
 q3: {  
 answer: Boolean,  
 cat2\_4: [String],  
 cat2\_4\_text: {  
 a: String,  
 },  
 cat2\_4\_other: String,  
 },  
 q4: {  
 answer: Boolean,  
 cat1: [String],  
 cat1\_text: {  
 a: String,  
 },  
 cat1\_other: String,  
 cat3: [String],  
 cat3\_text: {  
 a: String,  
 },  
 cat3\_other: String,  
 },  
 q5: {  
 answer: Boolean,  
 },  
 },  
 misc: {  
 q1: {  
 answer: Boolean,  
 cat1: [String],  
 cat1\_text: {  
 a: String  
 },  
 cat1\_1: [String],  
 cat1\_1\_other: String,  
 cat1\_other: String,  
 cat2\_1a: [String],  
 cat2\_1a\_other: String,  
 cat2\_1b: [String],  
 cat2\_1b\_other: String,  
 cat2\_2a: [String],  
 cat2\_2a\_other: String,  
 cat2\_3b: [String],  
 cat2\_3b\_other: String,  
 cat2\_4: [String],  
 cat2\_4\_other: String,  
 cat3: [String],  
 cat3\_other: String,  
 },  
 },  
 dv: {  
 acceptable: Boolean,  
 occupantCapacityOfRooms: [  
 {  
 floor: String,  
 compartment: String,  
 roomName: String,  
 roomUse: String,  
 areaId: String,  
 ufa: Number,  
 FSFactor: Number,  
 approvedCapacity: Number,  
 approvedCapacity\_previousApprovedCapacity: Number,  
 newOccupantCapacity: Number,  
 revisedOccupantCapacity: Number,  
 changeInPopulation: Number,  
 noOfExit: Number,  
 exitCapacity1: Number,  
 exitCapacity2: Number,  
 exitCapacity3: Number,  
 exitCapacity4: Number,  
 exitCapacity5: Number,  
 exitCapacity6: Number,  
 exitCapacity7: Number,  
 exitCapacity8: Number,  
 exitCapacity9: Number,  
 exitCapacity10: Number,  
 exitCapacity: [Number],  
 remark: String,  
 },  
 ],  
 adequacyOfExitsFromStoreys: [  
 {  
 floor: String,  
 compartment: String,  
 capacity: Number,  
 minimumNoOfExitDoorRouteRequired: Number,  
 minimumNoOfExitDoorRouteProvided: Number,  
 minimumWidthOfExitDoorTotalRequired: Number,  
 minimumWidthOfExitDoorTotalProvided: Number,  
 minimumWidthOfExiRouteTotalRequired: Number,  
 minimumWidthOfExiRouteTotalProvided: Number,  
 minimumWidthOfExitDoorEachRequired: Number,  
 minimumWidthOfExitDoorEachProvided: Number,  
 minimumWidthOfExiRouteEachRequired: Number,  
 minimumWidthOfExiRouteEachProvided: Number,  
 checkingResult: String,  
 },  
 ],  
 adequacyOfStaircases: [  
 {  
 floor: String,  
 approvedPopulation: Number,  
 totalAcceptedPopulation: Number,  
 staircasePopulation: [Number],  
 },  
 ],  
 adequacyOfStaircasesResult: {  
 totalOccupantCapacity: {  
 approvedPopulation: Number,  
 totalAcceptedPopulation: Number,  
 staircasePopulation: [Number],  
 },  
 totalDischargeValue: {  
 totalAcceptedPopulation: Number,  
 staircasePopulation: [Number],  
 },  
 checkingResult: [String],  
 },  
 },  
 document\_checklist: {  
 q1: {  
 date: String,  
 },  
 q2: {  
 answer: String,  
 remarks: String,  
 },  
 q3: {  
 answer: String,  
 remarks: String,  
 },  
 q4: {  
 answer: String,  
 remarks: String,  
 },  
 q5: {  
 answer: String,  
 remarks: String,  
 },  
 q6: {  
 answer: String,  
 remarks: String,  
 },  
 q7: {  
 answer: String,  
 remarks: String,  
 },  
 q8: {  
 answer: String,  
 remarks: String,  
 },  
 q9: {  
 answer: String,  
 remarks: [String],  
 remarks\_options: [Boolean]  
 },  
 },  
 bs\_recommendation: {  
 q1: {  
 answer: Boolean,  
 remarks: String,  
 },  
 q2: {  
 answer: Boolean,  
 remarks: String,  
 },  
 },  
 team: String,  
 updatedAt: { type: Date },  
 createdAt: { type: Date, default: Date.now },  
});  
module.exports = CaseSchema;

## File: bd-scs-backend-backend/models/Eminute.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const EminuteStatus = {  
 PENDING: "PENDING",  
 ENDORSED: "ENDORSED",  
 RETURN: "RETURN",  
 RECOMMEND: "RECOMMEND",  
};  
const EminuteSchema = new Schema({  
 submissionCase: { type: Schema.Types.ObjectId, ref: "Case" },  
 application: { type: Schema.Types.ObjectId, ref: "Application" },  
 sysFileRefId: String,  
 efolio: String,  
 from: String,  
 to: String,  
 status: {  
 type: String,  
 enum: Object.values(EminuteStatus),  
 },  
 comment: String,  
 eminuteId: String,  
 subject: String,  
 content: String,  
 updatedAt: { type: Date },  
 createdAt: { type: Date, default: Date.now },  
});  
module.exports = EminuteSchema;  
module.exports.EminuteStatus = EminuteStatus;

## File: bd-scs-backend-backend/models/index.js

"use strict";  
const fs = require("fs");  
const path = require("path");  
const Sequelize = require("sequelize");  
const process = require("process");  
const basename = path.basename(\_\_filename);  
const env = process.env.NODE\_ENV || "development";  
const config = require(\_\_dirname + "/../config/config.json")[env];  
const db = {};  
let sequelize;  
if (config.use\_env\_variable) {  
 sequelize = new Sequelize(process.env[config.use\_env\_variable], config);  
} else {  
 sequelize = new Sequelize(  
 config.database,  
 config.username,  
 config.password,  
 config  
 );  
}  
fs.readdirSync(\_\_dirname)  
 .filter((file) => {  
 return (  
 file.indexOf(".") !== 0 &&  
 file !== basename &&  
 file.slice(-3) === ".js" &&  
 file.indexOf(".test.js") === -1  
 );  
 })  
 .forEach((file) => {  
 if (  
 [  
 "AdrBlkFileRef.js",  
 "Application\_old.js",  
 "Application.js",  
 "Attachment.js",  
 "BsBlock.js",  
 "Case.js",  
 "Eminute.js",  
 "Notification.js",  
 "OAuthToken.js",  
 "Submission.js",  
 "SysFileRef.js",  
 "Task.js",  
 "User.js",  
 ].includes(file)  
 ) {  
 return;  
 }  
 const model = require(path.join(\_\_dirname, file))(  
 sequelize,  
 Sequelize.DataTypes  
 );  
 db[model.name] = model;  
 });  
Object.keys(db).forEach((modelName) => {  
 if (db[modelName].associate) {  
 db[modelName].associate(db);  
 }  
});  
db.sequelize = sequelize;  
db.Sequelize = Sequelize;  
module.exports = db;

## File: bd-scs-backend-backend/models/Notification.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const NotificationType = {  
 NEW\_TASK: "NEW\_TASK",  
 NEW\_EMINUTE: "NEW\_EMINUTE",  
};  
const NotificationSchema = new Schema({  
 submissionCase: { type: Schema.Types.ObjectId, ref: "Case" },  
 application: { type: Schema.Types.ObjectId, ref: "Application" },  
 task: { type: Schema.Types.ObjectId, ref: "Task" },  
 eminute: { type: Schema.Types.ObjectId, ref: "Eminute" },  
 user: String,  
 notificationType: String,  
 requireSendEmail: Boolean,  
 emailSentAt: { type: Date },  
 updatedAt: { type: Date },  
 createdAt: { type: Date, default: Date.now },  
});  
module.exports = NotificationSchema;  
module.exports.NotificationType = NotificationType;

## File: bd-scs-backend-backend/models/OAuthToken.js

const mongoose = require('mongoose');  
const Schema = mongoose.Schema;  
const OAuthTokenSchema = new Schema({  
 accessToken: String,  
 accessTokenExpiresAt: Date,  
 refreshToken: String,  
 refreshTokenExpiresAt: Date,  
 client: Object,  
 user: { type: Schema.Types.ObjectId, ref: 'User' },  
});  
module.exports = OAuthTokenSchema;

## File: bd-scs-backend-backend/models/Submission.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const SubmissionType = {  
 JOKC02: "JOKC02",  
 CCCAIMH3: "CCCAIMH/3",  
 CCCAIC2: "CCCAIC2",  
 A2: "A2",  
 B2: "B2",  
 C2: "C2",  
 E2: "E2",  
 F2: "F2",  
 G2: "G2",  
 "submit-doc": "submit-doc",  
 "inspect-req": "inspect-req",  
 "withdraw-app": "withdraw-app",  
 "appoint-ap": "appoint-ap",  
 "update-info": "update-info",  
};  
const SubmissionStatus = {  
 DRAFT: "DRAFT",  
};  
const SubmissionSchema = new Schema({  
 application: { type: Schema.Types.ObjectId, ref: "Application" },  
 assignedGR: { type: Schema.Types.ObjectId, ref: "User" },  
 assignedBS: { type: Schema.Types.ObjectId, ref: "User" },  
 submissionCase: { type: Schema.Types.ObjectId, ref: "Case" },  
 ApplicationNo: String,  
 SubmissionType: {  
 type: String,  
 enum: Object.values(SubmissionType),  
 },  
 FormName: String,  
 ApplicantName: String,  
 ApplicantTitle: String,  
 ApplicantNameEN: String,  
 ApplicantEmail: String,  
 ApplicantMobile: String,  
 ApplicantHKIC: String,  
 ApplicantPosition: String,  
 ApplicationDate: String,  
 ApplicantPostTitle: String,  
 ApplicantRepOrgEN: String,  
 ApplicantRepOrg: String,  
 ApplicantAddressEN: String,  
 ApplicantTel: String,  
 AddressOfPremiseEN: String,  
 AddressOfPremiseENFloor: String,  
 AddressOfPremiseENUnit: String,  
 AddressOfPremiseCN: String,  
 AddressOfPremiseCNFloor: String,  
 AddressOfPremiseCNUnit: String,  
 CentreNameEnglish: String,  
 CentreNameChinese: String,  
 PhoneOfSchool: String,  
 ContactPTitle: String,  
 ContactIdentity: String,  
 ContactPName: String,  
 ContactNameEnglish: String,  
 ContactNameChinese: String,  
 ApplicantAddress: String,  
 ContactPAddress: String,  
 ContactAddressLine1: String,  
 ContactAddressLine2: String,  
 ContactPEmail: String,  
 ContactPTel: String,  
 ContactPMobile: String,  
 ContactPFax: String,  
 ContactCertRegNo: String,  
 ContactRegExpiryDate: String,  
 NameOfSchoolEN: String,  
 NameOfSchoolCN: String,  
 DescriptionOfSchool: String,  
 SchoolPremisesAddressLine1: String,  
 SchoolPremisesAddressLine2: String,  
 OtherSchoolDesc: String,  
 BdRefNo: String,  
 AgeOfStudent: String,  
 EstimatedNoOfStudent: Number,  
 BDLetterRefDate: String,  
 ChineseNameOfAp1: String,  
 EnglishNameOfAp1: String,  
 EmailOfAp1: String,  
 MobileOfAp1: String,  
 RegTypeOfAp1: String,  
 CertRegNoOfAp1: String,  
 ChineseNameOfAp2: String,  
 EnglishNameOfAp2: String,  
 EmailOfAp2: String,  
 MobileOfAp2: String,  
 RegTypeOfAp2: String,  
 CertRegNoOfAp2: String,  
 ChineseNameOfRse: String,  
 EnglishNameOfRse: String,  
 EmailOfRse: String,  
 MobileOfRse: String,  
 RegTypeOfRse: String,  
 CertRegNoOfRse: String,  
 AmendmentOfLayoutPlan: String,  
 Status: {  
 type: String,  
 enum: Object.values(SubmissionStatus),  
 },  
 updatedAt: { type: Date },  
 createdAt: { type: Date, default: Date.now },  
});  
module.exports = SubmissionSchema;  
module.exports.SubmissionType = SubmissionType;

## File: bd-scs-backend-backend/models/Sys\_Meta\_Data\_T.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../utils/SQLDBHelper");  
class SysMetaDataModel extends Model {}  
SysMetaDataModel.init(  
 {  
 SYS\_META\_DATA\_ID: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 REC\_TYPE: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 CODE: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 PARENT\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 ENGLISH\_DESCRIPTION: {  
 type: DataTypes.STRING(2500),  
 allowNull: true,  
 },  
 CHINESE\_DESCRIPTION: {  
 type: DataTypes.STRING(2500),  
 allowNull: true,  
 },  
 IS\_ACTIVE: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ORDERING: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 CREATED\_DT: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 CREATED\_POST: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CREATED\_NAME: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CREATED\_SECTION: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 LAST\_MODIFIED\_DT: {  
 type: DataTypes.DATE,  
 allowNull: false,  
 },  
 LAST\_MODIFIED\_POST: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 LAST\_MODIFIED\_NAME: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 LAST\_MODIFIED\_SECTION: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 MULTI\_INSP\_FOR\_LSO\_IND: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 REC\_SRC: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "SysMetaDataT",  
 tableName: "Sys\_Meta\_Data\_T",  
 timestamps: false,  
 }  
);  
module.exports = SysMetaDataModel;

## File: bd-scs-backend-backend/models/SysFileRef.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const SysFileRefSchema = new Schema({  
 sysFileRefId: { type: String, required: true },  
 frefPref: { type: String, required: false },  
 frefSeq: { type: String, required: false },  
 frefYr: { type: String, required: false },  
 frefSuf: { type: String, required: false },  
 display: { type: String, required: false },  
 dvExceed: { type: String, required: false },  
 dvStatusDt: { type: Date, required: false },  
 createdDt: { type: Date, default: Date.now },  
 createdPost: { type: String, required: false },  
 createdName: { type: String, required: false },  
 createdSection: { type: String, required: false },  
 lastModifiedDt: { type: Date, default: Date.now },  
 lastModifiedPost: { type: String, required: false },  
 lastModifiedName: { type: String, required: false },  
 lastModifiedSection: { type: String, required: false },  
});  
module.exports = SysFileRefSchema;

## File: bd-scs-backend-backend/models/Task.js

const mongoose = require("mongoose");  
const { TASKS } = require("../config/task");  
const Schema = mongoose.Schema;  
const TaskType = {  
 ASSIGN\_GR: "ASSIGN\_GR",  
 GR\_HANDLE: "GR\_HANDLE",  
 DESK\_STUDY: "DESK\_STUDY",  
 INITIAL\_SITE\_INSPECTION: "INITIAL\_SITE\_INSPECTION",  
 SEARCH\_STRUCTURAL\_DATA: "SEARCH\_STRUCTURAL\_DATA",  
 ENDORSE\_SO\_EMINUTES: "ENDORSE\_SO\_EMINUTES",  
 ENDORSE\_TO\_EMINUTES: "ENDORSE\_TO\_EMINUTES",  
 ENDORSE\_SE\_EMINUTES: "ENDORSE\_SE\_EMINUTES",  
 ENDORSE\_BS\_EMINUTES: "ENDORSE\_BS\_EMINUTES",  
 PREPARE\_STRUCTURAL\_ADVICE: "PREPARE\_STRUCTURAL\_ADVICE",  
 CHECK\_PLAN\_AND\_SITE\_CONDITION: "CHECK\_PLAN\_AND\_SITE\_CONDITION",  
 PREPARE\_LETTER\_CERT: "PREPARE\_LETTER\_CERT",  
 ENDORSE\_LETTER\_CERT: "ENDORSE\_LETTER\_CERT",  
 ENDORSE\_OBJECTION\_SPECIAL\_CASE: "ENDORSE\_OBJECTION\_SPECIAL\_CASE",  
};  
const newTaskTypes = TASKS.map((task) => task.type).filter(  
 (type) => !Object.values(TaskType).includes(type)  
);  
const TaskStatus = {  
 COMPLETED: "COMPLETED",  
 ACTIVE: "ACTIVE",  
 INACTIVE: "INACTIVE",  
};  
const TaskSchema = new Schema({  
 name: String,  
 taskType: {  
 type: String,  
 enum: [...Object.values(TaskType), ...newTaskTypes],  
 },  
 status: {  
 type: String,  
 enum: Object.values(TaskStatus),  
 default: TaskStatus.INACTIVE,  
 },  
 application: {  
 type: Schema.Types.ObjectId,  
 ref: "Application",  
 },  
 submissionCase: {  
 type: Schema.Types.ObjectId,  
 ref: "Case",  
 },  
 user: String,  
 team: String,  
 updatedAt: { type: Date },  
 createdAt: { type: Date, default: Date.now },  
});  
module.exports = TaskSchema;  
module.exports.TaskType = TaskType;  
module.exports.TaskStatus = TaskStatus;

## File: bd-scs-backend-backend/models/User.js

const mongoose = require("mongoose");  
const Schema = mongoose.Schema;  
const USER\_TYPES = {  
 SUPER\_ADMIN: "SUPER\_ADMIN",  
 ADMIN: "ADMIN",  
 USER: "USER",  
};  
const ROLES = {  
 "Pre-GR": "Pre-GR",  
 GR: "GR",  
 SO: "SO",  
 BS: "BS",  
 SBS: "SBS",  
 CBS: "CBS",  
 TO: "TO",  
 SE: "SE",  
 SEE: "SEE",  
 SSE: "SSE",  
 EDB: "EDB",  
 SWD: "SWD",  
 "EDB-JOKC": "EDB-JOKC",  
};  
const DEPARTMENTS = {  
 BD: "BD",  
 EBD: "EBD",  
 SWD: "SWD",  
 EBDC: "EBDC",  
};  
const UserSchema = new Schema({  
 osdpLoginId: {  
 type: String,  
 unique: true,  
 },  
 name: String,  
 email: String,  
 password: String,  
 team: String,  
 role: {  
 type: String,  
 enum: Object.values(ROLES),  
 },  
 group: String,  
 phoneNumber: String,  
 address: String,  
 position: String,  
 department: {  
 type: String,  
 enum: Object.values(DEPARTMENTS),  
 },  
 delegateTo: String,  
 osdpEmail: String,  
 departmentId: String,  
 luPostName: String,  
 bdgis: String,  
 letterName: String,  
 letterNameCn: String,  
 letterPosition: String,  
 letterPositionCn: String,  
 letterLongPosition: String,  
 letterLongPositionCn: String,  
 notificationEmail: String,  
 userType: {  
 type: String,  
 enum: Object.values(USER\_TYPES),  
 },  
 lastLoginAt: Date,  
 lock: {  
 type: Boolean,  
 default: false,  
 },  
 resetPasswordToken: String,  
 resetPasswordTokenExpiresAt: Date,  
 updatedAt: { type: Date },  
 updatedBy: { type: Schema.Types.ObjectId, ref: "User" },  
 createdAt: { type: Date },  
});  
module.exports = UserSchema;  
module.exports.ROLES = ROLES;

## File: bd-scs-backend-backend/routes/applications.js

var express = require("express");  
var router = express.Router();  
const MongoDBHelper = require("../utils/MongoDBHelper");  
const collections = require("../config/collections");  
const { APPLICATION\_TYPES, SubmissionType } = require("../models/Application");  
const { TaskType } = require("../models/Task");  
const { generateApplicationNo } = require("../utils/application");  
const multer = require("multer");  
const path = require("path");  
const mongoose = require("mongoose");  
const ObjectId = mongoose.Types.ObjectId;  
const REPLY\_DAYS = require("../config/replyDays");  
require("dotenv").config();  
const { QueryTypes } = require("sequelize");  
function buildAddressEN(ad) {  
 const parts = [  
 ad.osadR\_E1,  
 ad.osadR\_E2,  
 ad.osadR\_E3,  
 ad.osadR\_E4,  
 ad.osadR\_E5,  
 ad.bldG\_NAME\_E1,  
 ad.osadR\_LOT\_E1,  
 ad.osadR\_LOT\_E2,  
 ad.osadR\_LOT\_E3,  
 ad.osadR\_LOT\_E4,  
 ad.osadR\_ST\_E1,  
 ad.osadR\_ST\_E2,  
 ad.district\_e,  
 ];  
 return parts.filter((part) => part && part.trim()).join(", ");  
}  
const storage = multer.diskStorage({  
 destination: function (req, file, cb) {  
 cb(null, "uploads/");  
 },  
 filename: function (req, file, cb) {  
 cb(null, Date.now() + path.extname(file.originalname));  
 },  
});  
const upload = multer({ storage: storage });  
const fs = require("fs");  
const { TASKS } = require("../config/task");  
const { BS\_TEAM\_MAPPINGS } = require("../config/user");  
const { NotificationType } = require("../models/Notification");  
const { AttachemntType } = require("../models/Attachment");  
const { default: axios } = require("axios");  
const { HttpsProxyAgent } = require('https-proxy-agent');  
const sequelize = require("../utils/SQLDBHelper");  
const { findAddress } = require("../utils/addressUtils");  
const httpsAgent = new HttpsProxyAgent('http://10.5.241.158:8080');  
const uploadsDir = "uploads";  
if (!fs.existsSync(uploadsDir)) {  
 fs.mkdirSync(uploadsDir);  
}  
router.get("/", async function (req, res, next) {  
 try {  
 const query = {};  
 if (req.query.ApplicationNo) {  
 query.ApplicationNo = { $regex: req.query.ApplicationNo, $options: "i" };  
 }  
 if (req.query.NameOfSchoolCN) {  
 query.NameOfSchoolCN = {  
 $regex: req.query.NameOfSchoolCN,  
 $options: "i",  
 };  
 }  
 if (req.query.NameOfSchoolEN) {  
 query.NameOfSchoolEN = {  
 $regex: req.query.NameOfSchoolEN,  
 $options: "i",  
 };  
 }  
 if (req.query.AddressOfPremiseCN) {  
 query.AddressOfPremiseCN = {  
 $regex: req.query.AddressOfPremiseCN,  
 $options: "i",  
 };  
 }  
 if (req.query.AddressOfPremiseEN) {  
 query.AddressOfPremiseEN = {  
 $regex: req.query.AddressOfPremiseEN,  
 $options: "i",  
 };  
 }  
 if (req.query.ApplicationType) {  
 query.ApplicationType = {  
 $regex: req.query.ApplicationType,  
 $options: "i",  
 };  
 }  
 const ApplicationModel = MongoDBHelper.getCollection(  
 collections.Application  
 );  
 console.log("Query", query);  
 const applications = await ApplicationModel.find(query).sort({  
 createdAt: -1,  
 });  
 res.send(applications);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post("/", async function (req, res, next) {  
 try {  
 const ApplicationModel = MongoDBHelper.getCollection(  
 collections.Application  
 );  
 console.log("Request", JSON.stringify(req.body));  
 const SubmissionModel = MongoDBHelper.getCollection(collections.Submission);  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 req.body.SubmissionType = SubmissionType.PAPER;  
 let assignedBS = null;  
 let assignedSBS = null;  
 let assignedGR = null;  
 if (req.body.RelatedPremise) {  
 const blockId = req.body.RelatedPremise;  
 const BsBlockModel = MongoDBHelper.getCollection(collections.BsBlock);  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 if (blockId) {  
 const matchBlockId = await BsBlockModel.findOne({ blockId: blockId });  
 assignedBS = (  
 await UserModel.findOne({  
 role: "BS",  
 bdgis: matchBlockId.bdgis,  
 })  
 )?.position;  
 if (assignedBS) {  
 assignedSBS = (  
 await UserModel.findOne({  
 team: assignedBS,  
 role: "SBS",  
 })  
 )?.position;  
 assignedGR = (  
 await UserModel.findOne({  
 team: assignedBS,  
 role: "GR",  
 })  
 )?.position;  
 }  
 }  
 }  
 const application = await ApplicationModel.create({  
 ...req.body,  
 ApplicationNo:  
 req.body.ApplicationNo ||  
 (await generateApplicationNo(req.body.ApplicationType)),  
 ApplicationDate: req.body.Date,  
 assignedBS,  
 assignedSBS,  
 assignedGR,  
 });  
 res.send(application);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post("/find-Address", async (req, res, next) => {  
 try {  
 let applicationReq = req.body;  
 let addresses = await findAddress(applicationReq);  
 let response = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: addresses,  
 };  
 return res.status(200).json(response);  
 } catch (err) {  
 next(err);  
 }  
});  
router.get("/:applicationId", async function (req, res, next) {  
 try {  
 const ApplicationModel = MongoDBHelper.getCollection(  
 collections.Application  
 );  
 const SysFileRefModel = MongoDBHelper.getCollection(collections.SysFileRef);  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 console.log("AID", req.params.applicationId);  
 const application = await ApplicationModel.findOne(  
 ObjectId.isValid(req.params.applicationId)  
 ? { \_id: req.params.applicationId }  
 : { applicationNo: req.params.applicationId }  
 )  
 .populate("assignedBS", "name")  
 .lean();  
 const sysFileRef = await SysFileRefModel.findOne({  
 sysFileRefId: application.FileReference,  
 });  
 const attachments = await AttachmentModel.find({  
 application: req.params.applicationId,  
 type: AttachemntType.APPLICATION,  
 });  
 if (application.RelatedPremise) {  
 const relatedPremiseDetails = (  
 await findAddress({  
 BuildingId: application.RelatedPremise,  
 })  
 )?.[0];  
 application.RelatedPremiseType = relatedPremiseDetails.bt\_e;  
 application.RelatedPremiseAddress = buildAddressEN(relatedPremiseDetails);  
 application.District = relatedPremiseDetails.district\_e;  
 application.Region = relatedPremiseDetails.region\_e;  
 application.Area = relatedPremiseDetails.area\_e;  
 }  
 application.FileReferenceDisplay = sysFileRef?.display;  
 application.attachments = attachments;  
 res.send(application);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/:applicationId/history", async function (req, res, next) {  
 try {  
 const ApplicationModel = MongoDBHelper.getCollection(  
 collections.Application  
 );  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const SysFileRefModel = MongoDBHelper.getCollection(collections.SysFileRef);  
 const application = await ApplicationModel.findOne({  
 \_id: req.params.applicationId,  
 }).lean();  
 const applicationHistory = await AttachmentModel.find({  
 application: req.params.applicationId,  
 sysFileRefId: application.FileReference,  
 efolio: { $ne: null },  
 });  
 const applicationHistoryWithFileRef = await Promise.all(  
 applicationHistory.map(async (attachment) => {  
 const fileRef = await SysFileRefModel.findOne({  
 sysFileRefId: attachment.sysFileRefId,  
 });  
 return {  
 ...attachment.toObject(),  
 fileReference: fileRef?.display,  
 };  
 })  
 );  
 res.send(applicationHistoryWithFileRef);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post("/:applicationId", async function (req, res, next) {  
 try {  
 const ApplicationModel = MongoDBHelper.getCollection(  
 collections.Application  
 );  
 if (req.body.RelatedPremise) {  
 const blockId = req.body.RelatedPremise;  
 const BsBlockModel = MongoDBHelper.getCollection(collections.BsBlock);  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 if (blockId) {  
 const matchBlockId = await BsBlockModel.findOne({ blockId: blockId });  
 req.body.assignedBS = (  
 await UserModel.findOne({  
 role: "BS",  
 bdgis: matchBlockId.bdgis,  
 })  
 )?.position;  
 if (req.body.assignedBS) {  
 req.body.assignedSBS = await UserModel.find({  
 team: req.body.assignedBS,  
 role: "SBS",  
 })?.position;  
 req.body.assignedGR = await UserModel.find({  
 team: req.body.assignedBS,  
 role: "GR",  
 })?.position;  
 }  
 }  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 const unhandledCases = await CaseModel.find({  
 application: req.params.applicationId,  
 caseOfficerReceive: null,  
 });  
 for (const caseItem of unhandledCases) {  
 await CaseModel.update(  
 { \_id: caseItem.\_id },  
 {  
 caseOfficerReceive: req.body.assignedBS,  
 caseOfficerReply: req.body.assignedBS,  
 seniorCaseOfficerReceive: req.body.assignedSBS,  
 seniorCaseOfficerReply: req.body.assignedSBS,  
 assignedGR: req.body.assignedGR,  
 team: req.body.assignedBS,  
 }  
 );  
 const teamMembers = await UserModel.find({  
 team: req.body.assignedBS,  
 });  
 const teamGroup = {  
 SO: teamMembers.find((user) => user.role == "SO"),  
 TO: teamMembers.find((user) => user.role == "TO"),  
 SE: teamMembers.find((user) => user.role == "SE"),  
 BS: teamMembers.find((user) => user.role == "BS"),  
 SSE: teamMembers.find((user) => user.role == "SSE"),  
 SBS: teamMembers.find((user) => user.role == "SBS"),  
 CBS: teamMembers.find((user) => user.role == "CBS"),  
 };  
 let catNature =  
 caseItem.Nature +  
 caseItem.Category +  
 (caseItem?.ViaSCS ? "VIASCS" : "");  
 if (!catNature || !possibleCatNature.includes(catNature)) {  
 catNature = "NEWSCH";  
 }  
 await TaskModel.deleteMany({ submissionCase: caseItem.\_id });  
 let newTasks = TASKS.filter((task) =>  
 task.catNature?.includes(catNature)  
 ).map((task) => {  
 let taskDetails = {  
 taskType: task.type,  
 application: application.\_id,  
 submissionCase: caseItem.\_id,  
 user:  
 (teamGroup && teamGroup[task.doneBy]?.position) || "UNASSIGNED",  
 team,  
 status: task.defaultStatus || "INACTIVE",  
 };  
 return taskDetails;  
 });  
 const insertedTasks = await TaskModel.insertMany(newTasks);  
 }  
 }  
 const application = await ApplicationModel.findOneAndUpdate(  
 { \_id: req.params.applicationId },  
 { ...req.body },  
 { new: true }  
 );  
 if (req.body.assignedGR) {  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 await TaskModel.update(  
 {  
 taskType: "CREATE\_APPLICATION\_CASE",  
 application: req.params.applicationId,  
 status: "ACTIVE",  
 },  
 { user: req.body.assignedGR }  
 );  
 }  
 console.log(application);  
 res.send(application);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/:applicationId/cases", async function (req, res, next) {  
 try {  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const SysFileRefModel = MongoDBHelper.getCollection(collections.SysFileRef);  
 const cases = await CaseModel.find({  
 application: req.params.applicationId,  
 })  
 .sort({ createdAt: -1 })  
 .populate("caseOfficerReceive");  
 const casesWithEfolio = await Promise.all(  
 cases.map(async (caseItem) => {  
 const efolioAttachment = await AttachmentModel.findOne({  
 submissionCase: caseItem.\_id,  
 type: AttachemntType.CREATE\_CASE,  
 });  
 const fileReference = await SysFileRefModel.findOne({  
 sysFileRefId: efolioAttachment?.sysFileRefId,  
 });  
 return {  
 ...caseItem.toObject(),  
 efolio: efolioAttachment?.efolio,  
 filePartNo: efolioAttachment?.filePartNo,  
 fileReference: fileReference?.display,  
 };  
 })  
 );  
 res.send(casesWithEfolio);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post("/:applicationId/cases", async function (req, res, next) {  
 try {  
 const ApplicationModel = MongoDBHelper.getCollection(  
 collections.Application  
 );  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 const NotificationModel = MongoDBHelper.getCollection(  
 collections.Notification  
 );  
 const application = await ApplicationModel.findOne({  
 \_id: req.params.applicationId,  
 });  
 if (application) {  
 let team = "UNASSIGNED";  
 let teamGroup = null;  
 if (req.body.caseOfficerReceive) {  
 team = req.body.caseOfficerReceive;  
 const teamMembers = await UserModel.find({  
 team,  
 });  
 teamGroup = {  
 SO: teamMembers.find((user) => user.role == "SO"),  
 TO: teamMembers.find((user) => user.role == "TO"),  
 SE: teamMembers.find((user) => user.role == "SE"),  
 BS: teamMembers.find((user) => user.role == "BS"),  
 SSE: teamMembers.find((user) => user.role == "SSE"),  
 SBS: teamMembers.find((user) => user.role == "SBS"),  
 CBS: teamMembers.find((user) => user.role == "CBS"),  
 };  
 }  
 const possibleCatNature = [  
 ...new Set(TASKS.map((task) => task.catNature).flat()),  
 ];  
 const { Nature, Category, ViaSCS } = req.body;  
 let catNature = Nature + Category + (ViaSCS ? "VIASCS" : "");  
 console.log("Before catNature", catNature);  
 if (!catNature || !possibleCatNature.includes(catNature)) {  
 catNature = "NEWSCH";  
 }  
 const targetReplyDays = REPLY\_DAYS[catNature] || 7;  
 console.log("Target Reply Days", targetReplyDays);  
 let targetReplyDate = new Date();  
 targetReplyDate.setDate(targetReplyDate.getDate() + targetReplyDays);  
 console.log("Target Reply Date", targetReplyDate);  
 const caseItem = await CaseModel.create({  
 ...req.body,  
 application: req.params.applicationId,  
 caseOfficerReply: req.body.caseOfficerReceive,  
 seniorCaseOfficerReply: req.body.seniorCaseOfficerReceive,  
 team: team,  
 TargetReplyDate: targetReplyDate,  
 });  
 console.log("CatNature", catNature);  
 let newTasks = TASKS.filter((task) =>  
 task.catNature?.includes(catNature)  
 ).map((task) => {  
 let taskDetails = {  
 taskType: task.type,  
 application: application.\_id,  
 submissionCase: caseItem.\_id,  
 user: (teamGroup && teamGroup[task.doneBy]?.position) || "UNASSIGNED",  
 team,  
 status: task.defaultStatus || "INACTIVE",  
 };  
 return taskDetails;  
 });  
 const insertedTasks = await TaskModel.insertMany(newTasks);  
 const notifications = insertedTasks.map((task) => ({  
 user: task.user,  
 notificationType: NotificationType.NEW\_TASK,  
 task: task.\_id,  
 requireSendEmail: false,  
 }));  
 const insertedNotifications = await NotificationModel.insertMany(  
 notifications  
 );  
 console.log(caseItem);  
 const AttachmentModel = MongoDBHelper.getCollection(  
 collections.Attachment  
 );  
 console.log("Application File Reference", application.FileReference);  
 const efolioCount = await AttachmentModel.countDocuments({  
 sysFileRefId: application.FileReference,  
 });  
 const attachment = await AttachmentModel.create({  
 application: application.\_id,  
 submissionCase: caseItem.\_id,  
 sysFileRefId: application.FileReference,  
 efolio: "" + (efolioCount + 1),  
 type: AttachemntType.CREATE\_CASE,  
 receivedDate: new Date(),  
 });  
 console.log(attachment);  
 res.send(caseItem);  
 } else {  
 next({ status: 400, message: "Application not ready to create case" });  
 }  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
module.exports = router;

## File: bd-scs-backend-backend/routes/attachments.js

require("dotenv").config();  
var express = require("express");  
var router = express.Router();  
const MongoDBHelper = require("../utils/MongoDBHelper");  
const collections = require("../config/collections");  
const { hkpostSign } = require("../utils/hkpostUtils");  
const fs = require("fs");  
const multer = require("multer");  
const path = require("path");  
const { AttachemntType } = require("../models/Attachment");  
const { lookup } = require("mime-types");  
const axios = require("axios");  
const { HttpsProxyAgent } = require('https-proxy-agent');  
const httpsAgent = new HttpsProxyAgent('http://10.5.241.158:8080');  
const sendEmail = require("../utils/sendEmail");  
const DocxMerger = require("docx-merger");  
const docx4js = require("docx4js");  
const WordExtractor = require("word-extractor");  
const storage = multer.diskStorage({  
 destination: function (req, file, cb) {  
 cb(null, "uploads/");  
 },  
 filename: function (req, file, cb) {  
 cb(null, Date.now() + path.extname(file.originalname));  
 },  
});  
const upload = multer({ storage: storage });  
const FRONTEND\_API\_URL = process.env.FRONTEND\_API\_URL;  
router.post("/", upload.single("file"), async function (req, res, next) {  
 try {  
 if (!req.file) {  
 return res.status(400).send("No file uploaded.");  
 }  
 const ApplicationModel = MongoDBHelper.getCollection(  
 collections.Application  
 );  
 const application = await ApplicationModel.findOne({  
 \_id: req.body.application,  
 });  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const existingFiles = await AttachmentModel.find({  
 application: req.body.application,  
 type: req.body.type,  
 });  
 let efolio = null;  
 if (  
 req.body.type === AttachemntType.CASE ||  
 req.body.type === AttachemntType.PREPARE\_LETTER  
 ) {  
 efolio = null;  
 } else if(req.body.type === AttachemntType.ISSUE\_LETTER && existingFiles.length > 0) {  
 if (req.file.mimetype !== 'application/vnd.openxmlformats-officedocument.wordprocessingml.document') {  
 return res.status(400).send("Uploaded file is not a DOCX file.");  
 }  
 } else {  
 const efolioCount = await AttachmentModel.countDocuments({  
 sysFileRefId: application.FileReference,  
 efolio: { $ne: null },  
 });  
 efolio = "" + (efolioCount + 1);  
 }  
 const attachment = await AttachmentModel.create({  
 ...req.body,  
 sysFileRefId: application.FileReference,  
 efolio,  
 file: req.file,  
 receivedDate: new Date(),  
 });  
 console.log(attachment);  
 res.send(attachment);  
 } catch (err) {  
 next(err);  
 }  
});  
/\* Update attachment. \*/  
router.post(  
 "/:attachmentId",  
 upload.single("file"),  
 async function (req, res, next) {  
 try {  
 const AttachmentModel = MongoDBHelper.getCollection(  
 collections.Attachment  
 );  
 const existingFiles = await AttachmentModel.find({  
 application: req.body.application,  
 type: req.body.type,  
 });  
 if(req.body.type === AttachemntType.ISSUE\_LETTER && existingFiles.length > 0) {  
 if (req.file.mimetype !== 'application/vnd.openxmlformats-officedocument.wordprocessingml.document') {  
 return res.status(400).send("Uploaded file is not a DOCX file.");  
 }  
 const formData = new FormData();  
 const file = new Blob([fs.readFileSync(req.file.path)], {  
 type: lookup(req.file.path),  
 });  
 formData.append("file", file, req.file.filename);  
 const response = await axios.post(`${process.env.WORD\_CONTROLLER\_URL}/WordAnalysis/upload`, formData, {  
 headers: {  
 'Content-Type': 'multipart/form-data'  
 }  
 });  
 const trimmedMarkers = response.data;  
 Object.keys(trimmedMarkers).forEach(key => {  
 trimmedMarkers[key] = trimmedMarkers[key].replace(/^\(\w+\)/g, '').trim();  
 });  
 // console.log(analysisResult);  
 // const extractor = new WordExtractor();  
 // const docxObj = await extractor.extract(req.file.path);  
 // const docxText = docxObj.getBody();  
 // console.log(docxText);  
 //  
 const ApplicationModel = MongoDBHelper.getCollection(collections.Application);  
 const application = await ApplicationModel.findOne({ \_id: req.body.application });  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const caseData = await CaseModel.findOne({ application: req.body.application });  
 // // TEMP: find an \_ in the docx text  
 // const markers = docxText.match(/\_.+\_/g);  
 // console.log(markers);  
 // const trimmedMarkers = markers.map(marker => marker.replace(/\_/g, ''));  
 console.log(trimmedMarkers);  
 // a. find the corresponding Proforma question  
 // b. update the content  
 // here we assume the list of trimmed markers is the same as the list of options previously selected  
 // so we will loop through the options BS selected  
 let markersIndex = 0;  
 for(let markerKey of Object.keys(trimmedMarkers)) {  
 // we care only those markers have \_ because all proforma questions have \_ in the text  
 if(!markerKey.includes('\_')) {  
 continue;  
 }  
 let markerText = trimmedMarkers[markerKey];  
 if(markerText.includes('\t')) {  
 markerText = markerText.split('\t')[1];  
 }  
 let markerKeyParts = markerKey.split('\_');  
 switch(markerKeyParts[0]) {  
 case 'cat1':  
 switch(markerKeyParts[1]) {  
 case 'ubw':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'q3':  
 switch(markerKeyParts[4]) {  
 case 'cat1':  
 if(!caseData.ubw.q1.q3.cat1\_text) {  
 caseData.ubw.q1.q3.cat1\_text = {};  
 }  
 caseData.ubw.q1.q3.cat1\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat1other':  
 caseData.ubw.q1.q3.cat1\_other = trimmedMarkers[markerKey];  
 break;  
 case 'cat11':  
 if(!caseData.ubw.q1.q3.cat1\_1\_text) {  
 caseData.ubw.q1.q3.cat1\_1\_text = {};  
 }  
 caseData.ubw.q1.q3.cat1\_1\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat11other':  
 caseData.ubw.q1.q3.cat1\_1\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'frc':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'q2b':  
 switch(markerKeyParts[4]) {  
 case 'cat1':  
 if(!caseData.frc.q1.q2b.cat1\_text) {  
 caseData.frc.q1.q2b.cat1\_text = {};  
 }  
 caseData.frc.q1.q2b.cat1\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat1other':  
 caseData.frc.q1.q2b.cat1\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 case 'q2':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'cat1':  
 if(!caseData.frc.q2.q1.cat1\_text) {  
 caseData.frc.q2.q1.cat1\_text = {};  
 }  
 caseData.frc.q2.q1.cat1\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat1other':  
 caseData.frc.q2.q1.cat1\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 }  
 break;  
 case 'structsch':  
 switch(markerKeyParts[2]) {  
 case 'q5':  
 switch(markerKeyParts[3]) {  
 case 'optionscat1':  
 if(!caseData.structural\_schnlhkinds.q5.options\_cat1\_text) {  
 caseData.structural\_schnlhkinds.q5.options\_cat1\_text = {};  
 }  
 caseData.structural\_schnlhkinds.q5.options\_cat1\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat1other':  
 caseData.structural\_schnlhkinds.q5.cat1\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 case 'q6':  
 switch(markerKeyParts[3]) {  
 case 'optionscat1':  
 if(!caseData.structural\_schnlhkinds.q6.options\_cat1\_text) {  
 caseData.structural\_schnlhkinds.q6.options\_cat1\_text = {};  
 }  
 caseData.structural\_schnlhkinds.q6.options\_cat1\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat1other':  
 caseData.structural\_schnlhkinds.q6.cat1\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'mws':  
 switch(markerKeyParts[1]) {  
 case 'structsch':  
 switch(markerKeyParts[2]) {  
 case 'q4':  
 switch(markerKeyParts[3]) {  
 case 'optionscat11':  
 if(!caseData.structural\_schnlhkinds.q4.options\_cat1\_1\_text) {  
 caseData.structural\_schnlhkinds.q4.options\_cat1\_1\_text = {};  
 }  
 caseData.structural\_schnlhkinds.q4.options\_cat1\_1\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'optionscat12':  
 if(!caseData.structural\_schnlhkinds.q4.options\_cat1\_2\_text) {  
 caseData.structural\_schnlhkinds.q4.options\_cat1\_2\_text = {};  
 }  
 caseData.structural\_schnlhkinds.q4.options\_cat1\_2\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'optionscat13':  
 if(!caseData.structural\_schnlhkinds.q4.options\_cat1\_3\_text) {  
 caseData.structural\_schnlhkinds.q4.options\_cat1\_3\_text = {};  
 }  
 caseData.structural\_schnlhkinds.q4.options\_cat1\_3\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'optionscat14':  
 if(!caseData.structural\_schnlhkinds.q4.options\_cat1\_4\_text) {  
 caseData.structural\_schnlhkinds.q4.options\_cat1\_4\_text = {};  
 }  
 caseData.structural\_schnlhkinds.q4.options\_cat1\_4\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'othercat14':  
 caseData.structural\_schnlhkinds.q4.other\_cat1\_4 = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 case 'ubw':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'q3':  
 switch(markerKeyParts[4]) {  
 case 'cat11':  
 if(!caseData.ubw.q1.q3.cat1\_1\_text) {  
 caseData.ubw.q1.q3.cat1\_1\_text = {};  
 }  
 caseData.ubw.q1.q3.cat1\_1\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'ubw':  
 switch(markerKeyParts[1]) {  
 case 'q2':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'options':  
 switch(markerKeyParts[4]) {  
 default:  
 if(!caseData.ubw.q2.q1.options\_text) {  
 caseData.ubw.q2.q1.options\_text = {};  
 }  
 caseData.ubw.q2.q1.options\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 case 'other':  
 caseData.ubw.q2.q1.other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'c21a':  
 switch(markerKeyParts[1]) {  
 case 'ubw':  
 switch(markerKeyParts[2]) {  
 case 'q2':  
 switch(markerKeyParts[3]) {  
 case 'q2':  
 switch(markerKeyParts[4]) {  
 case 'cat21a':  
 if(!caseData.ubw.q2.q2.cat2\_1a\_text) {  
 caseData.ubw.q2.q2.cat2\_1a\_text = {};  
 }  
 caseData.ubw.q2.q2.cat2\_1a\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 case 'othercat21a':  
 caseData.ubw.q2.q2.cat2\_1a\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'misc':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'cat21aother':  
 caseData.misc.q1.cat2\_1a\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'c21b':  
 switch(markerKeyParts[1]) {  
 case 'ubw':  
 switch(markerKeyParts[2]) {  
 case 'q2':  
 switch(markerKeyParts[3]) {  
 case 'q1':  
 switch(markerKeyParts[4]) {  
 case 'option':  
 if(!caseData.ubw.q2.q1.option\_text) {  
 caseData.ubw.q2.q1.option\_text = {};  
 }  
 caseData.ubw.q2.q1.option\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 case 'other':  
 caseData.ubw.q2.q1.other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'misc':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'cat21bother':  
 caseData.misc.q1.cat2\_1b\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'c22a':  
 switch(markerKeyParts[1]) {  
 case 'structsch':  
 switch(markerKeyParts[2]) {  
 case 'q3':  
 switch(markerKeyParts[3]) {  
 case 'optionscat22a':  
 if(!caseData.structural\_schnlhkinds.q3.options\_cat2\_2a\_text) {  
 caseData.structural\_schnlhkinds.q3.options\_cat2\_2a\_text = {};  
 }  
 caseData.structural\_schnlhkinds.q3.options\_cat2\_2a\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'othercat22a':  
 caseData.structural\_schnlhkinds.q3.other\_cat2\_2a = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 case 'misc':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'cat22aother':  
 caseData.misc.q1.cat2\_2a\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'c23b':  
 switch(markerKeyParts[1]) {  
 case 'frc':  
 switch(markerKeyParts[2]) {  
 case 'q3':  
 switch(markerKeyParts[3]) {  
 case 'cat23b':  
 if(!caseData.frc.q3.cat2\_3b\_text) {  
 caseData.frc.q3.cat2\_3b\_text = {};  
 }  
 caseData.frc.q3.cat2\_3b\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'othercat23b':  
 caseData.frc.q3.cat2\_3b\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'c24':  
 switch(markerKeyParts[1]) {  
 case 'moe':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'q9':  
 switch(markerKeyParts[4]) {  
 case 'cat24':  
 if(!caseData.moe.q1.q9.cat2\_4\_text) {  
 caseData.moe.q1.q9.cat2\_4\_text = {};  
 }  
 caseData.moe.q1.q9.cat2\_4\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat24other':  
 caseData.moe.q1.q9.cat2\_4\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 case 'q2':  
 switch(markerKeyParts[3]) {  
 case 'q1':  
 switch(markerKeyParts[4]) {  
 case 'cat24':  
 if(!caseData.moe.q2.q1.cat2\_4\_text) {  
 caseData.moe.q2.q1.cat2\_4\_text = {};  
 }  
 caseData.moe.q2.q1.cat2\_4\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat24other':  
 caseData.moe.q2.q1.cat2\_4\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 case 'q3':  
 switch(markerKeyParts[4]) {  
 case 'cat24':  
 if(!caseData.moe.q2.q3.cat2\_4\_text) {  
 caseData.moe.q2.q3.cat2\_4\_text = {};  
 }  
 caseData.moe.q2.q3.cat2\_4\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat24other':  
 caseData.moe.q2.q3.cat2\_4\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'ubw':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'q3':  
 switch(markerKeyParts[4]) {  
 case 'cat24':  
 if(!caseData.ubw.q1.q3.cat2\_4\_text) {  
 caseData.ubw.q1.q3.cat2\_4\_text = {};  
 }  
 caseData.ubw.q1.q3.cat2\_4\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat24other':  
 caseData.ubw.q1.q3.cat2\_4\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 case 'q3':  
 switch(markerKeyParts[3]) {  
 case 'cat24':  
 if(!caseData.ubw.q1.q3.cat2\_4\_text) {  
 caseData.ubw.q1.q3.cat2\_4\_text = {};  
 }  
 caseData.ubw.q1.q3.cat2\_4\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat24other':  
 caseData.ubw.q1.q3.cat2\_4\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 case 'c3':  
 switch(markerKeyParts[1]) {  
 case 'ubw':  
 switch(markerKeyParts[2]) {  
 case 'q4':  
 switch(markerKeyParts[3]) {  
 case 'cat3':  
 if(!caseData.ubw.q4.cat2\_3\_text) {  
 caseData.ubw.q4.cat2\_3\_text = {};  
 }  
 caseData.ubw.q4.cat2\_3\_text[markerKeyParts[4]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat3other':  
 caseData.ubw.q4.cat2\_3\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 case 'moe':  
 switch(markerKeyParts[2]) {  
 case 'q1':  
 switch(markerKeyParts[3]) {  
 case 'q9':  
 switch(markerKeyParts[4]) {  
 case 'cat3':  
 if(!caseData.moe.q1.q9.cat2\_3\_text) {  
 caseData.moe.q1.q9.cat2\_3\_text = {};  
 }  
 caseData.moe.q1.q9.cat2\_3\_text[markerKeyParts[5]] = trimmedMarkers[markerKey];  
 break;  
 case 'cat3other':  
 caseData.moe.q1.q9.cat2\_3\_other = trimmedMarkers[markerKey];  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 break;  
 }  
 }  
 await caseData.save();  
 }  
 let update = req.body;  
 delete update.file;  
 if (req.file) {  
 update.file = req.file;  
 }  
 console.log(update, "update attachment");  
 const attachment = await AttachmentModel.findOneAndUpdate(  
 { \_id: req.params.attachmentId },  
 update,  
 { new: true }  
 );  
 console.log(attachment);  
 const formData = new FormData();  
 const file = new Blob([fs.readFileSync(req.file.path)], {  
 type: lookup(req.file.path),  
 });  
 formData.append("file", file, req.file.filename);  
 const response = await axios.post(`${process.env.WORD\_CONTROLLER\_URL}/WordConvert/pdf`, formData, {  
 headers: {  
 'Content-Type': 'multipart/form-data'  
 },  
 responseType: 'arraybuffer',  
 });  
 const timestamp = Date.now();  
 const pdfPath = path.join("uploads", `${req.file.filename.replace(".docx", "")}\_${timestamp}.pdf`);  
 await fs.promises.writeFile(pdfPath, response.data);  
 console.log(update, "update attachment");  
 const result = await AttachmentModel.findOneAndUpdate(  
 {  
 submissionCase: update.submissionCase,  
 type: AttachemntType.ISSUE\_LETTER,  
 "file.mimetype": "application/pdf"  
 },  
 {  
 $set: {  
 file: {  
 fieldname: "file",  
 originalname: `${req.file.filename.replace(".docx", "")}\_${timestamp}.pdf`,  
 encoding: "7bit",  
 mimetype: "application/pdf",  
 destination: "uploads/",  
 filename: `${req.file.filename.replace(".docx", "")}\_${timestamp}.pdf`,  
 path: pdfPath,  
 size: response.data.length,  
 }  
 }  
 },  
 { new: true }  
 );  
 console.log(result);  
 res.send(attachment);  
 } catch (err) {  
 next(err);  
 }  
 }  
);  
router.delete("/:attachmentId", async function (req, res, next) {  
 try {  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const attachment = await AttachmentModel.deleteOne({  
 \_id: req.params.attachmentId,  
 });  
 res.send(attachment);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/", async function (req, res, next) {  
 try {  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const attachments = await AttachmentModel.find({ ...req.query });  
 console.log(attachments);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/:attachmentId", async function (req, res, next) {  
 try {  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const attachment = await AttachmentModel.findOne({  
 \_id: req.params.attachmentId,  
 });  
 if (!attachment) {  
 return res.status(404).send("Attachment not found");  
 }  
 console.log(attachment);  
 res.setHeader(  
 "Content-Disposition",  
 `attachment; filename=${attachment.file.originalname}`  
 );  
 res.setHeader("Content-Type", attachment.file.mimetype);  
 var s = fs.createReadStream(attachment.file.path).on("error", (error) => {  
 console.log("Read file error");  
 });  
 s.pipe(res);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post(  
 "/sign/:attachmentId",  
 upload.single("file"),  
 async function (req, res, next) {  
 try {  
 const AttachmentModel = MongoDBHelper.getCollection(  
 collections.Attachment  
 );  
 const attachment = await AttachmentModel.findOne({  
 \_id: req.params.attachmentId,  
 }).populate("application");  
 if (!attachment) {  
 return res.status(404).send("Attachment not found");  
 }  
 if (!attachment.application) {  
 return res.status(400).send("Attachment does not have an application");  
 }  
 const application = attachment.application;  
 let sign = req.body;  
 delete sign.file;  
 console.log(req.file, "req.file");  
 if (req.file) {  
 sign.file = req.file;  
 }  
 await hkpostSign(attachment, sign);  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 await TaskModel.updateOne(  
 {  
 application: application.\_id,  
 type: "LETTER\_SIGNATURE",  
 status: "ACTIVE",  
 },  
 { status: "COMPLETED" }  
 );  
 const formData = new FormData();  
 formData.append("applicationNo", application.ApplicationNo);  
 formData.append("submissionId", attachment.submissionCase);  
 const file = new Blob([fs.readFileSync(attachment.file.path)], {  
 type: lookup(attachment.file.path),  
 });  
 formData.append("file", file, "Prepare\_Letter.pdf");  
 formData.append("updateType", "Prepare Letter");  
 formData.append("stage", 3);  
 await axios.request({  
 url: `${FRONTEND\_API\_URL}/application/backend-update`,  
 method: 'post',  
 data: formData,  
 headers: { "Content-Type": "multipart/form-data" },  
 httpsAgent  
 })  
 await sendEmail(  
 application.ApplicantEmail,  
 `New reply: ${application.ApplicationNo}`,  
 `  
Dear Sir/Madam,  
Please login LSCP to view the application reply.  
The application number is ${application.ApplicationNo}.  
For enquiry, please contact us at telephone no. 2626 1616 (Handled by “1823”) or via email non-reply\_LSCP@bd.gov.hk.  
Please do not reply to this email as it comes from an automated mailbox.  
Buildings Department  
 `  
 );  
 res.send("Signed");  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
 }  
);  
router.post("/:caseId/resetLetter", async function (req, res, next) {  
 try {  
 console.log("caseId", req.params.caseId);  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const caseItem = await CaseModel.findOne({ \_id: req.params.caseId });  
 if (caseItem) {  
 if(caseItem.deck\_study) {  
 if(caseItem.deck\_study.q5) {  
 caseItem.deck\_study.q5.answer\_cat1\_text = undefined;  
 }  
 if(caseItem.deck\_study.q4) {  
 caseItem.deck\_study.q4.options\_cat2\_3a\_text = undefined;  
 }  
 }  
 if(caseItem.frc) {  
 if(caseItem.frc.q1) {  
 if(caseItem.frc.q1.q2a) {  
 caseItem.frc.q1.q2a.cat3\_text = undefined;  
 }  
 if(caseItem.frc.q1.q2b) {  
 caseItem.frc.q1.q2b.cat1\_text = undefined;  
 }  
 }  
 if(caseItem.frc.q2) {  
 if(caseItem.frc.q2.q1) {  
 caseItem.frc.q2.q1.cat1\_text = undefined;  
 }  
 }  
 if(caseItem.frc.q3) {  
 if(caseItem.frc.q3.cat2\_3b\_text) {  
 caseItem.frc.q3.cat2\_3b\_text = undefined;  
 }  
 }  
 if(caseItem.frc.q4) {  
 if(caseItem.frc.q4.cat2\_3b\_text) {  
 caseItem.frc.q4.cat2\_3b\_text = undefined;  
 }  
 }  
 }  
 if(caseItem.misc) {  
 if(caseItem.misc.q1) {  
 if(caseItem.misc.q1.cat1) {  
 caseItem.misc.q1.cat1\_text = undefined;  
 }  
 if(caseItem.misc.q1.cat1\_1) {  
 caseItem.misc.q1.cat1\_1\_text = undefined;  
 }  
 }  
 }  
 if(caseItem.structural\_schnlhkinds) {  
 if(caseItem.structural\_schnlhkinds.q1) {  
 caseItem.structural\_schnlhkinds.q1.options\_cat3\_text = undefined;  
 }  
 if(caseItem.structural\_schnlhkinds.q3) {  
 caseItem.structural\_schnlhkinds.q3.options\_cat2\_2a\_text = undefined;  
 }  
 if(caseItem.structural\_schnlhkinds.q4) {  
 caseItem.structural\_schnlhkinds.q4.options\_cat1\_1\_text = undefined;  
 }  
 if(caseItem.structural\_schnlhkinds.q4) {  
 caseItem.structural\_schnlhkinds.q4.options\_cat1\_2\_text = undefined;  
 }  
 if(caseItem.structural\_schnlhkinds.q4) {  
 caseItem.structural\_schnlhkinds.q4.options\_cat1\_3\_text = undefined;  
 }  
 if(caseItem.structural\_schnlhkinds.q4) {  
 caseItem.structural\_schnlhkinds.q4.options\_cat1\_4\_text = undefined;  
 }  
 if(caseItem.structural\_schnlhkinds.q5) {  
 caseItem.structural\_schnlhkinds.q5.options\_cat1\_text = undefined;  
 }  
 if(caseItem.structural\_schnlhkinds.q6) {  
 caseItem.structural\_schnlhkinds.q6.options\_cat1\_text = undefined;  
 }  
 }  
 if(caseItem.ubw) {  
 if(caseItem.ubw.q1) {  
 caseItem.ubw.q1.options\_text = undefined;  
 if(caseItem.ubw.q1.q3) {  
 caseItem.ubw.q1.q3.cat1\_1\_text = undefined;  
 }  
 if(caseItem.ubw.q1.q3) {  
 caseItem.ubw.q1.q3.cat1\_text = undefined;  
 }  
 if(caseItem.ubw.q1.q3) {  
 caseItem.ubw.q1.q3.cat2\_4\_text = undefined;  
 }  
 }  
 if(caseItem.ubw.q2) {  
 if(caseItem.ubw.q2.q1) {  
 caseItem.ubw.q2.q1.option\_text = undefined;  
 }  
 if(caseItem.ubw.q2.q2) {  
 caseItem.ubw.q2.q2.cat2\_1a\_text = undefined;  
 }  
 }  
 if(caseItem.ubw.q3) {  
 if(caseItem.ubw.q3) {  
 caseItem.ubw.q3.cat2\_4\_text = undefined;  
 }  
 }  
 if(caseItem.ubw.q4) {  
 caseItem.ubw.q4.cat1\_text = undefined;  
 caseItem.ubw.q4.cat3\_text = undefined;  
 }  
 }  
 console.log('Modified caseItem:', caseItem.toJSON());  
 await caseItem.save();  
 res.send("Letter reset");  
 } else {  
 next({ status: 400, message: "Case not found" });  
 }  
 } catch (err) {  
 next(err);  
 }  
});  
router.post("/:caseId/issueLetter", async function (req, res, next) {  
 try {  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const caseItem = await CaseModel.findOne({ \_id: req.params.caseId });  
 if (caseItem) {  
 const attachments = await AttachmentModel.find({  
 type: "PREPARE\_LETTER",  
 submissionCase: req.params.caseId,  
 }).sort({ subType: 1 });  
 const pdfObject = await mergeDocxToPdf(attachments);  
 if (pdfObject) {  
 await AttachmentModel.deleteMany(  
 {  
 type: "ISSUE\_LETTER",  
 subType: "ISSUE\_LETTER",  
 application: caseItem.application,  
 submissionCase: req.params.caseId,  
 });  
 await AttachmentModel.create(  
 {  
 type: "ISSUE\_LETTER",  
 subType: "ISSUE\_LETTER",  
 application: caseItem.application,  
 submissionCase: req.params.caseId,  
 file: pdfObject[0],  
 }  
 );  
 await AttachmentModel.create(  
 {  
 type: "ISSUE\_LETTER",  
 subType: "ISSUE\_LETTER",  
 application: caseItem.application,  
 submissionCase: req.params.caseId,  
 file: pdfObject[1],  
 }  
 );  
 res.send("Letter issued");  
 } else {  
 next({ status: 400, message: "Cannot generate PDF" });  
 }  
 } else {  
 next({ status: 400, message: "Case not found" });  
 }  
 } catch (err) {  
 next(err);  
 }  
});  
const mergeDocxToPdf = async (attachments) => {  
 const fs = require("fs");  
 const path = require("path");  
 const { promisify } = require("util");  
 const readFile = promisify(fs.readFile);  
 const writeFile = promisify(fs.writeFile);  
 const libre = require("libreoffice-convert");  
 const { Document, Packer, patchDocument } = require("docx");  
 const { default: PDFMerger } = await import("pdf-merger-js");  
 const pdfs = [];  
 const docxs = [];  
 const mergeDocxFormData = new FormData();  
 for (let i = 0; i < attachments.length; i++) {  
 const attachment = attachments[i];  
 const docxBuffer = await readFile(attachment.file.path);  
 const formData = new FormData();  
 const file = new Blob([fs.readFileSync(attachment.file.path)], {  
 type: lookup(attachment.file.path),  
 });  
 formData.append("file", file, attachment.file.filename);  
 const response = await axios.post(`${process.env.WORD\_CONTROLLER\_URL}/WordConvert/pdf`, formData, {  
 headers: {  
 'Content-Type': 'multipart/form-data'  
 },  
 responseType: 'arraybuffer',  
 });  
 const timestamp = Date.now();  
 const pdfPath = path.join("uploads", `${attachment.file.filename.replace(".docx", "")}\_${timestamp}.pdf`);  
 await fs.promises.writeFile(pdfPath, response.data);  
 pdfs.push(pdfPath);  
 docxs.push(fs.readFileSync(attachment.file.path));  
 const docxfile = new Blob([fs.readFileSync(attachment.file.path)], {  
 type: lookup(attachment.file.path),  
 });  
 mergeDocxFormData.append("files", docxfile, attachment.file.filename);  
 }  
 const pdfMergeInstance = new PDFMerger();  
 for (const pdf of pdfs) {  
 const pdfBuffer = await readFile(pdf);  
 await pdfMergeInstance.add(pdfBuffer);  
 }  
 const mergedPdfBuffer = await pdfMergeInstance.saveAsBuffer();  
 pdfs.forEach((pdf) => {  
 fs.unlinkSync(pdf);  
 });  
 const timestamp = Date.now();  
 const pdfPath = path.join("uploads", `issue ${timestamp}.pdf`);  
 await writeFile(pdfPath, mergedPdfBuffer);  
 const response = await axios.post(`${process.env.WORD\_CONTROLLER\_URL}/WordCombine/combine`, mergeDocxFormData, {  
 headers: {  
 'Content-Type': 'multipart/form-data'  
 },  
 responseType: 'arraybuffer',  
 });  
 const mergedDocPath = path.join("uploads", `issue ${timestamp}.docx`);  
 await fs.promises.writeFile(mergedDocPath, response.data);  
 const mergedDocSize = fs.statSync(mergedDocPath).size;  
 return [  
 {  
 fieldname: "file",  
 originalname: `issue ${timestamp}.docx`,  
 encoding: "7bit",  
 mimetype: "application/vnd.openxmlformats-officedocument.wordprocessingml.document",  
 destination: "uploads/",  
 filename: `issue ${timestamp}.docx`,  
 path: mergedDocPath,  
 size: mergedDocSize,  
 },  
 {  
 fieldname: "file",  
 originalname: `issue ${timestamp}.pdf`,  
 encoding: "7bit",  
 mimetype: "application/pdf",  
 destination: "uploads/",  
 filename: `issue ${timestamp}.pdf`,  
 path: pdfPath,  
 size: mergedPdfBuffer.length,  
 }  
 ];  
};  
module.exports = router;

## File: bd-scs-backend-backend/routes/auth.js

var express = require("express");  
var router = express.Router();  
const OAuth2Server = require("oauth2-server");  
const MongoDBHelper = require("../utils/MongoDBHelper");  
const collections = require("../config/collections");  
router.post("/token", async function (req, res, next) {  
 try {  
 const OAuth2Server = require("oauth2-server");  
 const oauth = new OAuth2Server({  
 model: require("./OAuthModel"),  
 grants: ["password", "refresh\_token"],  
 accessTokenLifetime: parseInt(process.env.ACCESS\_TOKEN\_LIFETIME),  
 refreshTokenLifetime: parseInt(process.env.REFRESH\_TOKEN\_LIFETIME),  
 });  
 let request = new OAuth2Server.Request(req);  
 let response = new OAuth2Server.Response(res);  
 let tokenInfo = await oauth.token(request, response);  
 console.log("Body", req.body);  
 if (req.body.grant\_type == "refresh\_token")  
 return res.status(200).send(tokenInfo);  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 const user = await UserModel.findOne({  
 osdpLoginId: req.body.username.toLowerCase(),  
 }).lean();  
 const delegateList = await UserModel.find({  
 delegateTo: { $eq: user.\_id },  
 }).select("name osdpLoginId password");  
 if (delegateList.length > 0) {  
 delegateList.push({  
 \_id: user.\_id,  
 name: user.name,  
 osdpLoginId: user.osdpLoginId,  
 password: user.password,  
 });  
 if (!req.body.chosen\_user) {  
 return res.status(202).send(delegateList);  
 } else {  
 const chosen\_user = delegateList.find(  
 (user) => user.\_id.toString() === req.body.chosen\_user  
 );  
 if (!chosen\_user) {  
 return res.status(401).send("Invalid delegation");  
 } else {  
 req.body.username = chosen\_user.osdpLoginId;  
 req.body.password = chosen\_user.password;  
 request = new OAuth2Server.Request(req);  
 tokenInfo = await oauth.token(request, response);  
 }  
 }  
 }  
 return res.status(200).send(tokenInfo);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
module.exports = router;

## File: bd-scs-backend-backend/routes/cases.js

var express = require("express");  
var router = express.Router();  
const MongoDBHelper = require("../utils/MongoDBHelper");  
const collections = require("../config/collections");  
const multer = require("multer");  
const path = require("path");  
const { ObjectId } = require("mongodb");  
const axios = require("axios");  
const { HttpsProxyAgent } = require('https-proxy-agent');  
const httpsAgent = new HttpsProxyAgent('http://10.5.241.158:8080');  
const { EminuteStatus } = require("../models/Eminute");  
const { TASKS } = require("../config/task");  
const { generateLetterTemplate } = require("../utils/letter");  
const LETTER\_TEMPLATES = require("../config/letterTemplates");  
var stream = require("stream");  
const { Readable } = require("stream");  
const { AttachemntType } = require("../models/Attachment");  
const { NotificationType } = require("../models/Notification");  
const { type } = require("os");  
const { findAddress } = require("../utils/addressUtils");  
const storage = multer.diskStorage({  
 destination: function (req, file, cb) {  
 cb(null, "uploads/");  
 },  
 filename: function (req, file, cb) {  
 cb(null, Date.now() + path.extname(file.originalname));  
 },  
});  
const upload = multer({ storage: storage });  
const filterCases = (cases, cat, nature, from, to) => {  
 return cases.filter(  
 (c) =>  
 (!cat || c.Category == cat) &&  
 (!nature || c.Nature == nature) &&  
 (!from || (c.TargetReplyDate && c.TargetReplyDate >= from)) &&  
 (!to || (c.TargetReplyDate && c.TargetReplyDate <= to))  
 );  
};  
router.get("/caseSummary", async function (req, res, next) {  
 try {  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const allCases = await CaseModel.find({}).populate(  
 "application",  
 "ApplicationType"  
 );  
 const newSchCount = filterCases(allCases, "SCH", "NEW").length;  
 const revSchCount = filterCases(allCases, "SCH", "REV").length;  
 const altSchCount = filterCases(allCases, "SCH", "ALT").length;  
 const schoolCount = newSchCount + revSchCount + altSchCount;  
 const newCccCount = filterCases(allCases, "CCC", "NEW").length;  
 const revCccCount = filterCases(allCases, "CCC", "REV").length;  
 const altCccCount = filterCases(allCases, "CCC", "ALT").length;  
 const rnlCccCount = filterCases(allCases, "CCC", "RNL").length;  
 const cccCount = newCccCount + revCccCount + altCccCount + rnlCccCount;  
 const newNlheCount = filterCases(allCases, "NLHE", "NEW").length;  
 const revNlheCount = filterCases(allCases, "NLHE", "REV").length;  
 const altNlheCount = filterCases(allCases, "NLHE", "ALT").length;  
 const nlheCount = newNlheCount + revNlheCount + altNlheCount;  
 const currentDate = new Date();  
 const sevenDaysAfter = new Date(currentDate);  
 sevenDaysAfter.setDate(currentDate.getDate() + 7);  
 const sevenDaysNewSchCount = filterCases(  
 allCases,  
 "SCH",  
 "NEW",  
 currentDate,  
 sevenDaysAfter  
 ).length;  
 const sevenDaysRevSchCount = filterCases(  
 allCases,  
 "SCH",  
 "REV",  
 currentDate,  
 sevenDaysAfter  
 ).length;  
 const sevenDaysAltSchCount = filterCases(  
 allCases,  
 "SCH",  
 "ALT",  
 currentDate,  
 sevenDaysAfter  
 ).length;  
 const sevenDaysSchCount =  
 sevenDaysNewSchCount + sevenDaysRevSchCount + sevenDaysAltSchCount;  
 const sevenDaysNewCccCount = filterCases(  
 allCases,  
 "CCC",  
 "NEW",  
 currentDate,  
 sevenDaysAfter  
 ).length;  
 const sevenDaysRevCccCount = filterCases(  
 allCases,  
 "CCC",  
 "REV",  
 currentDate,  
 sevenDaysAfter  
 ).length;  
 const sevenDaysAltCccCount = filterCases(  
 allCases,  
 "CCC",  
 "ALT",  
 currentDate,  
 sevenDaysAfter  
 ).length;  
 const sevenDaysRnlCccCount = filterCases(  
 allCases,  
 "CCC",  
 "RNL",  
 currentDate,  
 sevenDaysAfter  
 ).length;  
 const sevenDaysCccCount =  
 sevenDaysNewCccCount +  
 sevenDaysRevCccCount +  
 sevenDaysAltCccCount +  
 sevenDaysRnlCccCount;  
 const sevenDaysNewNlheCount = filterCases(  
 allCases,  
 "NLHE",  
 "NEW",  
 currentDate,  
 sevenDaysAfter  
 ).length;  
 const sevenDaysRevNlheCount = filterCases(  
 allCases,  
 "NLHE",  
 "REV",  
 currentDate,  
 sevenDaysAfter  
 ).length;  
 const sevenDaysAltNlheCount = filterCases(  
 allCases,  
 "NLHE",  
 "ALT",  
 currentDate,  
 sevenDaysAfter  
 ).length;  
 const sevenDaysNlheCount =  
 sevenDaysNewNlheCount + sevenDaysRevNlheCount + sevenDaysAltNlheCount;  
 const overdueNewSchCount = filterCases(  
 allCases,  
 "SCH",  
 "NEW",  
 null,  
 currentDate  
 ).length;  
 const overdueRevSchCount = filterCases(  
 allCases,  
 "SCH",  
 "REV",  
 null,  
 currentDate  
 ).length;  
 const overdueAltSchCount = filterCases(  
 allCases,  
 "SCH",  
 "ALT",  
 null,  
 currentDate  
 ).length;  
 const overdueSchCount =  
 overdueNewSchCount + overdueRevSchCount + overdueAltSchCount;  
 const overdueNewCccCount = filterCases(  
 allCases,  
 "CCC",  
 "NEW",  
 null,  
 currentDate  
 ).length;  
 console.log(  
 "overdue\_cases\_NEW\_CCC",  
 filterCases(allCases, "CCC", "NEW", null, currentDate).map(  
 (c) => c.TargetReplyDate  
 )  
 );  
 const overdueRevCccCount = filterCases(  
 allCases,  
 "CCC",  
 "REV",  
 null,  
 currentDate  
 ).length;  
 console.log(  
 "overdue\_cases\_REV\_CCC",  
 filterCases(allCases, "CCC", "REV", null, currentDate).map(  
 (c) => c.TargetReplyDate  
 )  
 );  
 const overdueAltCccCount = filterCases(  
 allCases,  
 "CCC",  
 "ALT",  
 null,  
 currentDate  
 ).length;  
 console.log(  
 "overdue\_cases\_ALT\_CCC",  
 filterCases(allCases, "CCC", "ALT", null, currentDate).map(  
 (c) => c.TargetReplyDate  
 )  
 );  
 const overdueRnlCccCount = filterCases(  
 allCases,  
 "CCC",  
 "RNL",  
 null,  
 currentDate  
 ).length;  
 console.log(  
 "overdue\_cases\_RNL\_CCC",  
 filterCases(allCases, "CCC", "RNL", null, currentDate).map(  
 (c) => c.TargetReplyDate  
 )  
 );  
 const overdueCccCount =  
 overdueNewCccCount +  
 overdueRevCccCount +  
 overdueAltCccCount +  
 overdueRnlCccCount;  
 const overdueNewNlheCount = filterCases(  
 allCases,  
 "NLHE",  
 "NEW",  
 null,  
 currentDate  
 ).length;  
 const overdueRevNlheCount = filterCases(  
 allCases,  
 "NLHE",  
 "REV",  
 null,  
 currentDate  
 ).length;  
 const overdueAltNlheCount = filterCases(  
 allCases,  
 "NLHE",  
 "ALT",  
 null,  
 currentDate  
 ).length;  
 const overdueNlheCount =  
 overdueNewNlheCount + overdueRevNlheCount + overdueAltNlheCount;  
 const summary = {  
 total: {  
 sch: schoolCount,  
 newsch: newSchCount,  
 revsch: revSchCount,  
 altsch: altSchCount,  
 ccc: cccCount,  
 newccc: newCccCount,  
 revccc: revCccCount,  
 altccc: altCccCount,  
 rnlccc: rnlCccCount,  
 nlhe: nlheCount,  
 newnlhe: newNlheCount,  
 revnlhe: revNlheCount,  
 altnlhe: altNlheCount,  
 },  
 sevenDaysBeforeTargetReplyDate: {  
 sch: sevenDaysSchCount,  
 newsch: sevenDaysNewSchCount,  
 revsch: sevenDaysRevSchCount,  
 altsch: sevenDaysAltSchCount,  
 ccc: sevenDaysCccCount,  
 newccc: sevenDaysNewCccCount,  
 revccc: sevenDaysRevCccCount,  
 altccc: sevenDaysAltCccCount,  
 rnlccc: sevenDaysRnlCccCount,  
 nlhe: sevenDaysNlheCount,  
 newnlhe: sevenDaysNewNlheCount,  
 revnlhe: sevenDaysRevNlheCount,  
 altnlhe: sevenDaysAltNlheCount,  
 },  
 overdue: {  
 sch: overdueSchCount,  
 newsch: overdueNewSchCount,  
 revsch: overdueRevSchCount,  
 altsch: overdueAltSchCount,  
 ccc: overdueCccCount,  
 newccc: overdueNewCccCount,  
 revccc: overdueRevCccCount,  
 altccc: overdueAltCccCount,  
 rnlccc: overdueRnlCccCount,  
 nlhe: overdueNlheCount,  
 newnlhe: overdueNewNlheCount,  
 revnlhe: overdueRevNlheCount,  
 altnlhe: overdueAltNlheCount,  
 },  
 };  
 if (allCases) {  
 } else {  
 next({ status: 400, message: "Cannot fetch case" });  
 }  
 res.send(summary);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/", async function (req, res, next) {  
 try {  
 const query = {};  
 if (req.query.LicensingCaseNo) {  
 try {  
 query.\_id = new ObjectId(req.query.LicensingCaseNo);  
 } catch (error) {  
 return res  
 .status(400)  
 .send({ error: "Invalid Licensing Case Number format." });  
 }  
 }  
 if (req.query.ApplicationNo) {  
 const ApplicationModel = MongoDBHelper.getCollection(  
 collections.Application  
 );  
 const applications = await ApplicationModel.find({  
 ApplicationNo: {  
 $regex: req.query.ApplicationNo,  
 $options: "i",  
 },  
 });  
 query.application = {  
 $in: applications.map((app) => app.\_id),  
 };  
 }  
 if (req.query.Category) {  
 query.Category = {  
 $regex: req.query.Category,  
 $options: "i",  
 };  
 }  
 if (req.query.Nature) {  
 query.Nature = {  
 $regex: req.query.Nature,  
 $options: "i",  
 };  
 }  
 if (req.query.FileReference) {  
 query.FileReference = {  
 $regex: req.query.FileReference,  
 $options: "i",  
 };  
 }  
 if (req.query.overdue) {  
 query.TargetReplyDate = {  
 $lte: new Date(),  
 };  
 }  
 if (req.query.sevenDaysBeforeTargetReplyDate) {  
 query.TargetReplyDate = {  
 $gte: new Date(),  
 $lte: new Date(new Date().setDate(new Date().getDate() + 7)),  
 };  
 }  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const cases = await CaseModel.find(query).sort({  
 createdAt: -1,  
 });  
 res.send(cases);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/:caseId", async function (req, res, next) {  
 try {  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const SysFileRefModel = MongoDBHelper.getCollection(collections.SysFileRef);  
 const caseItem = await CaseModel.findOne({ \_id: req.params.caseId })  
 .populate("application")  
 .populate("caseOfficerReceive", "name position")  
 .lean();  
 if (caseItem?.application?.FileReference) {  
 const sysFileRef = await SysFileRefModel.findOne({  
 sysFileRefId: caseItem.application.FileReference,  
 });  
 caseItem.FileReference = sysFileRef?.display;  
 }  
 console.log("CaseItem", caseItem.application);  
 if (caseItem?.application.RelatedPremise) {  
 const relatedPremiseDetails = (  
 await findAddress({  
 BuildingId: caseItem.application.RelatedPremise,  
 })  
 )?.[0];  
 caseItem.RelatedPremiseType = relatedPremiseDetails.bt\_e;  
 caseItem.District = relatedPremiseDetails.district\_e;  
 caseItem.Region = relatedPremiseDetails.region\_e;  
 caseItem.Area = relatedPremiseDetails.area\_e;  
 }  
 if (caseItem) {  
 const attachments = await AttachmentModel.find({  
 submissionCase: req.params.caseId,  
 type: AttachemntType.CASE,  
 });  
 caseItem.attachments = attachments;  
 const site\_inspection = await AttachmentModel.findOne({  
 submissionCase: req.params.caseId,  
 type: AttachemntType.INSPECTION\_REPORT,  
 });  
 const prepareLetter = await AttachmentModel.find({  
 submissionCase: req.params.caseId,  
 type: AttachemntType.PREPARE\_LETTER,  
 });  
 const issueLetter = await AttachmentModel.find({  
 submissionCase: req.params.caseId,  
 type: AttachemntType.ISSUE\_LETTER,  
 });  
 caseItem.site\_inspection = site\_inspection;  
 caseItem.prepareLetter = prepareLetter;  
 caseItem.issueLetter = issueLetter;  
 const tasks = await TaskModel.find({ submissionCase: req.params.caseId });  
 caseItem.tasks = tasks  
 .map((t) => {  
 const taskDetails = TASKS.find((task) => task.type === t.taskType);  
 return { ...t.toObject(), ...taskDetails };  
 })  
 .sort((a, b) => a.zIndex - b.zIndex);  
 res.send(caseItem);  
 } else {  
 next({ status: 400, message: "Case not found" });  
 }  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post("/:caseId", async function (req, res, next) {  
 try {  
 const SubmissionModel = MongoDBHelper.getCollection(collections.Submission);  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 console.log(req.body);  
 if (req.body.dv) {  
 const adequacyOfExitsFromStoreysFail =  
 req.body.dv.adequacyOfExitsFromStoreys.filter(  
 (row) => row.checkingResult != "yes"  
 );  
 const adequacyOfStaircasesFail =  
 req.body.dv.adequacyOfStaircasesResult.checkingResult.filter(  
 (row) => row != "yes"  
 );  
 console.log(adequacyOfExitsFromStoreysFail, adequacyOfStaircasesFail);  
 if (  
 adequacyOfExitsFromStoreysFail.length > 0 ||  
 adequacyOfStaircasesFail.length > 0  
 ) {  
 req.body.dv.acceptable = false;  
 } else {  
 req.body.dv.acceptable = true;  
 }  
 }  
 if (req.body.caseOfficerReceive) {  
 }  
 const caseItem = await CaseModel.findOneAndUpdate(  
 { \_id: req.params.caseId },  
 { ...req.body, updatedAt: new Date() },  
 { new: true }  
 );  
 console.log(caseItem);  
 res.send(caseItem);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/:caseId/letterTemplate", async function (req, res, next) {  
 try {  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const caseData = await CaseModel.findOne({  
 \_id: req.params.caseId,  
 })  
 .populate("application")  
 .lean();  
 if (!caseData) {  
 console.log("Case not found");  
 return next({ status: 404, message: "Case not found" });  
 }  
 const efolioAttachment = await AttachmentModel.findOne({  
 submissionCase: caseData.\_id,  
 type: AttachemntType.CREATE\_CASE,  
 });  
 caseData.efolio = efolioAttachment;  
 const templateBuffer = await generateLetterTemplate(  
 req.query.type,  
 caseData  
 );  
 if (!templateBuffer) {  
 return next({ status: 400, message: "Invalid letter type" });  
 }  
 res.setHeader(  
 "Content-Disposition",  
 `attachment; filename=${LETTER\_TEMPLATES[req.query.type].name}`  
 );  
 res.setHeader(  
 "Content-Type",  
 "application/vnd.openxmlformats-officedocument.wordprocessingml.document"  
 );  
 var readStream = new stream.PassThrough();  
 readStream.end(templateBuffer);  
 readStream.pipe(res);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/:caseId/eminutes", async function (req, res, next) {  
 try {  
 const EminuteModel = MongoDBHelper.getCollection(collections.Eminute);  
 const minutes = await EminuteModel.find({  
 submissionCase: req.params.caseId,  
 })  
 .populate(["from", "to"])  
 .lean();  
 console.log(minutes);  
 res.send(minutes);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/:caseId/eminutes/:eminuteId", async function (req, res, next) {  
 try {  
 const EminuteModel = MongoDBHelper.getCollection(collections.Eminute);  
 const minute = await EminuteModel.findOne({  
 \_id: req.params.eminuteId,  
 submissionCase: req.params.caseId,  
 })  
 .populate(["from", "to"])  
 .lean();  
 console.log(minute);  
 res.send(minute);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post("/:caseId/eminutes/:eminuteId", async function (req, res, next) {  
 try {  
 const EminuteModel = MongoDBHelper.getCollection(collections.Eminute);  
 const minute = await EminuteModel.findOneAndUpdate(  
 {  
 \_id: req.params.eminuteId,  
 submissionCase: req.params.caseId,  
 },  
 { ...req.body },  
 { new: true }  
 )  
 .populate(["from", "to"])  
 .lean();  
 const { status } = req.body;  
 if (status == EminuteStatus.ENDORSED) {  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 const tasks = await TaskModel.find({ submissionCase: req.params.caseId });  
 await Promise.all(  
 tasks.map(async (task) => {  
 const taskDetails = TASKS.find((t) => t.type === task.taskType);  
 const endorseAction = taskDetails.endorseAction;  
 if (  
 endorseAction &&  
 endorseAction[0] == minute.from?.split("/")?.[0] &&  
 endorseAction[1] == minute.to?.split("/")?.[0] &&  
 endorseAction[2] == task.status  
 ) {  
 await TaskModel.findOneAndUpdate(  
 { \_id: task.\_id },  
 { status: endorseAction[3] }  
 );  
 }  
 })  
 );  
 }  
 console.log(minute);  
 res.send(minute);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post("/:caseId/eminutes", async function (req, res, next) {  
 try {  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 const ApplicationModel = MongoDBHelper.getCollection(  
 collections.Application  
 );  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const EminuteModel = MongoDBHelper.getCollection(collections.Eminute);  
 const NotificationModel = MongoDBHelper.getCollection(  
 collections.Notification  
 );  
 const caseItem = await CaseModel.findOne({  
 \_id: req.params.caseId,  
 })  
 .populate("application")  
 .lean();  
 if (caseItem) {  
 const sysFileRefEminutesCount = await EminuteModel.countDocuments({  
 sysFileRefId: caseItem.application.FileReference,  
 });  
 const caseEminutesCount = await EminuteModel.countDocuments({  
 submissionCase: caseItem.\_id,  
 });  
 const eminute = await EminuteModel.create({  
 submissionCase: caseItem.\_id,  
 from: req.body.from,  
 to: req.body.to,  
 content: req.body.content,  
 subject: req.body.subject,  
 eminuteId: "E-" + String(caseEminutesCount).padStart(3, "0"),  
 sysFileRefId: caseItem.application.FileReference,  
 efolio: "" + (sysFileRefEminutesCount + 1),  
 status: EminuteStatus.PENDING,  
 });  
 const notification = await NotificationModel.create({  
 user: eminute.to,  
 notificationType: NotificationType.NEW\_EMINUTE,  
 eminute: eminute.\_id,  
 requireSendEmail: false,  
 });  
 // Check if any eminute action is matched  
 const tasks = await TaskModel.find({ submissionCase: req.params.caseId });  
 // Change task status if eminute action is matched  
 await Promise.all(  
 tasks.map(async (task) => {  
 const taskDetails = TASKS.find((t) => t.type === task.taskType);  
 console.log("Tasks", task.taskType, ":", taskDetails);  
 const eminuteAction = taskDetails.eminuteAction;  
 console.log(  
 "Eminute",  
 eminute.from?.split("/")?.[0],  
 eminute.to?.split("/")?.[0],  
 task.status,  
 eminuteAction  
 );  
 if (  
 eminuteAction &&  
 eminuteAction[0] == eminute.from?.split("/")?.[0] &&  
 eminuteAction[1] == eminute.to?.split("/")?.[0] &&  
 eminuteAction[2] == task.status  
 ) {  
 await TaskModel.findOneAndUpdate(  
 { \_id: task.\_id },  
 { status: eminuteAction[3] }  
 );  
 }  
 })  
 );  
 res.send(eminute);  
 } else {  
 next({ status: 400, message: "Case not found" });  
 }  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post(  
 "/:caseId/siteinspection",  
 upload.single("file"),  
 async function (req, res, next) {  
 try {  
 if (!req.file) {  
 return res.status(400).send("No file uploaded.");  
 }  
 const AttachmentModel = MongoDBHelper.getCollection(  
 collections.Attachment  
 );  
 const existingReport = await AttachmentModel.findOne({  
 submissionCase: req.params.caseId,  
 type: AttachemntType.INSPECTION\_REPORT,  
 });  
 if (existingReport) {  
 existingReport.file = req.file;  
 existingReport.updatedAt = new Date();  
 await existingReport.save();  
 res.send(existingReport);  
 } else {  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const caseItem = await CaseModel.findOne({  
 \_id: req.params.caseId,  
 }).populate("application");  
 const efolioCount = await AttachmentModel.countDocuments({  
 sysFileRefId: caseItem.application.FileReference,  
 });  
 const attachment = await AttachmentModel.create({  
 application: caseItem.application,  
 submissionCase: req.params.caseId,  
 type: AttachemntType.INSPECTION\_REPORT,  
 sysFileRefId: caseItem.application.FileReference,  
 efolio: "" + (efolioCount + 1),  
 file: req.file,  
 receivedDate: new Date(),  
 });  
 console.log(attachment);  
 res.send(attachment);  
 }  
 } catch (err) {  
 next(err);  
 }  
 }  
);  
/\* Add case attachment. \*/  
router.post(  
 "/:caseId/attachments",  
 upload.single("file"),  
 async function (req, res, next) {  
 try {  
 console.log("FFFFFF", req.body);  
 if (!req.file) {  
 return res.status(400).send("No file uploaded.");  
 }  
 const AttachmentModel = MongoDBHelper.getCollection(  
 collections.Attachment  
 );  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const caseItem = await CaseModel.findOne({ \_id: req.params.caseId });  
 if (caseItem) {  
 const attachment = await AttachmentModel.create({  
 ...req.body,  
 application: caseItem.application,  
 submissionCase: req.params.caseId,  
 file: req.file,  
 });  
 console.log(attachment);  
 res.send(attachment);  
 } else {  
 next({ status: 400, message: "Case not found" });  
 }  
 } catch (err) {  
 next(err);  
 }  
 }  
);  
router.post("/:caseId/issueLetter", async function (req, res, next) {  
 try {  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const caseItem = await CaseModel.findOne({ \_id: req.params.caseId });  
 if (caseItem) {  
 const attachments = await AttachmentModel.find({  
 type: "PREPARE\_LETTER",  
 submissionCase: req.params.caseId,  
 }).sort({ subType: 1 });  
 const pdfObject = await mergeDocxToPdf(attachments);  
 if (pdfObject) {  
 const pdfAttachment = await AttachmentModel.create({  
 type: "ISSUE\_LETTER",  
 subType: "ISSUE\_LETTER",  
 application: caseItem.application,  
 submissionCase: req.params.caseId,  
 file: pdfObject,  
 });  
 console.log(pdfAttachment, "Letter issued");  
 res.send("Letter issued");  
 } else {  
 next({ status: 400, message: "Cannot generate PDF" });  
 }  
 } else {  
 next({ status: 400, message: "Case not found" });  
 }  
 } catch (err) {  
 next(err);  
 }  
});  
const mergeDocxToPdf = async (attachments) => {  
 const fs = require("fs");  
 const path = require("path");  
 const { promisify } = require("util");  
 const readFile = promisify(fs.readFile);  
 const writeFile = promisify(fs.writeFile);  
 const libre = require("libreoffice-convert");  
 const { default: PDFMerger } = await import("pdf-merger-js");  
 const pdfs = [];  
 for (let i = 0; i < attachments.length; i++) {  
 const attachment = attachments[i];  
 const docxBuffer = await readFile(attachment.file.path);  
 const formData = new FormData();  
 const file = new Blob([fs.readFileSync(attachment.file.path)], {  
 type: lookup(attachment.file.path),  
 });  
 formData.append("file", file, attachment.file.filename);  
 const response = await axios.post(`${process.env.WORD\_CONTROLLER\_URL}/WordConvert/pdf`, formData, {  
 headers: {  
 'Content-Type': 'multipart/form-data'  
 },  
 responseType: 'arraybuffer',  
 });  
 const timestamp = Date.now();  
 const pdfPath = path.join("uploads", `${attachment.file.filename.replace(".docx", "")}\_${timestamp}.pdf`);  
 await fs.promises.writeFile(pdfPath, response.data);  
 pdfs.push(pdfPath);  
 }  
 const pdfMergeInstance = new PDFMerger();  
 for (const pdf of pdfs) {  
 const pdfBuffer = await readFile(pdf);  
 await pdfMergeInstance.add(pdfBuffer);  
 }  
 const mergedPdfBuffer = await pdfMergeInstance.saveAsBuffer();  
 pdfs.forEach((pdf) => {  
 fs.unlinkSync(pdf);  
 });  
 const timestamp = Date.now();  
 const pdfPath = path.join("uploads", `merged\_${timestamp}.pdf`);  
 await writeFile(pdfPath, mergedPdfBuffer);  
 return {  
 fieldname: "file",  
 originalname: `merged\_${timestamp}.pdf`,  
 encoding: "7bit",  
 mimetype: "application/pdf",  
 destination: "uploads/",  
 filename: `merged\_${timestamp}.pdf`,  
 path: pdfPath,  
 size: mergedPdfBuffer.length,  
 };  
};  
module.exports = router;

## File: bd-scs-backend-backend/routes/fileReferences.js

var express = require("express");  
var router = express.Router();  
const MongoDBHelper = require("../utils/MongoDBHelper");  
const collections = require("../config/collections");  
const multer = require("multer");  
const path = require("path");  
router.post("/", async function (req, res) {  
 try {  
 await MongoDBHelper.connectionDB();  
 const SysFileRef = MongoDBHelper.getCollection(collections.SysFileRef);  
 const { frefPref, frefSeq, frefYr, frefSuf } = req.body;  
 let query = {};  
 console.log("Body", req.body);  
 if (frefPref) query.frefPref = frefPref;  
 if (frefSeq) query.frefSeq = frefSeq;  
 if (frefYr) query.frefYr = frefYr;  
 if (frefSuf) query.frefSuf = frefSuf;  
 const sysFileRefs = await SysFileRef.find(query).limit(30);  
 res.json(sysFileRefs);  
 } catch (error) {  
 console.error("Error getting data:", error);  
 res.status(500).send("Error getting data");  
 }  
});  
router.post("/application-histories", async function (req, res) {  
 try {  
 await MongoDBHelper.connectionDB();  
 const SysFileRef = MongoDBHelper.getCollection(collections.SysFileRef);  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const { frefPref, frefSeq, frefYr, frefSuf } = req.body;  
 let query = {};  
 console.log("Body", req.body);  
 if (frefPref) query.frefPref = frefPref;  
 if (frefSeq) query.frefSeq = frefSeq;  
 if (frefYr) query.frefYr = frefYr;  
 if (frefSuf) query.frefSuf = frefSuf;  
 const sysFileRefs = await SysFileRef.find(query);  
 console.log("sysFileRefs", sysFileRefs);  
 const attachments = await AttachmentModel.find({  
 efolio: { $ne: null }, sysFileRefId: { $in: sysFileRefs.map((sysFileRef) => sysFileRef.sysFileRefId) } }).lean();  
 res.json(attachments.map((attachment) => {  
 const sysFileRef = sysFileRefs.find((sysFileRef) => sysFileRef.sysFileRefId === attachment.sysFileRefId);  
 attachment.sysFileRefDisplay = sysFileRef.display;  
 return attachment;  
 }));  
 } catch (error) {  
 console.error("Error getting data:", error);  
 res.status(500).send("Error getting data");  
 }  
});  
module.exports = router;

## File: bd-scs-backend-backend/routes/index.js

const { sync } = require("../scripts/syncFrontendSubmissions");  
var express = require('express');  
var router = express.Router();  
router.get('/', function(req, res, next) {  
 res.render('index', { title: 'Express' });  
});  
router.get('/forcesync', async function (req, res, next) {  
 try {  
 let result = await sync();  
 res.send('completed');  
 } catch (e) {  
 next(e);  
 }  
});  
module.exports = router;

## File: bd-scs-backend-backend/routes/OAuthModel.js

const collections = require("../config/collections");  
const MongoDBHelper = require("../utils/MongoDBHelper");  
module.exports = {  
 generateAccessToken: async function (client, user, scope) {  
 console.log("OAuthServer", "generateAccessToken", client, user, scope);  
 var jwt = require("jsonwebtoken");  
 return jwt.sign(user, process.env.JWT\_SECRET, {  
 expiresIn: parseInt(process.env.ACCESS\_TOKEN\_LIFETIME),  
 });  
 },  
 generateRefreshToken: async function (client, user, scope) {  
 console.log("OAuthServer", "generateRefreshToken", client, user, scope);  
 const { v4: uuidv4 } = require("uuid");  
 return uuidv4();  
 },  
 getAccessToken: async function (accessToken) {  
 console.log("OAuthServer", "getAccessToken", accessToken);  
 const OAuthToken = MongoDBHelper.getCollection(collections.OAuthToken);  
 const oauthToken = await OAuthToken.findOne({ accessToken })  
 .populate("user")  
 .lean();  
 console.log("getAccessToken", oauthToken);  
 return oauthToken;  
 },  
 revokeToken: async function (token) {  
 console.log("OAuthServer", "revokeToken", token);  
 const OAuthToken = MongoDBHelper.getCollection(collections.OAuthToken);  
 const result = await OAuthToken.deleteOne({  
 refreshToken: token.refreshToken,  
 });  
 console.log("revoke token result", result);  
 return true;  
 },  
 getRefreshToken: async function (refreshToken) {  
 console.log("OAuthServer", "getRefreshToken", refreshToken);  
 const OAuthToken = MongoDBHelper.getCollection(collections.OAuthToken);  
 const oauthToken = await OAuthToken.findOne({ refreshToken })  
 .populate("user")  
 .lean();  
 console.log("getRefreshToken", oauthToken);  
 return oauthToken;  
 },  
 getClient: async function (clientId, clientSecret) {  
 console.log("OAuthServer", "getClient", clientId, clientSecret);  
 if (  
 clientId == process.env.CLIENT\_ID &&  
 clientSecret == process.env.CLIENT\_SECRET  
 ) {  
 return {  
 id: clientId,  
 grants: ["password", "refresh\_token"],  
 };  
 }  
 },  
 getUser: async function (username, password) {  
 console.log("OAuthServer", "getUser", username, password);  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 let user = await UserModel.findOne({  
 osdpLoginId: username.toLowerCase(),  
 }).lean();  
 if (user) {  
 if (user.lock) {  
 throw new Error("Account locked");  
 }  
 var bcrypt = require("bcryptjs");  
 var validPassword = bcrypt.compareSync(password, user.password);  
 if (password == user.password || validPassword) {  
 console.log("getUser", user);  
 let update = { lastLoginAt: new Date() };  
 user = await UserModel.findOneAndUpdate({ \_id: user.\_id }, update, {  
 new: true,  
 }).lean();  
 delete user.password;  
 return user;  
 }  
 }  
 return null;  
 },  
 saveToken: async function (token, client, user) {  
 console.log("OAuthServer", "saveToken", token, client, user);  
 const OAuthToken = MongoDBHelper.getCollection(collections.OAuthToken);  
 const oauthtoken = new OAuthToken({  
 ...token,  
 client: client,  
 user: user.\_id,  
 });  
 await oauthtoken.save();  
 return {  
 accessToken: oauthtoken.accessToken,  
 accessTokenExpiresAt: oauthtoken.accessTokenExpiresAt,  
 refreshToken: oauthtoken.refreshToken,  
 refreshTokenExpiresAt: oauthtoken.refreshTokenExpiresAt,  
 client,  
 user,  
 };  
 },  
 validateScope(user, client, scope) {  
 console.log("OAuthServer", "validateScope", user, client, scope);  
 return ["read", "write"];  
 },  
 verifyScope: async function (accessToken, scope) {  
 console.log("OAuthServer", "verifyScope", accessToken, scope);  
 return true;  
 },  
};

## File: bd-scs-backend-backend/routes/submissions.js

var express = require("express");  
var router = express.Router();  
const MongoDBHelper = require("../utils/MongoDBHelper");  
const collections = require("../config/collections");  
const { SubmissionType } = require("../models/Submission");  
const { TaskType } = require("../models/Task");  
router.get("/:submissionId", async function (req, res, next) {  
 try {  
 const SubmissionModel = MongoDBHelper.getCollection(collections.Submission);  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const submission = await SubmissionModel.findOne({  
 \_id: req.params.submissionId,  
 })  
 .populate(["assignedBS", "assignedGR"])  
 .lean();  
 const attachments = await AttachmentModel.find({  
 submission: req.params.submissionId,  
 }).lean();  
 submission.attachments = attachments;  
 console.log(submission);  
 res.send(submission);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post("/:submissionId/assignGR", async function (req, res, next) {  
 try {  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 const SubmissionModel = MongoDBHelper.getCollection(collections.Submission);  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 let gr = req.body.gr;  
 const user = await UserModel.findOne({  
 \_id: gr,  
 role: "GR",  
 }).lean();  
 if (user) {  
 const submission = await SubmissionModel.findOneAndUpdate(  
 {  
 \_id: req.params.submissionId,  
 },  
 { assignedGR: user.\_id },  
 { new: true },  
 ).lean();  
 const task = await TaskModel.create({  
 application: submission.application,  
 submission: submission.\_id,  
 taskType: TaskType.GR\_HANDLE,  
 user: gr,  
 });  
 console.log(submission);  
 res.send(submission);  
 } else {  
 next({ status: 400, message: "Invalid GR" });  
 }  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post("/:submissionId/case", async function (req, res, next) {  
 try {  
 const SubmissionModel = MongoDBHelper.getCollection(collections.Submission);  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 const CaseModel = MongoDBHelper.getCollection(collections.Case);  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 const submission = await SubmissionModel.findOne({  
 \_id: req.params.submissionId,  
 }).populate("assignedBS");  
 if (submission && submission.assignedBS && submission.assignedGR) {  
 let team = submission.assignedBS.team;  
 const teamMembers = await UserModel.find({ team });  
 const teamGroup = {  
 SO: teamMembers.find((user) => user.role == "SO"),  
 TO: teamMembers.find((user) => user.role == "TO"),  
 SE: teamMembers.find((user) => user.role == "SE"),  
 BS: teamMembers.find((user) => user.role == "BS"),  
 SSE: teamMembers.find((user) => user.role == "SSE"),  
 SBS: teamMembers.find((user) => user.role == "SBS"),  
 CBS: teamMembers.find((user) => user.role == "CBS"),  
 };  
 const caseItem = await CaseModel.create({  
 submission: req.params.submissionId,  
 application: submission.application,  
 team: team,  
 });  
 submission.submissionCase = caseItem.\_id;  
 await submission.save();  
 let tasks = [  
 {  
 taskType: TaskType.DESK\_STUDY,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["SO"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.INITIAL\_SITE\_INSPECTION,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["SO"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.ENDORSE\_SO\_EMINUTES,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["BS"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.SEARCH\_STRUCTURAL\_DATA,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["TO"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.ENDORSE\_TO\_EMINUTES,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["SE"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.PREPARE\_STRUCTURAL\_ADVICE,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["SE"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.ENDORSE\_SE\_EMINUTES,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["SSE"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.CHECK\_PLAN\_AND\_SITE\_CONDITION,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["BS"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.PREPARE\_LETTER\_CERT,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["BS"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.ENDORSE\_BS\_EMINUTES,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["SBS"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.ENDORSE\_LETTER\_CERT,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["SBS"]?.\_id,  
 team,  
 },  
 {  
 taskType: TaskType.ENDORSE\_OBJECTION\_SPECIAL\_CASE,  
 application: submission.application,  
 submission: submission.\_id,  
 submissionCase: caseItem.\_id,  
 user: teamGroup["CBS"]?.\_id,  
 team,  
 },  
 ];  
 const insertedTasks = await TaskModel.insertMany(tasks);  
 console.log(caseItem);  
 res.send(caseItem);  
 } else {  
 next({ status: 400, message: "Submission not ready to create case" });  
 }  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
module.exports = router;

## File: bd-scs-backend-backend/routes/tasks.js

var express = require("express");  
var router = express.Router();  
const MongoDBHelper = require("../utils/MongoDBHelper");  
const collections = require("../config/collections");  
const requireUser = require("../middlewares/requireUser");  
const mongoose = require("mongoose");  
router.get("/", async function (req, res, next) {  
 try {  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 let filter = {};  
 if (req.query.application) {  
 filter.application = req.query.application;  
 }  
 if (req.query.case) {  
 filter.submissionCase = req.query.case;  
 }  
 const tasks = await TaskModel.find(filter).populate(["application"]).sort({  
 createdAt: -1,  
 });  
 res.send(tasks);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/me", requireUser, async function (req, res, next) {  
 try {  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 let filter = {  
 user: req.user.position,  
 };  
 if (req.user.role == "Pre-GR") {  
 filter = { $or: [{ user: "UNASSIGNED" }, { user: req.user.position }] };  
 } else {  
 filter.user = req.user.position;  
 }  
 if (req.query.application) {  
 filter.application = req.query.application;  
 }  
 if (req.query.submission) {  
 filter.submission = req.query.submission;  
 }  
 if (req.query.case) {  
 filter.submissionCase = req.query.case;  
 }  
 if (req.query.status) {  
 filter.status = req.query.status;  
 }  
 const tasks = await TaskModel.find(filter).populate(["application"]).sort({  
 createdAt: -1,  
 });  
 res.send(tasks);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/my-case-status", requireUser, async function (req, res, next) {  
 try {  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 const tasks = await TaskModel.aggregate([  
 {  
 $match: {  
 user: req.user.role == "Pre-GR" ? "UNASSIGNED" : req.user.position,  
 status: { $eq: "ACTIVE" },  
 },  
 },  
 {  
 $group: {  
 \_id: "$submissionCase",  
 oldestTask: { $first: "$$ROOT" },  
 },  
 },  
 {  
 $sort: { "oldestTask.\_id": 1 },  
 },  
 {  
 $replaceRoot: { newRoot: "$oldestTask" },  
 },  
 {  
 $lookup: {  
 from: "cases",  
 localField: "submissionCase",  
 foreignField: "\_id",  
 as: "caseDetails",  
 },  
 },  
 {  
 $unwind: "$caseDetails",  
 },  
 {  
 $lookup: {  
 from: "applications",  
 localField: "application",  
 foreignField: "\_id",  
 as: "applicationDetails",  
 },  
 },  
 {  
 $lookup: {  
 from: "sysfilerefs",  
 localField: "applicationDetails.FileReference",  
 foreignField: "sysFileRefId",  
 as: "fileReference",  
 },  
 },  
 {  
 $unwind: "$applicationDetails",  
 },  
 ]);  
 console.log("tasks", tasks);  
 return res.send(tasks);  
 } catch (err) {  
 console.log("Case err", err);  
 next(err);  
 }  
});  
router.get("/team", requireUser, async function (req, res, next) {  
 try {  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 if (!req.user.team) {  
 return res.send([]);  
 }  
 let filter = {  
 team: req.user.team,  
 };  
 if (req.query.application) {  
 filter.application = req.query.application;  
 }  
 if (req.query.submission) {  
 filter.submission = req.query.submission;  
 }  
 if (req.query.case) {  
 filter.submissionCase = req.query.case;  
 }  
 const tasks = await TaskModel.find(filter)  
 .populate(["application", "user"])  
 .sort({  
 createdAt: -1,  
 });  
 res.send(tasks);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.get("/:taskId", async function (req, res, next) {  
 try {  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 const task = await TaskModel.findOne({ \_id: req.params.taskId });  
 res.send(task);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
router.post("/:taskId", async function (req, res, next) {  
 try {  
 const TaskModel = MongoDBHelper.getCollection(collections.Task);  
 const task = await TaskModel.findOneAndUpdate(  
 { \_id: req.params.taskId },  
 { ...req.body },  
 { new: true }  
 );  
 console.log(task);  
 res.send(task);  
 } catch (e) {  
 console.log(e);  
 next(e);  
 }  
});  
module.exports = router;

## File: bd-scs-backend-backend/routes/users.js

var express = require("express");  
var router = express.Router();  
const MongoDBHelper = require("../utils/MongoDBHelper");  
const collections = require("../config/collections");  
const requireUser = require("../middlewares/requireUser");  
const bcrypt = require("bcryptjs");  
router.get("/me", requireUser, async function (req, res, next) {  
 try {  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 const user = await UserModel.findOne({ \_id: req.user.\_id }).select(  
 "-password",  
 );  
 res.send(user);  
 } catch (e) {  
 next(e);  
 }  
});  
router.get("/notification", requireUser, async function (req, res, next) {  
 try {  
 const NotificationModel = MongoDBHelper.getCollection(collections.Notification);  
 const notifications = await NotificationModel.find({ user: req.user.position });  
 res.send(notifications);  
 } catch (e) {  
 next(e);  
 }  
});  
router.get("/", async function (req, res, next) {  
 try {  
 console.log("Query", req.query);  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 const users = await UserModel.find(req.query).select("-password");  
 res.send(users);  
 } catch (e) {  
 next(e);  
 }  
});  
router.get("/positions", async function (req, res, next) {  
 try {  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 let filter = {};  
 if (req.query.role) filter.role = req.query.role;  
 if (req.query.team) filter.team = req.query.team;  
 const positions = await UserModel.find(filter).distinct("position");  
 return res.send(positions);  
 } catch (e) {  
 next(e);  
 }  
});  
router.get("/:userId", async function (req, res, next) {  
 try {  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 const user = await UserModel.findOne({ \_id: req.params.userId }).select(  
 "-password",  
 );  
 res.send(user);  
 } catch (e) {  
 next(e);  
 }  
});  
router.post("/", async function (req, res, next) {  
 try {  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 const user = await UserModel.create({  
 ...req.body,  
 password: req.body.password && bcrypt.hashSync(req.body.password, 10),  
 });  
 console.log(user);  
 res.send(user);  
 } catch (e) {  
 next(e);  
 }  
});  
router.patch("/:userId/delegate", async function (req, res, next) {  
 try {  
 console.log("I am here");  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 const user = await UserModel.findOneAndUpdate(  
 { \_id: req.params.userId },  
 { delegateTo: req.body.delegateTo },  
 { upsert: true },  
 );  
 console.log(user);  
 res.status(201).send({  
 success: true,  
 });  
 } catch (e) {  
 next(e);  
 }  
});  
router.post("/:userId", async function (req, res, next) {  
 try {  
 const UserModel = MongoDBHelper.getCollection(collections.User);  
 let updatedUser = { ...req.body };  
 if (updatedUser.password) {  
 updatedUser.password = bcrypt.hashSync(req.body.password, 10);  
 }  
 const user = await UserModel.findOneAndUpdate(  
 { \_id: req.params.userId },  
 {  
 $set: updatedUser,  
 },  
 { new: true },  
 );  
 let { password, ...result } = user;  
 return res.send(result);  
 } catch (e) {  
 next(e);  
 }  
});  
module.exports = router;

## File: bd-scs-backend-backend/utils/addressUtils.js

const { QueryTypes } = require("sequelize");  
const sequelize = require("./SQLDBHelper");  
const { isArray, transform, isObject, isDate } = require("lodash");  
function convertToCamelCase(name) {  
 if (!name || !isUpperCase(name[0])) {  
 return name;  
 }  
 const chars = name.split("");  
 fixCasing(chars);  
 return chars.join("");  
}  
function isUpperCase(char) {  
 return char === char.toUpperCase() && char !== char.toLowerCase();  
}  
function fixCasing(chars) {  
 for (let i = 0; i < chars.length; i++) {  
 if (i === 1 && !isUpperCase(chars[i])) {  
 break;  
 }  
 const hasNext = i + 1 < chars.length;  
 // Stop when the next character is already lowercase.  
 if (i > 0 && hasNext && !isUpperCase(chars[i + 1])) {  
 // If the next character is a space, lowercase the current character before exiting.  
 if (chars[i + 1] === " ") {  
 chars[i] = chars[i].toLowerCase();  
 }  
 break;  
 }  
 chars[i] = chars[i].toLowerCase();  
 }  
}  
const capitalizeKeys = (obj) => {  
 // Create a new object to store the updated key-value pairs  
 const capitalizedObj = {};  
 // Loop through each key in the original object  
 for (let key in obj) {  
 if (obj.hasOwnProperty(key)) {  
 // Capitalize the first letter of the key  
 let capitalizedKey = "";  
 if (key.startsWith("bd")) {  
 capitalizedKey =  
 key.charAt(0).toUpperCase() +  
 key.charAt(1).toUpperCase() +  
 key.slice(2);  
 } else {  
 capitalizedKey = key.charAt(0).toUpperCase() + key.slice(1);  
 }  
 if (typeof obj[key] === "object" && !Array.isArray(obj[key])) {  
 capitalizedObj[capitalizedKey] = capitalizeKeys(obj[key]);  
 } else {  
 capitalizedObj[capitalizedKey] = obj[key];  
 }  
 }  
 }  
 return capitalizedObj;  
};  
const camelize = (obj) =>  
 transform(obj, (result, value, key, target) => {  
 const camelKey = isArray(target) ? key : convertToCamelCase(key);  
 result[camelKey] =  
 isObject(value) && !isDate(value) ? camelize(value) : value;  
 });  
async function findAddress(applicationReq) {  
 try {  
 applicationReq = capitalizeKeys(applicationReq);  
 const {  
 BuildingId = "",  
 BuildingName = "",  
 StreetName = "",  
 DD = "",  
 Lot = "",  
 } = applicationReq;  
 let addresses = [];  
 if (BuildingId.trim() !== "") {  
 try {  
 const \_id = parseInt(BuildingId, 10);  
 const query = await sequelize.query(  
 `  
 SELECT  
 TOP 10  
 a.\*,  
 district.ENGLISH\_DESCRIPTION AS district\_e,  
 district.CHINESE\_DESCRIPTION AS district\_c,  
 (SELECT ENGLISH\_DESCRIPTION AS region\_e FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.SYS\_REGION\_ID) AS region\_e,  
 (SELECT CHINESE\_DESCRIPTION AS region\_c FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.SYS\_REGION\_ID) AS region\_c,  
 area.ENGLISH\_DESCRIPTION AS area\_e,  
 area.CHINESE\_DESCRIPTION AS area\_c,  
 (SELECT ENGLISH\_DESCRIPTION AS bt\_e FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.BLK\_TYPE\_ID) AS bt\_e,  
 (SELECT CHINESE\_DESCRIPTION AS bt\_c FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.BLK\_TYPE\_ID) AS bt\_c  
 FROM AdrBlk\_T a  
 LEFT JOIN Sys\_Meta\_Data\_T district  
 ON a.SYS\_DISTRICT\_ID = district.SYS\_META\_DATA\_ID  
 LEFT JOIN Sys\_Meta\_Data\_T region  
 ON a.SYS\_REGION\_ID = district.SYS\_META\_DATA\_ID  
 LEFT JOIN Sys\_Meta\_Data\_T area  
 ON a.AREA\_ID = area.SYS\_META\_DATA\_ID  
 WHERE a.ADR\_BLK\_ID = :id  
 AND a.OBSOLETE = 'N'  
 `,  
 {  
 replacements: { id: \_id },  
 type: QueryTypes.SELECT,  
 }  
 );  
 addresses = query;  
 } catch (e) {  
 console.error(e);  
 throw e;  
 }  
 } else {  
 const query = await sequelize.query(  
 `  
 SELECT TOP 10  
 a.\*,  
 district.ENGLISH\_DESCRIPTION AS district\_e,  
 district.CHINESE\_DESCRIPTION AS district\_c,  
 (SELECT ENGLISH\_DESCRIPTION AS region\_e FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.SYS\_REGION\_ID) AS region\_e,  
 (SELECT CHINESE\_DESCRIPTION AS region\_c FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.SYS\_REGION\_ID) AS region\_c,  
 area.ENGLISH\_DESCRIPTION AS area\_e,  
 area.CHINESE\_DESCRIPTION AS area\_c,  
 (SELECT ENGLISH\_DESCRIPTION AS bt\_e FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.BLK\_TYPE\_ID) AS bt\_e,  
 (SELECT CHINESE\_DESCRIPTION AS bt\_c FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.BLK\_TYPE\_ID) AS bt\_c  
 FROM AdrBlk\_T a  
 LEFT JOIN Sys\_Meta\_Data\_T district  
 ON a.SYS\_DISTRICT\_ID = district.SYS\_META\_DATA\_ID  
 LEFT JOIN Sys\_Meta\_Data\_T area  
 ON a.AREA\_ID = area.SYS\_META\_DATA\_ID  
 WHERE (  
 a.BLDG\_NAME\_E1 LIKE :buildingName  
 OR a.BLDG\_NAME\_E2 LIKE :buildingName  
 OR a.BLDG\_NAME\_E3 LIKE :buildingName  
 OR a.BLDG\_NAME\_C1 LIKE :buildingName  
 OR a.BLDG\_NAME\_C2 LIKE :buildingName  
 OR a.BLDG\_NAME\_C3 LIKE :buildingName  
 ) AND (  
 a.OSADR\_ST\_E1 LIKE :streetName  
 OR a.OSADR\_ST\_E2 LIKE :streetName  
 OR a.OSADR\_ST\_C LIKE :streetName  
 ) AND (  
 a.OSADR\_LOT\_E1 LIKE :dd  
 OR a.OSADR\_LOT\_E2 LIKE :dd  
 OR a.OSADR\_LOT\_E3 LIKE :dd  
 OR a.OSADR\_LOT\_E4 LIKE :dd  
 OR a.OSADR\_LOT\_C1 LIKE :dd  
 OR a.OSADR\_LOT\_C2 LIKE :dd  
 OR a.OSADR\_LOT\_C3 LIKE :dd  
 OR a.OSADR\_LOT\_C4 LIKE :dd  
 ) AND (  
 a.OSADR\_LOT\_E1 LIKE :lot  
 OR a.OSADR\_LOT\_E2 LIKE :lot  
 OR a.OSADR\_LOT\_E3 LIKE :lot  
 OR a.OSADR\_LOT\_E4 LIKE :lot  
 OR a.OSADR\_LOT\_C1 LIKE :lot  
 OR a.OSADR\_LOT\_C2 LIKE :lot  
 OR a.OSADR\_LOT\_C3 LIKE :lot  
 OR a.OSADR\_LOT\_C4 LIKE :lot  
 ) AND a.OBSOLETE = 'N'  
 `,  
 {  
 replacements: {  
 buildingName: `%${BuildingName}%`,  
 streetName: `%${StreetName}%`,  
 dd: `%${DD}%`,  
 lot: `%${Lot}%`,  
 },  
 type: QueryTypes.SELECT,  
 }  
 );  
 addresses = query;  
 }  
 return camelize(addresses);  
 } catch (err) {  
 console.error(err);  
 throw err;  
 }  
}  
module.exports = {  
 findAddress,  
};

## File: bd-scs-backend-backend/utils/application.js

const mongoose = require("mongoose");  
const { APPLICATION\_NO\_TYPES } = require("../models/Application");  
const MongoDBHelper = require("./MongoDBHelper");  
const collections = require("../config/collections");  
async function generateApplicationNo(applicationType) {  
 const ApplicationModel = MongoDBHelper.getCollection(collections.Application);  
 const year = new Date().getFullYear();  
 if (!Object.keys(APPLICATION\_NO\_TYPES).includes(applicationType)) {  
 return null;  
 }  
 const type = APPLICATION\_NO\_TYPES[applicationType];  
 const regexPattern = new RegExp(`^B${year}${type}`);  
 const lastApplication = await ApplicationModel.findOne({  
 ApplicationNo: { $regex: regexPattern },  
 })  
 .sort({ ApplicationNo: -1 })  
 .exec();  
 let count = 1;  
 if (lastApplication) {  
 const lastCount = parseInt(lastApplication.ApplicationNo.slice(-5));  
 count = lastCount + 1;  
 }  
 const paddedCount = count.toString().padStart(5, "0");  
 const newApplicationNo = `B${year}${type}${paddedCount}`;  
 return newApplicationNo;  
}  
module.exports = { generateApplicationNo };

## File: bd-scs-backend-backend/utils/hkpostUtils.js

const fs = require("fs");  
var P12Signer = require("@signpdf/signer-p12").P12Signer;  
const MongoDBHelper = require("../utils/MongoDBHelper");  
const collections = require("../config/collections");  
var { PDFDocument, StandardFonts, rgb } = require("pdf-lib");  
var pdflibAddPlaceholder =  
 require("@signpdf/placeholder-pdf-lib").pdflibAddPlaceholder;  
var signpdf = require("@signpdf/signpdf").default;  
const hkpostSign = async (attachment, sign) => {  
 const certFile = fs.readFileSync(sign.file.path);  
 const certPassword = sign.password;  
 console.log(certFile.length, "certFile");  
 console.log(certPassword, "certPassword");  
 var signer = new P12Signer(certFile, {  
 passphrase: certPassword,  
 });  
 const UPLOAD\_DIR = "/uploads";  
 const targetFile = attachment.file.path;  
 const AttachmentModel = MongoDBHelper.getCollection(collections.Attachment);  
 const pdfBuffer = await signWithPlaceholder(  
 targetFile,  
 targetFile,  
 {  
 signatureLength: 9932,  
 },  
 signer,  
 {  
 x: 395,  
 y: 390,  
 pageNo: 1,  
 fontSize: 12,  
 displayText: "Signed By " + "user",  
 }  
 );  
 const a = await AttachmentModel.findOneAndUpdate(  
 { \_id: attachment.\_id },  
 { "file.signed": true },  
 { new: true }  
 );  
};  
const signWithPlaceholder = async (  
 filePath,  
 targetPath,  
 placeholder = null,  
 signer = null,  
 visibleSignature = null  
) => {  
 console.log(filePath);  
 var pdfBuffer = fs.readFileSync(filePath);  
 let pdfDoc = await PDFDocument.load(pdfBuffer);  
 if (visibleSignature) {  
 const { x, y, pageNo, fontSize, displayText } = visibleSignature;  
 const pages = pdfDoc.getPages();  
 const firstPage = pages[pageNo];  
 const { width, height } = firstPage.getSize();  
 const helveticaFont = await pdfDoc.embedFont(StandardFonts.Helvetica);  
 firstPage.drawText(displayText, {  
 x: x,  
 y: y,  
 size: fontSize,  
 font: helveticaFont,  
 color: rgb(0, 0, 0),  
 });  
 }  
 if (placeholder) {  
 pdflibAddPlaceholder({  
 pdfDoc: pdfDoc,  
 reason: "The user is declaring consent through JavaScript.",  
 contactInfo: "signpdf@example.com",  
 name: "John Doe",  
 location: "Free Text Str., Free World",  
 team: "Signority",  
 byteRangePlaceholder: "\*\*\*\*\*\*\*\*\*\*",  
 ...placeholder,  
 });  
 }  
 if (placeholder || visibleSignature) {  
 pdfBuffer = await pdfDoc.save();  
 }  
 if (signer) {  
 pdfBuffer = await signpdf.sign(pdfBuffer, signer);  
 }  
 if (targetPath) {  
 fs.writeFileSync(targetPath, pdfBuffer);  
 }  
 return pdfBuffer;  
};  
module.exports = { hkpostSign };

## File: bd-scs-backend-backend/utils/letter.js

const fs = require("fs");  
const { createReport } = require("docx-templates");  
const { CAT\_DESCRIPTIONS } = require("../config/cat");  
const LETTER\_TEMPLATES = require("../config/letterTemplates");  
const { pre } = require("../models/Task");  
const { lookup } = require("mime-types");  
const axios = require("axios");  
const MongoDBHelper = require("./MongoDBHelper");  
const collections = require("../config/collections");  
const seqNumToChar = (num) => {  
 if(num > 26) {  
 return String.fromCharCode(96 + Math.floor(num / 26)) + String.fromCharCode(96 + num % 26);  
 }  
 return String.fromCharCode(96 + num);  
}  
const prepareMWS = (caseData) => {  
 let result = {  
 structsch: {  
 q4: {  
 optionscat11: {},  
 optionscat12: {},  
 optionscat13: {},  
 optionscat14: {},  
 }  
 },  
 ubw: {  
 q1: {  
 q3:{  
 cat11: {}  
 }  
 }  
 }  
 };  
 let seq = 1;  
 for (let c of (caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_1 ?? [])) {  
 if(c == "other") { continue; }  
 if(caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_1\_text  
 && caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_1\_text[c]) {  
 result.structsch.q4.optionscat11[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_1\_text[c];  
 } else {  
 result.structsch.q4.optionscat11[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["structural\_schnlhkinds\_q4\_options\_cat1\_1"][c];  
 }  
 seq++;  
 }  
 for (let c of (caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_2 ?? [])) {  
 if(c == "other") { continue; }  
 if(caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_2\_text  
 && caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_2\_text[c]) {  
 result.structsch.q4.optionscat12[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_2\_text[c];  
 } else {  
 result.structsch.q4.optionscat12[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["structural\_schnlhkinds\_q4\_options\_cat1\_2"][c];  
 }  
 seq++;  
 }  
 for (let c of (caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_3 ?? [])) {  
 if(c == "other") { continue; }  
 if(caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_3\_text  
 && caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_3\_text[c]) {  
 result.structsch.q4.optionscat13[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_3\_text[c];  
 } else {  
 result.structsch.q4.optionscat13[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["structural\_schnlhkinds\_q4\_options\_cat1\_3"][c];  
 }  
 seq++;  
 }  
 for (let c of (caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_4 ?? [])) {  
 if(c == "other") { continue; }  
 if(caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_4\_text  
 && caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_4\_text[c]) {  
 result.structsch.q4.optionscat14[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_4\_text[c];  
 } else {  
 result.structsch.q4.optionscat14[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["structural\_schnlhkinds\_q4\_options\_cat1\_4"][c];  
 }  
 seq++;  
 }  
 if (caseData?.structural\_schnlhkinds?.q4?.options\_cat1\_4?.includes("other")) {  
 result.structsch.q4.othercat14 = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q4?.other\_cat1\_4 ?? "";  
 }  
 // UBW - 1.3.1 The following MW are subjected to the control of the MWCS:  
 for (let c of caseData?.ubw?.q1?.q3?.cat1\_1 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q1?.q3?.cat1\_1\_text  
 && caseData?.ubw?.q1?.q3?.cat1\_1\_text[c]) {  
 result.ubw.q1.q3.cat11[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q1?.q3?.cat1\_1\_text[c];  
 } else {  
 result.ubw.q1.q3.cat11[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q1\_q3\_cat1\_1"][c];  
 }  
 seq++;  
 }  
 return result;  
};  
const prepareUBW = (caseData) => {  
 let result = {};  
 {  
 let q1Seq = 1;  
 for (let c of caseData?.ubw?.q1?.options ?? []) {  
 if(result.q1 == undefined || result.q1.options == undefined) { result.q1 = { options: {} }; }  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q1?.options\_text  
 && caseData?.ubw?.q1?.options\_text[c]) {  
 result.q1.options[c] = "(" + seqNumToChar(q1Seq) + ")\t" + caseData?.ubw?.q1?.options\_text[c];  
 } else {  
 result.q1.options[c] = "(" + seqNumToChar(q1Seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q1\_options"][c];  
 }  
 q1Seq++;  
 }  
 if (caseData?.ubw?.q1?.options?.includes("other")) {  
 result.q1.other = "(" + seqNumToChar(q1Seq) + ")\t" + caseData?.ubw?.q1?.other ?? "";  
 }  
 }  
 {  
 let q2Seq = 1;  
 for (let c of caseData?.ubw?.q2?.q1?.option ?? []) {  
 if(result.q2 == undefined || result.q2.q1 == undefined || result.q2.q1.options == undefined) { result.q2 = { q1: { options: {} } }; }  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q2?.q1?.option\_text  
 && caseData?.ubw?.q2?.q1?.option\_text[c]) {  
 result.q2.q1.options[c] = "(" + seqNumToChar(q2Seq) + ")\t" + caseData?.ubw?.q2?.q1?.option\_text[c];  
 } else {  
 result.q2.q1.options[c] = "(" + seqNumToChar(q2Seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q2\_q1\_option"][c];  
 }  
 q2Seq++;  
 }  
 if (caseData?.ubw?.q2?.q1?.other) {  
 result.q2.q1.other = "(" + seqNumToChar(q2Seq) + ")\t" + caseData?.ubw?.q2?.q1?.other ?? "";  
 }  
 }  
 return result;  
};  
const prepareCat1 = (caseData) => {  
 let result = {  
 deck: {  
 q5: {},  
 },  
 structsch: {  
 q5: { optionscat1: {} },  
 q6: { optionscat1: {} },  
 },  
 frc: {  
 q1: {  
 q2b: { cat1: {} },  
 q3: { cat11: {} },  
 },  
 q2: {  
 q1: { cat1: {} },  
 },  
 q6: {  
 cat1: {}  
 }  
 },  
 ubw: {  
 q1: {  
 q3: { cat1: {}, cat11: {}},  
 },  
 q4: {  
 cat1: {},  
 },  
 },  
 misc: {  
 q1: { cat1: {}},  
 },  
 };  
 let seq = 1;  
 // Desk Study Q5 - Is the address of premises correctly described ?  
 if (caseData?.deck\_study?.q5?.answer\_cat1) {  
 if(caseData?.deck\_study?.q5?.answer\_cat1\_text) {  
 result.deck.q5.answercat1 = "(" + seqNumToChar(seq) + ")\t" + caseData?.deck\_study?.q5?.answer\_cat1\_text;  
 } else {  
 result.deck.q5.answercat1 = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["desk\_study\_q5\_cat1"];  
 }  
 seq++;  
 }  
 for (let c of caseData?.structural\_schnlhkinds?.q5?.options\_cat1 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.structural\_schnlhkinds?.q5?.options\_cat1\_text  
 && caseData?.structural\_schnlhkinds?.q5?.options\_cat1\_text[c]) {  
 result.structsch.q5.optionscat1[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q5?.options\_cat1\_text[c];  
 } else {  
 result.structsch.q5.optionscat1[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["structural\_schnlhkinds\_q5\_options\_cat1"][c];  
 }  
 seq++;  
 }  
 if (caseData?.structural\_schnlhkinds?.q5?.options\_cat1?.includes("other")) {  
 result.structsch.q5.othercat1 = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q5?.other\_cat1 ?? "";  
 seq++;  
 }  
 // Structural BS - Q6 - Additional comment from SE  
 for (let c of caseData?.structural\_schnlhkinds?.q6?.options\_cat1 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.structural\_schnlhkinds?.q6?.options\_cat1\_text  
 && caseData?.structural\_schnlhkinds?.q6?.options\_cat1\_text[c]) {  
 result.structsch.q6.optionscat1[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q6?.options\_cat1\_text[c];  
 } else {  
 result.structsch.q6.optionscat1[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["structural\_schnlhkinds\_q6\_options\_cat1"][c];  
 }  
 seq++;  
 }  
 if (caseData?.structural\_schnlhkinds?.q6?.options\_cat1?.includes("other")) {  
 result.structsch.q6.othercat1 = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q6?.other\_cat1 ?? "";  
 seq++;  
 }  
 // FRC - 1.2b Is FRR certification required for the common internal corridor ?  
 for (let c of caseData?.frc?.q1?.q2b?.cat1 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.frc?.q1?.q2b?.cat1\_text  
 && caseData?.frc?.q1?.q2b?.cat1\_text[c]) {  
 result.frc.q1.q2b.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q1?.q2b?.cat1\_text[c];  
 } else {  
 result.frc.q1.q2b.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["frc\_q1\_q2b\_cat1"][c];  
 }  
 seq++;  
 }  
 if (caseData?.frc?.q1?.q2b?.cat1?.includes("other")) {  
 result.frc.q1.q2b.cat1other = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q1?.q2b?.cat1\_other ?? "";  
 seq++;  
 }  
 // FRC - 2.1 Are the propsoed school premises and sub-divided non-licence area under the same manamgenet and control of applicant ?  
 for (let c of caseData?.frc?.q2?.q1?.cat1 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.frc?.q2?.q1?.cat1\_text  
 && caseData?.frc?.q2?.q1?.cat1\_text[c]) {  
 result.frc.q2.q1.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q2?.q1?.cat1\_text[c];  
 } else {  
 result.frc.q2.q1.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["frc\_q2\_q1\_cat1"][c];  
 }  
 seq++;  
 }  
 if (caseData?.frc?.q2?.q1?.cat1?.includes("other")) {  
 result.frc.q2.q1.cat1other = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q2?.q1?.cat1\_other ?? "";  
 seq++;  
 }  
 // FRC - 6  
 for (let c of caseData?.frc?.q6?.cat1 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.frc?.q6?.cat1\_text  
 && caseData?.frc?.q6?.cat1\_text[c]) {  
 result.frc.q6.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q6?.cat1\_text[c];  
 } else {  
 result.frc.q6.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["frc\_q6\_cat1"][c];  
 }  
 seq++;  
 }  
 if (caseData?.frc?.q6?.cat1?.includes("other")) {  
 result.frc.q6.cat1other = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q6?.cat1\_other ?? "";  
 seq++;  
 }  
 // UBW - 1.3.1 The following MW are subjected to the control of the MWCS:  
 for (let c of caseData?.ubw?.q1?.q3?.cat1\_1 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q1?.q3?.cat1\_1\_text  
 && caseData?.ubw?.q1?.q3?.cat1\_1\_text[c]) {  
 result.ubw.q1.q3.cat11[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q1?.q3?.cat1\_1\_text[c];  
 } else {  
 result.ubw.q1.q3.cat11[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q1\_q3\_cat1\_1"][c];  
 }  
 seq++;  
 }  
 if(caseData?.ubw?.q1?.q3?.cat1\_1?.includes("other")) {  
 result.ubw.q1.q3.cat11other = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q1?.q3?.cat1\_1\_other ?? "";  
 seq++;  
 }  
 // UBW - 1.3  
 for (let c of caseData?.ubw?.q1?.q3?.cat1 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q1?.q3?.cat1\_text  
 && caseData?.ubw?.q1?.q3?.cat1\_text[c]) {  
 result.ubw.q1.q3.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q1?.q3?.cat1\_text[c];  
 } else {  
 result.ubw.q1.q3.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q1\_q3\_cat1"][c];  
 }  
 seq++;  
 }  
 if (caseData?.ubw?.q1?.q3?.cat1?.includes("other")) {  
 result.ubw.q1.q3.cat1other = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q1?.q3?.cat1\_other ?? "";  
 seq++;  
 }  
 // UBW - 4 Any Dedicated area / Public Passage to be affected ?  
 for (let c of caseData?.ubw?.q4?.cat1 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q4?.cat1\_text  
 && caseData?.ubw?.q4?.cat1\_text[c]) {  
 result.ubw.q4.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q4?.cat1\_text[c];  
 } else {  
 result.ubw.q4.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q4\_cat1"][c];  
 }  
 seq++;  
 }  
 if (caseData?.ubw?.q4?.cat1?.includes("other")) {  
 result.ubw.q4.cat1other = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q4?.cat1\_other ?? "";  
 seq++;  
 }  
 // MISC - 1. 4 sets of duly signed revised plans to incorporate the following matters should be submitted:  
 for (let c of caseData?.misc?.q1?.cat1 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.misc?.q1?.cat1\_text  
 && caseData?.misc?.q1?.cat1\_text[c]) {  
 result.misc.q1.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.misc?.q1?.cat1\_text[c];  
 } else {  
 result.misc.q1.cat1[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["misc\_q1\_cat1"][c];  
 }  
 seq++;  
 }  
 if (caseData?.misc?.q1?.cat1\_other) {  
 result.misc.q1.cat1other = "(" + seqNumToChar(seq) + ")\t" + caseData?.misc?.q1?.cat1\_other ?? "";  
 seq++;  
 }  
 if (caseData?.misc?.q1?.cat1\_1\_other) {  
 result.misc.q1.cat1\_1\_other = "(" + seqNumToChar(seq) + ")\t" + caseData?.misc?.q1?.cat1\_1\_other ?? "";  
 seq++;  
 }  
 return result;  
};  
const prepareCat2\_1b = (caseData) => {  
 let result = {  
 ubw: {  
 q2: {  
 q1: {  
 option: {}  
 },  
 },  
 },  
 misc: {  
 q1: {},  
 },  
 };  
 let seq = 1;  
 for (let c of caseData?.ubw?.q2?.q1?.option ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q2?.q1?.option\_text  
 && caseData?.ubw?.q2?.q1?.option\_text[c]) {  
 result.ubw.q2.q1.option[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q2?.q1?.option\_text[c];  
 } else {  
 result.ubw.q2.q1.option[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q2\_q1\_option"][c];  
 }  
 seq++;  
 }  
 if (caseData?.ubw?.q2?.q1?.other) {  
 result.ubw.q2.q1.other = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q2?.q1?.other;  
 seq++;  
 }  
 if (caseData?.misc?.q1?.cat2\_1b\_other) {  
 result.misc.q1.cat21bother = "(" + seqNumToChar(seq) + ")\t" + caseData?.misc?.q1?.cat2\_1b\_other;  
 seq++;  
 }  
 return result;  
};  
const prepareCat2\_3a = (caseData) => {  
 let result = {  
 deck: {  
 q4: {  
 optionscat23a: {},  
 },  
 },  
 misc: {  
 q1: {},  
 },  
 };  
 let seq = 1;  
 for (let c of caseData?.deck\_study?.q4?.options\_cat2\_3a ?? []) {  
 if(caseData?.deck\_study?.q4?.options\_cat2\_3a\_text  
 && caseData?.deck\_study?.q4?.options\_cat2\_3a\_text[c]) {  
 result.deck.q4.optionscat23a[c] = caseData?.deck\_study?.q4?.options\_cat2\_3a\_text[c];  
 } else {  
 result.deck.q4.optionscat23a[c] = CAT\_DESCRIPTIONS["deck\_study\_q4\_options\_cat2\_3a"][c];  
 }  
 seq++;  
 }  
 return result;  
};  
const prepareCat2\_1a = (caseData) => {  
 let result = {  
 ubw: {  
 q2: {  
 q2: {  
 cat21a: {},  
 },  
 },  
 },  
 misc: {  
 q1: {  
 cat21a: {},  
 },  
 },  
 };  
 let seq = 1;  
 for (let c of caseData?.ubw?.q2?.q2?.cat2\_1a ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q2?.q2?.cat2\_1a\_text  
 && caseData?.ubw?.q2?.q2?.cat2\_1a\_text[c]) {  
 result.ubw.q2.q2.cat21a[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q2?.q2?.cat2\_1a\_text[c];  
 } else {  
 result.ubw.q2.q2.cat21a[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q2\_q2\_cat2\_1a"][c];  
 }  
 seq++;  
 }  
 if (caseData?.ubw?.q2?.q2?.other\_cat2\_1a) {  
 result.ubw.q2.q2.cat21aother = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q2?.q2?.other\_cat2\_1a;  
 seq++;  
 }  
 if (caseData?.misc?.q1?.cat2\_1a\_other) {  
 result.misc.q1.cat21a = "(" + seqNumToChar(seq) + ")\t" + caseData?.misc?.q1?.cat2\_1a\_other;  
 seq++;  
 }  
 return result;  
};  
const prepareCat2\_3b = (caseData) => {  
 let result = {  
 frc: {  
 q2: {  
 q2: {  
 cat23b: {},  
 },  
 },  
 q3: {  
 cat23b: {},  
 },  
 q4: {  
 cat23b: {},  
 },  
 },  
 misc: {  
 q1: {},  
 },  
 };  
 let seq = 1;  
 for (let c of caseData?.frc?.q2?.q2?.cat2\_3b ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.frc?.q2?.q2?.cat2\_3b\_text  
 && caseData?.frc?.q2?.q2?.cat2\_3b\_text[c]) {  
 result.frc.q2.q2.cat23b[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q2?.q2?.cat2\_3b\_text[c];  
 } else {  
 result.frc.q2.q2.cat23b[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["frc\_q2\_q2\_cat2\_3b"][c];  
 }  
 seq++;  
 }  
 if (caseData?.frc?.q2?.q2?.cat2\_3b\_other) {  
 result.frc.q2.q2.cat23bother = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q2?.q2?.cat2\_3b\_other;  
 seq++;  
 }  
 for (let c of caseData?.frc?.q3?.cat2\_3b ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.frc?.q3?.cat2\_3b\_text  
 && caseData?.frc?.q3?.cat2\_3b\_text[c]) {  
 result.frc.q3.cat23b[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q3?.cat2\_3b\_text[c];  
 } else {  
 result.frc.q3.cat23b[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["frc\_q3\_cat2\_3b"][c];  
 }  
 seq++;  
 }  
 if (caseData?.frc?.q3?.cat2\_3b\_other) {  
 result.frc.q3.cat23bother = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q3?.cat2\_3b\_other;  
 seq++;  
 }  
 for (let c of caseData?.frc?.q4?.cat2\_3b ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.frc?.q4?.cat2\_3b\_text  
 && caseData?.frc?.q4?.cat2\_3b\_text[c]) {  
 result.frc.q4.cat23b[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q4?.cat2\_3b\_text[c];  
 } else {  
 result.frc.q4.cat23b[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["frc\_q4\_cat2\_3b"][c];  
 }  
 seq++;  
 }  
 if (caseData?.frc?.q4?.cat2\_3b\_other) {  
 result.frc.q4.cat23bother = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q4?.cat2\_3b\_other;  
 seq++;  
 }  
 if (caseData?.misc?.q1?.cat2\_3b\_other) {  
 result.misc.q1.cat23bother = "(" + seqNumToChar(seq) + ")\t" + caseData?.misc?.q1?.cat2\_3b\_other;  
 seq++;  
 }  
 return result;  
};  
const prepareCat2\_4 = (caseData) => {  
 let result = {  
 moe: {  
 q1: {  
 q9: {  
 cat24: {},  
 }  
 },  
 q2: {  
 q1: {  
 cat24: {},  
 },  
 q3: {  
 cat24: {},  
 }  
 }  
 },  
 ubw: {  
 q1: {  
 q3: {  
 cat24: {},  
 },  
 },  
 q3: {  
 cat24: {},  
 },  
 },  
 misc: {  
 q1: {},  
 }  
 };  
 let seq = 1;  
 for (let c of caseData?.moe?.q1?.q9?.cat2\_4 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.moe?.q1?.q9?.cat2\_4\_text  
 && caseData?.moe?.q1?.q9?.cat2\_4\_text[c]) {  
 result.moe.q1.q9.cat24[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.moe?.q1?.q9?.cat2\_4\_text[c];  
 } else {  
 result.moe.q1.q9.cat24[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["moe\_q1\_q9\_cat2\_4"][c];  
 }  
 seq++;  
 }  
 if (caseData?.moe?.q1?.q9?.cat2\_4\_other) {  
 result.moe.q1.q9.cat24other = "(" + seqNumToChar(seq) + ")\t" + caseData?.moe?.q1?.q9?.cat2\_4\_other;  
 seq++;  
 }  
 for (let c of caseData?.moe?.q2?.q1?.cat2\_4 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.moe?.q2?.q1?.cat2\_4\_text  
 && caseData?.moe?.q2?.q1?.cat2\_4\_text[c]) {  
 result.moe.q2.q1.cat24[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.moe?.q2?.q1?.cat2\_4\_text[c];  
 } else {  
 result.moe.q2.q1.cat24[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["moe\_q2\_q1\_cat2\_4"][c];  
 }  
 seq++;  
 }  
 if (caseData?.moe?.q2?.q1?.cat2\_4\_other) {  
 result.moe.q2.q1.cat24other = "(" + seqNumToChar(seq) + ")\t" + caseData?.moe?.q2?.q1?.cat2\_4\_other;  
 seq++;  
 }  
 for (let c of caseData?.moe?.q2?.q3?.cat2\_4 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.moe?.q2?.q3?.cat2\_4\_text  
 && caseData?.moe?.q2?.q3?.cat2\_4\_text[c]) {  
 result.moe.q2.q3.cat24[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.moe?.q2?.q3?.cat2\_4\_text[c];  
 } else {  
 result.moe.q2.q3.cat24[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["moe\_q2\_q3\_cat2\_4"][c];  
 }  
 seq++;  
 }  
 if (caseData?.moe?.q2?.q3?.cat2\_4\_other) {  
 result.moe.q2.q3.cat24other = "(" + seqNumToChar(seq) + ")\t" + caseData?.moe?.q2?.q3?.cat2\_4\_other;  
 seq++;  
 }  
 for (let c of caseData?.ubw?.q1?.q3?.cat2\_4 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q1?.q3?.cat2\_4\_text  
 && caseData?.ubw?.q1?.q3?.cat2\_4\_text[c]) {  
 result.ubw.q1.q3.cat24[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q1?.q3?.cat2\_4\_text[c];  
 } else {  
 result.ubw.q1.q3.cat24[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q1\_q3\_cat2\_4"][c];  
 }  
 seq++;  
 }  
 if (caseData?.ubw?.q1?.q3?.cat2\_4\_other) {  
 result.ubw.q1.q3.cat24other = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q1?.q3?.cat2\_4\_other;  
 seq++;  
 }  
 for (let c of caseData?.ubw?.q3?.cat2\_4 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q3?.cat2\_4\_text  
 && caseData?.ubw?.q3?.cat2\_4\_text[c]) {  
 result.ubw.q3.cat24[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q3?.cat2\_4\_text[c];  
 } else {  
 result.ubw.q3.cat24[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q3\_cat2\_4"][c];  
 }  
 seq++;  
 }  
 if (caseData?.ubw?.q3?.cat2\_4\_other) {  
 result.ubw.q3.cat24other = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q3?.cat2\_4\_other;  
 seq++;  
 }  
 if (caseData?.misc?.q1?.cat2\_4\_other) {  
 result.misc.q1.cat24other = "(" + seqNumToChar(seq) + ")\t" + caseData?.misc?.q1?.cat2\_4\_other;  
 seq++;  
 }  
 return result;  
};  
const prepareCat2\_2 = (caseData) => {  
 let result = {  
 structsch: {  
 q3: {  
 optionscat22a: {},  
 },  
 },  
 misc: {  
 q1: {},  
 },  
 };  
 let seq = 1;  
 for (let c of caseData?.structural\_schnlhkinds?.q3?.options\_cat2\_2a ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.structural\_schnlhkinds?.q3?.options\_cat2\_2a\_text  
 && caseData?.structural\_schnlhkinds?.q3?.options\_cat2\_2a\_text[c]) {  
 result.structsch.q3.optionscat22a[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q3?.options\_cat2\_2a\_text[c];  
 } else {  
 result.structsch.q3.optionscat22a[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["structural\_schnlhkinds\_q3\_cat2\_2a"][c];  
 }  
 seq++;  
 }  
 if (caseData?.structural\_schnlhkinds?.q3?.other\_cat2\_2a) {  
 result.structsch.q3.othercat22a = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q3?.other\_cat2\_2a;  
 seq++;  
 }  
 if (caseData?.misc?.q1?.cat2\_2a\_other) {  
 result.misc.q1.cat22aother = "(" + seqNumToChar(seq) + ")\t" + caseData?.misc?.q1?.cat2\_2a\_other;  
 seq++;  
 }  
 return result;  
};  
const prepareCat3 = (caseData) => {  
 let result = {  
 structsch: {  
 q1: {  
 optionscat3: {},  
 },  
 },  
 frc: {  
 q1: {  
 q2a: {  
 cat3: {},  
 },  
 },  
 },  
 moe: {  
 q1: {  
 q9: {  
 cat3: {},  
 },  
 },  
 },  
 ubw: {  
 q4: {  
 cat3: {},  
 },  
 },  
 misc: {  
 q1: {},  
 },  
 };  
 let seq = 1;  
 for (let c of caseData?.structural\_schnlhkinds?.q1?.options\_cat3 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.structural\_schnlhkinds?.q1?.options\_cat3\_text  
 && caseData?.structural\_schnlhkinds?.q1?.options\_cat3\_text[c]) {  
 result.structsch.q1.optionscat3[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q1?.options\_cat3\_text[c];  
 } else {  
 result.structsch.q1.optionscat3[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["structural\_schnlhkinds\_q1\_cat3"][c];  
 }  
 seq++;  
 }  
 for (let c of caseData?.structural\_schnlhkinds?.q1?.options\_cat3 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.structural\_schnlhkinds?.q1?.options\_cat3\_text  
 && caseData?.structural\_schnlhkinds?.q1?.options\_cat3\_text[c]) {  
 result.structsch.q1.optionscat3[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q1?.options\_cat3\_text[c];  
 } else {  
 result.structsch.q1.optionscat3[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["structural\_schnlhkinds\_q1\_cat3"][c];  
 }  
 seq++;  
 }  
 if (caseData?.structural\_schnlhkinds?.q1?.other\_cat3) {  
 result.structsch.q1.othercat3 = "(" + seqNumToChar(seq) + ")\t" + caseData?.structural\_schnlhkinds?.q1?.other\_cat3;  
 seq++;  
 }  
 for (let c of caseData?.frc?.q1?.q2a?.cat3 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.frc?.q1?.q2a?.cat3\_text  
 && caseData?.frc?.q1?.q2a?.cat3\_text[c]) {  
 result.frc.q1.q2a.cat3[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q1?.q2a?.cat3\_text[c];  
 } else {  
 result.frc.q1.q2a.cat3[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["frc\_q1\_q2a\_cat3"][c];  
 }  
 seq++;  
 }  
 if (caseData?.frc?.q1?.q2a?.cat3\_other) {  
 result.frc.q1.q2a.cat3other = "(" + seqNumToChar(seq) + ")\t" + caseData?.frc?.q1?.q2a?.cat3\_other;  
 seq++;  
 }  
 for (let c of caseData?.moe?.q1?.q9?.cat3 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.moe?.q1?.q9?.cat3\_text  
 && caseData?.moe?.q1?.q9?.cat3\_text[c]) {  
 result.moe.q1.q9.cat3[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.moe?.q1?.q9?.cat3\_text[c];  
 } else {  
 result.moe.q1.q9.cat3[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["moe\_q1\_q9\_cat3"][c];  
 }  
 seq++;  
 }  
 if (caseData?.moe?.q1?.q9?.cat3\_other) {  
 result.moe.q1.q9.cat3other = "(" + seqNumToChar(seq) + ")\t" + caseData?.moe?.q1?.q9?.cat3\_other;  
 seq++;  
 }  
 for (let c of caseData?.ubw?.q4?.cat3 ?? []) {  
 if(c == "other") { continue; }  
 if(caseData?.ubw?.q4?.cat3\_text  
 && caseData?.ubw?.q4?.cat3\_text[c]) {  
 result.ubw.q4.cat3[c] = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q4?.cat3\_text[c];  
 } else {  
 result.ubw.q4.cat3[c] = "(" + seqNumToChar(seq) + ")\t" + CAT\_DESCRIPTIONS["ubw\_q4\_cat3"][c];  
 }  
 seq++;  
 }  
 if (caseData?.ubw?.q4?.cat3\_other) {  
 result.ubw.q4.cat3other = "(" + seqNumToChar(seq) + ")\t" + caseData?.ubw?.q4?.cat3\_other;  
 seq++;  
 }  
 if (caseData?.misc?.q1?.cat3\_other) {  
 result.misc.q1.cat3other = "(" + seqNumToChar(seq) + ")\t" + caseData?.misc?.q1?.cat3\_other;  
 seq++;  
 }  
 return result;  
};  
const prepareFields = (caseData, fileReference) => {  
 let result = {  
 foliono: caseData?.efolio?.efolio,  
 yourref: "",  
 ourref: fileReference,  
 fileref: fileReference,  
 schoolname: caseData?.application?.NameOfSchoolEN,  
 schooladdress: caseData?.application?.AddressOfPremiseEN,  
 schooladdressfloor: caseData?.application?.AddressOfPremiseENFloor ? caseData?.application?.AddressOfPremiseENFloor + "/F" : "",  
 schooladdressunit: caseData?.application?.AddressOfPremiseENUnit ?? "",  
 applicantname: caseData?.application?.ApplicantNameEN,  
 applicantaddress: caseData?.application?.ApplicantAddress,  
 letterdate: new Date().toLocaleDateString("en-GB"),  
 prevletterdate: new Date().toLocaleDateString("en-GB"),  
 appno: caseData?.application?.ApplicationNo,  
 applicationdate: new Date().toLocaleDateString("en-GB"),  
 efoliono: caseData?.efolio?.efolio,  
 ubw: prepareUBW(caseData),  
 surname: "Appleseed",  
 bsbdgis: "LIC12",  
 bstitle: "Mr",  
 bslettername: "Tom Lee",  
 bsemail: "tomlee@bd.gov.hk",  
 bsphone: "1234",  
 bsbdgis1: "LIC12",  
 bsbdgis2: "LIC12",  
 date: new Date().toLocaleDateString("en-GB"),  
 selfcertapplydate: new Date().toLocaleDateString("en-GB"),  
 crossoutp3: true,  
 crossoutprevletterparagraph: true,  
 crossoutisissuecert: false,  
 crossoutisnotissuecert: true,  
 crossout8alp3: true,  
 crossout8alp5: true,  
 crossout8alp3remark: true,  
 endorsedate: new Date().toLocaleDateString("en-GB"),  
 mws: prepareMWS(caseData),  
 ubws: prepareUBW(caseData),  
 cat1: prepareCat1(caseData),  
 c21a: prepareCat2\_1a(caseData),  
 c21b: prepareCat2\_1b(caseData),  
 c22a: prepareCat2\_2(caseData),  
 c23a: prepareCat2\_3a(caseData),  
 c23b: prepareCat2\_3b(caseData),  
 c24: prepareCat2\_4(caseData),  
 c3: prepareCat3(caseData),  
 };  
 if(result?.ubw?.q2?.q1?.options && Object.keys(result.ubw.q2.q1.options).length > 0) {  
 result.crossoutp3 = false;  
 }  
 if(result.cat1.deck.q5.answercat1 ||  
 result.cat1.structsch.q5.optionscat1 ||  
 result.cat1.structsch.q5.othercat1 ||  
 result.cat1.structsch.q6.optionscat1 ||  
 result.cat1.structsch.q6.othercat1) {  
 result.cat1structuralsafety = 'Structual Safety';  
 }  
 if(result.cat1.frc.q1.q2b.cat1 ||  
 result.cat1.frc.q2.q1.cat1 ||  
 result.cat1.ubw.q1.q3.cat11 ||  
 result.cat1.ubw.q1.q3.cat11other) {  
 result.cat1frc = 'Fire Resisting Construction';  
 }  
 if(result.cat1.ubw.q1.q3.cat1 ||  
 result.cat1.ubw.q1.q3.cat1other ||  
 result.cat1.ubw.q4.cat1 ||  
 result.cat1.ubw.q4.cat1other) {  
 result.cat1ubw = 'Unauthorized Building Works';  
 }  
 if(result.cat1.misc.q1.cat1 ||  
 result.cat1.misc.q1.cat1other ||  
 result.cat1.misc.q1.cat11other) {  
 result.cat1misc = 'Miscellaneous';  
 }  
 if(result.c24.moe.q1.q9.cat24 || result.c24.moe.q2.q1.cat24 || result.c24.moe.q2.q3.cat24) {  
 result.cat24moe = 'Means of Escape';  
 }  
 if(result.c24.ubw.q1.q3.cat24 || result.c24.ubw.q3.cat24) {  
 result.cat24ubw = 'Unauthorized Building Works';  
 }  
 if(result.c3.structsch.q1.optionscat3 || result.c3.structsch.q1.othercat3) {  
 result.cat3structuralsafety = 'Structural Safety';  
 }  
 if(result.c3.frc.q1.q2a.cat3 || result.c3.frc.q1.q2a.cat3other) {  
 result.cat3frc = 'Fire Resisting Construction';  
 }  
 if(result.c3.moe.q1.q9.cat3 || result.c3.moe.q1.q9.cat3other) {  
 result.cat3moe = 'Means of Escape';  
 }  
 if(result.c3.ubw.q4.cat3 || result.c3.ubw.q4.cat3other) {  
 result.cat3ubw = 'Unauthorized Building Works';  
 }  
 return result;  
};  
function expendFields(fields) {  
 let result = {};  
 for (const key in fields) {  
 if (fields[key] instanceof Object) {  
 let subFields = expendFields(fields[key]);  
 for (const subKey in subFields) {  
 result[`${key}.${subKey}`] = subFields[subKey];  
 }  
 } else {  
 result[key] = fields[key];  
 }  
 }  
 return result;  
}  
async function generateLetterTemplate(type, caseData) {  
 if (!Object.keys(LETTER\_TEMPLATES).includes(type)) {  
 return null;  
 }  
 const SysFileRefModel = MongoDBHelper.getCollection(collections.SysFileRef);  
 const sysFileRef = await SysFileRefModel.findOne({  
 sysFileRefId: caseData.application.FileReference,  
 });  
 const template = fs.readFileSync(LETTER\_TEMPLATES[type].path);  
 const fields = prepareFields(caseData, sysFileRef?.display);  
 console.log("Fields", fields);  
 const expandedFields = expendFields(fields);  
 console.log("Expanded Fields", expandedFields);  
 const formData = new FormData();  
 const file = new Blob([template], {  
 type: lookup(LETTER\_TEMPLATES[type].path),  
 });  
 formData.append("file", file, LETTER\_TEMPLATES[type].name);  
 for(const key in expandedFields) {  
 formData.append(key, expandedFields[key]);  
 }  
 const response = await axios.post(`${process.env.WORD\_CONTROLLER\_URL}/WordMerge/merge`, formData, {  
 headers: {  
 'Content-Type': 'multipart/form-data'  
 },  
 responseType: 'arraybuffer',  
 });  
 return new Uint8Array(response.data);  
}  
async function generateAppendixX(caseData) {  
 const template = fs.readFileSync(LETTER\_TEMPLATES["appendix\_x"].path);  
 const buffer = await createReport({  
 template,  
 data: {  
 mws: [],  
 cat1: [],  
 cat2\_1a: [],  
 cat2\_1b: [],  
 cat2\_2: [],  
 cat2\_3a: [],  
 cat2\_3b: [],  
 cat2\_4: [],  
 cat3: [],  
 school\_name: "",  
 school\_address: "",  
 applicant\_name: "",  
 letter\_date: "",  
 },  
 });  
 return buffer;  
}  
async function generateAppendixII(caseData) {  
 // Get the appendix IIA template from the database  
 const template = fs.readFileSync(LETTER\_TEMPLATES["appendix\_ii"].path);  
 const buffer = await createReport({  
 template,  
 data: {},  
 });  
 console.log("Successfully generated Appendix II");  
 return buffer;  
}  
async function generateAppendixXI(caseData) {  
 const template = fs.readFileSync(LETTER\_TEMPLATES["appendix\_xi"].path);  
 const buffer = await createReport({  
 template,  
 data: {},  
 });  
 return buffer;  
}  
async function generate8blNoObjection(caseData) {  
 const template = fs.readFileSync(LETTER\_TEMPLATES["8bl\_no\_objection"].path);  
 const buffer = await createReport({  
 template,  
 data: {  
 app\_no: caseData?.application?.ApplicationNo,  
 surname: "Appleseed",  
 letter\_date: new Date().toLocaleDateString(),  
 bs\_bdgis: "LIC12",  
 bs\_letter\_name: "Tom Lee",  
 bs\_email: "tlee122@bd.gov.com",  
 },  
 });  
 return buffer;  
}  
module.exports = {  
 generateLetterTemplate,  
 generate8blNoObjection,  
 generateAppendixX,  
 generateAppendixII,  
 generateAppendixXI,  
};

## File: bd-scs-backend-backend/utils/MongoDBHelper.js

const mongoose = require("mongoose");  
const path = require("path");  
const collections = require("../config/collections");  
let conn = null;  
require("dotenv").config();  
const connectionDB = async (registerExtraModels) => {  
 if (conn == null || conn.readyState != 1) {  
 conn = mongoose.createConnection(process.env.MONGODB\_URI, {  
 appname: "ELarning-Serverless",  
 serverSelectionTimeoutMS: 5000,  
 });  
 await conn.asPromise();  
 for (const [key, value] of Object.entries(collections)) {  
 registerModel(value, require(path.join(\_\_dirname, `../models/${value}`)));  
 }  
 if (registerExtraModels) registerExtraModels(registerModel);  
 } else {  
 }  
};  
let registerModel = (name, schema) => {  
 return conn.model(name, schema);  
};  
let getCollection = (name) => {  
 return conn.model(name);  
};  
let getConnection = () => {  
 return conn;  
};  
module.exports = {  
 connectionDB,  
 getCollection,  
 getConnection,  
 registerModel,  
 ObjectId: mongoose.Types.ObjectId,  
 collections,  
 Error: mongoose.Error,  
};

## File: bd-scs-backend-backend/utils/sendEmail.js

const { SESClient, SendEmailCommand } = require("@aws-sdk/client-ses");  
const nodemailer = require("nodemailer");  
const createSendEmailCommand = (userEmail, subject, html) => {  
 return new SendEmailCommand({  
 Destination: {  
 ToAddresses: Array.isArray(userEmail) ? userEmail : [userEmail],  
 },  
 Message: {  
 Body: {  
 Html: {  
 Data: html,  
 },  
 Text: {  
 Data: html,  
 },  
 },  
 Subject: {  
 Data: subject,  
 },  
 },  
 Source: "info@ddroom.io",  
 });  
};  
const createSendEmailNodeMailer = (userEmail, subject, html) => {  
 console.log("SMTP\_HOST", process.env.SMTP\_HOST, typeof process.env.SMTP\_HOST);  
 console.log("SMTP\_PORT", process.env.SMTP\_PORT, typeof process.env.SMTP\_PORT);  
 console.log("SMTP\_TLS", process.env.SMTP\_TLS, typeof process.env.SMTP\_TLS);  
 console.log("SMTP\_USER", process.env.SMTP\_USER, typeof process.env.SMTP\_USER);  
 console.log("SMTP\_PASS", process.env.SMTP\_PASS, typeof process.env.SMTP\_PASS);  
 return new Promise((resolve, reject) => {  
 const transporter = nodemailer.createTransport({  
 host: process.env.SMTP\_HOST,  
 port: process.env.SMTP\_PORT,  
 secure: process.env.SMTP\_TLS,  
 auth: {  
 user: process.env.SMTP\_USER,  
 pass: process.env.SMTP\_PASS,  
 },  
 });  
 transporter.sendMail(  
 {  
 from: process.env.BD\_EMAIL,  
 to: userEmail,  
 subject: subject,  
 html: html,  
 },  
 (err, info) => {  
 if (err) {  
 reject(err);  
 } else {  
 resolve(info);  
 }  
 }  
 );  
 });  
};  
module.exports = async function (userEmail, subject, html) {  
 try {  
 if (process.env.USE\_NODEMAILER === "true")  
 return await createSendEmailNodeMailer(userEmail, subject, html);  
 else {  
 const sendEmailCommand = createSendEmailCommand(userEmail, subject, html);  
 const result = await sesClient.send(sendEmailCommand);  
 console.log("sendEmail result", result);  
 console.log("sendEmail done");  
 return result;  
 }  
 } catch (caught) {  
 if (caught instanceof Error && caught.name === "MessageRejected") {  
 const messageRejectedError = caught;  
 return messageRejectedError;  
 }  
 throw caught;  
 }  
};

## File: bd-scs-backend-backend/utils/SQLDBHelper.js

const { Sequelize } = require("sequelize");  
require("dotenv").config();  
console.log("DB\_NAME", process.env.SQL\_DB\_NAME);  
console.log("DB\_USER", process.env.SQL\_DB\_USER);  
console.log("DB\_PASSWORD", process.env.SQL\_DB\_PASSWORD);  
console.log("DB\_HOST", process.env.SQL\_DB\_HOST);  
const sequelize = new Sequelize(  
 process.env.SQL\_DB\_NAME,  
 process.env.SQL\_DB\_USER,  
 process.env.SQL\_DB\_PASSWORD,  
 {  
 host: process.env.SQL\_DB\_HOST,  
 dialect: process.env.SQL\_DB\_DIALECT,  
 }  
);  
module.exports = sequelize;

## File: bd-scs-backend-backend/app.js

require('dotenv').config();  
var createError = require("http-errors");  
var express = require("express");  
var path = require("path");  
var cookieParser = require("cookie-parser");  
var logger = require("morgan");  
var cors = require("cors");  
var indexRouter = require("./routes/index");  
var authRouter = require("./routes/auth");  
var usersRouter = require("./routes/users");  
var tasksRouter = require("./routes/tasks");  
var casesRouter = require("./routes/cases");  
var applicationsRouter = require("./routes/applications");  
var submissionsRouter = require("./routes/submissions");  
var attachmentsRouter = require("./routes/attachments");  
const fileReferencesRouter = require("./routes/fileReferences");  
var app = express();  
const sequelize = require('./utils/SQLDBHelper');  
const AdrBlkModel = require('./models/AdrBlk\_T');  
const SysMetaDataModel = require('./models/Sys\_Meta\_Data\_T');  
sequelize.authenticate().then(() => {  
 sequelize.sync().then(() => {  
 console.log("Tables synced");  
 });  
 console.log("Connection has been established successfully.");  
});  
app.set("views", path.join(\_\_dirname, "views"));  
app.set("view engine", "jade");  
app.use(logger("dev"));  
app.use(cors());  
app.use(express.json());  
app.use(express.urlencoded({ extended: false }));  
app.use(cookieParser());  
app.use(express.static(path.join(\_\_dirname, "public")));  
app.use((process.env.ROUTER\_ROOT ?? "") + "/", indexRouter);  
app.use((process.env.ROUTER\_ROOT ?? "") + "/auth", authRouter);  
app.use((process.env.ROUTER\_ROOT ?? "") + "/users", usersRouter);  
app.use((process.env.ROUTER\_ROOT ?? "") + "/tasks", tasksRouter);  
app.use((process.env.ROUTER\_ROOT ?? "") + "/cases", casesRouter);  
app.use((process.env.ROUTER\_ROOT ?? "") + "/applications", applicationsRouter);  
app.use((process.env.ROUTER\_ROOT ?? "") + "/submissions", submissionsRouter);  
app.use((process.env.ROUTER\_ROOT ?? "") + "/attachments", attachmentsRouter);  
app.use((process.env.ROUTER\_ROOT ?? "") + "/file-references", fileReferencesRouter);  
app.use(function (req, res, next) {  
 next(createError(404));  
});  
app.use(function (err, req, res, next) {  
 console.log(err);  
 res.locals.message = err.message;  
 res.locals.error = req.app.get("env") === "development" ? err : {};  
 res.status(err.status || 500).send({ error: err.message });  
});  
module.exports = app;

## File: bd-scs-backend-backend/package.json

{  
 "name": "environment",  
 "version": "0.0.0",  
 "private": true,  
 "scripts": {  
 "start": "node ./bin/www"  
 },  
 "dependencies": {  
 "@aws-sdk/client-ses": "^3.687.0",  
 "@signpdf/placeholder-pdf-lib": "^3.2.4",  
 "@signpdf/signer-p12": "^3.2.4",  
 "@signpdf/signpdf": "^3.2.4",  
 "axios": "^1.7.7",  
 "bcryptjs": "^2.4.3",  
 "cookie-parser": "~1.4.4",  
 "cors": "^2.8.5",  
 "csv-parser": "^3.1.0",  
 "debug": "~2.6.9",  
 "docx": "^9.2.0",  
 "docx-merger": "^1.2.2",  
 "docx-templates": "^4.13.0",  
 "docx4js": "^3.3.0",  
 "dotenv": "^16.4.5",  
 "express": "~4.16.1",  
 "http-errors": "~1.6.3",  
 "https-proxy-agent": "^7.0.6",  
 "jade": "~1.11.0",  
 "jsonwebtoken": "^9.0.2",  
 "libreoffice-convert": "^1.6.0",  
 "mongodb": "^6.8.0",  
 "mongoose": "^8.5.2",  
 "morgan": "~1.9.1",  
 "multer": "^1.4.5-lts.1",  
 "node-forge": "^1.3.1",  
 "nodemailer": "^6.10.0",  
 "oauth2-server": "^3.1.1",  
 "pdf-merger-js": "^5.1.2",  
 "sequelize": "^6.37.5",  
 "sequelize-cli": "^6.6.2",  
 "tedious": "^18.6.1",  
 "uuid": "^10.0.0",  
 "word-extractor": "^1.0.4",  
 "xlsx": "^0.18.5"  
 }  
}

## File: bd-scs-backend-backend/README.md

\_\_\_ \_\_\_\_\_\_ \_\_\_\_ \_ \_ \_\_\_   
 / \ \ / / \_\_\_| / \_\_\_| | \_\_\_ \_ \_ \_\_| |/ \_ \   
 / \_ \ \ /\ / /\\_\_\_ \ | | | |/ \_ \| | | |/ \_` | (\_) |  
 / \_\_\_ \ V V / \_\_\_) | | |\_\_\_| | (\_) | |\_| | (\_| |\\_\_, |  
 /\_/ \\_\\_/\\_/ |\_\_\_\_/ \\_\_\_\_|\_|\\_\_\_/ \\_\_,\_|\\_\_,\_| /\_/   
 -----------------------------------------------------------------   
  
  
Hi there! Welcome to AWS Cloud9!  
  
To get started, create some files, play with the terminal,  
or visit https://docs.aws.amazon.com/console/cloud9/ for our documentation.  
  
Happy coding!

## File: bd-scs-backend-web/src/apis/application.js

import { request } from './index'  
export const createApplication = (payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/applications`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getApplication = (id) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/applications/${id}`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getSubmission = (submissionId) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/submissions/${submissionId}`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getSubmissions = (applicationId) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/applications/${applicationId}/submissions`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const createSubmission = (applicationId, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/applications/${applicationId}/submissions`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const updateApplication = (id, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/applications/${id}`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const addAttachments = (applicationId, submissionId, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/applications/${applicationId}/submissions/${submissionId}/attachments`,  
 data: payload,  
 headers: {  
 'Content-Type': 'multipart/form-data',  
 },  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const downloadAttachment = (attachmentId) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/attachments/${attachmentId}`,  
 responseType: 'blob',  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getAdrBlk = (filter) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: '/applications/find-address',  
 data: filter,  
 })  
 .then((res) => {  
 console.log(res)  
 resolve(res.data)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getApplicationHistory = (filter) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/file-references/application-histories`,  
 data: filter,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getFileReferences = (filter) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/file-references`,  
 data: filter,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const assignGR = (id, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/submissions/${id}/assignGR`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}

## File: bd-scs-backend-web/src/apis/auth.js

import { request } from './index'  
export const login = (payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/auth/token`,  
 data: payload,  
 headers: {  
 'Content-Type': 'application/x-www-form-urlencoded',  
 },  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}

## File: bd-scs-backend-web/src/apis/case.js

import { request } from './index'  
import axios from 'axios'  
export const fetchCaseDetail = (id) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/cases/${id}`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const createCase = (id, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/applications/${id}/cases`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const createSiteInSpection = (id, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/cases/${id}/siteinspection`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const updateCase = (id, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/cases/${id}`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const createMinutes = (id, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/cases/${id}/eminutes`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const fetchEMinutes = (caseId, id) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/cases/${caseId}/eminutes/${id}`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const updateEMinutes = (caseId, id, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/cases/${caseId}/eminutes/${id}`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const addAttachments = (payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/attachments`,  
 data: payload,  
 headers: {  
 'Content-Type': 'multipart/form-data',  
 },  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const deleteAttachment = (id) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'delete',  
 url: `/attachments/${id}`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const updateAttachment = (id, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/attachments/${id}`,  
 data: payload,  
 headers: {  
 'Content-Type': 'multipart/form-data',  
 },  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const downloadAttachment = (attachmentId) => {  
 console.log('attachmentId', attachmentId)  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/attachments/${attachmentId}`,  
 responseType: 'blob',  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getCaseSummary = () => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/cases/caseSummary`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const resetLetter = (caseId) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/attachments/${caseId}/resetLetter`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const issueLetter = (caseId) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/attachments/${caseId}/issueLetter`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const signAttachment = (id, payload) => {  
 console.log('signAttachment', id, payload)  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/attachments/sign/${id}`,  
 data: payload,  
 headers: {  
 'Content-Type': 'multipart/form-data',  
 },  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}

## File: bd-scs-backend-web/src/apis/letterTemplate.js

const LETTER\_TEMPLATES = {  
 appendix\_x: {  
 name: 'Appendix X Template',  
 },  
 appendix\_ii: {  
 name: 'Appendix II Template',  
 path: 'letter\_templates/lu\_appendix\_ii\_template.docx',  
 },  
 appendix\_xi: {  
 name: 'Appendix XI Template',  
 path: 'letter\_templates/lu\_appendix\_xi\_template.docx',  
 },  
 '8bl\_no\_objection': {  
 name: '8BL No Objection Template',  
 path: 'letter\_templates/8bl\_no\_objection\_template.docx',  
 },  
}  
export { LETTER\_TEMPLATES }

## File: bd-scs-backend-web/src/apis/task.js

import { request } from './index'  
export const getTask = (id) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/tasks/${id}`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}

## File: bd-scs-backend-web/src/apis/user.js

import { request } from './index'  
export const fetchProfile = () => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/users/me`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const createUser = (payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/users`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getUser = (id) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/users/${id}`,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getUserPositions = (filter = {}) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/users/positions`,  
 params: filter,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getUsers = (filter = {}) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/users`,  
 params: filter,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const getMyTasks = (filter = {}) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'get',  
 url: `/tasks/me`,  
 params: filter,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const updateTask = (id, data) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/tasks/${id}`,  
 data: data,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const updateUser = (id, payload) => {  
 console.log(id, payload)  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'post',  
 url: `/users/${id}`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}  
export const delegateUser = (id, payload) => {  
 return new Promise((resolve, reject) => {  
 return request({  
 method: 'patch',  
 url: `/users/${id}/delegate`,  
 data: payload,  
 })  
 .then((res) => {  
 resolve(res)  
 })  
 .catch((error) => {  
 reject(error)  
 })  
 })  
}

## File: bd-scs-backend-web/src/components/ApplicationForm/CommonForm.jsx

import { Controller } from 'react-hook-form'  
import Grid from '@mui/material/Grid'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import KeyboardArrowDownIcon from '@mui/icons-material/KeyboardArrowDown'  
import { MenuItem, Select } from '@mui/material'  
const CommonForm = ({ control, name, isDesktop, disabled }) => {  
 return (  
 <Grid container>  
 <ResponsiveGrid isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name={`${name}.NameEN`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="English Name"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name={`${name}.NameCN`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Chinese Name"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <FormControl  
 title="Certificate of Registration Number"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <Grid item container direction="row" spacing={1}>  
 <Grid item xs={3}>  
 <Controller  
 name={`${name}.RegistrationNumberPrefix`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <Select  
 sx={{  
 height: 39,  
 width: '100%',  
 bgcolor: 'white',  
 }}  
 IconComponent={KeyboardArrowDownIcon}  
 {...field}  
 >  
 <MenuItem key={0} value={'AP(A)'}>  
 AP(A)  
 </MenuItem>  
 <MenuItem key={0} value={'AP(E)'}>  
 AP(E)  
 </MenuItem>  
 <MenuItem key={0} value={'AP(S)'}>  
 AP(S)  
 </MenuItem>  
 <MenuItem key={0} value={'RSE'}>  
 RSE  
 </MenuItem>  
 </Select>  
 )}  
 />  
 </Grid>  
 <Grid item xs={9}>  
 <Controller  
 name={`${name}.RegistrationNumber`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField  
 {...field}  
 disabled={disabled}  
 placeholder="e.g. 99/99"  
 fullWidth  
 />  
 )}  
 />  
 </Grid>  
 </Grid>  
 </FormControl>  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name={`${name}.Email`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Email"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} type="email" disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name={`${name}.Mobile`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Mobile Number"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} type="tel" disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </Grid>  
 )  
}  
export default CommonForm

## File: bd-scs-backend-web/src/components/ApplicationForm/constants.jsx

export const SchoolDeacriptionList = [  
 { label: 'Kindergarten', id: 'kinderg' },  
 { label: 'Primary', id: 'primary' },  
 {  
 label: 'Secondary',  
 id: 'secondary',  
 },  
 {  
 label: 'Tutorial School',  
 id: 'tutorial\_sch',  
 },  
 {  
 label: 'Other School',  
 id: 'other\_sch',  
 },  
]  
export const AgeOfStudentList = [  
 {  
 label: '3 years old or above',  
 id: '3above',  
 },  
 {  
 label: '6 years old or above',  
 id: '6above',  
 },  
]  
export const Category = [  
 { id: 'CCC', label: 'Child Care Centre (non-purpose built)' },  
 { id: 'KIND', label: 'Kindergarden (non-purpose built)' },  
 {  
 id: 'NLHE',  
 label: 'Non-local Higher Education (Non-local Higher Education)',  
 },  
 { id: 'SCH', label: 'School (non-purpose built)' },  
]  
export const Nature = [  
 {  
 id: 'ALT',  
 label: 'Alteration to licensed premises(Alteration to licensed premises)',  
 },  
 {  
 id: 'NEW',  
 label: 'New application except AVP(New application except AVP)',  
 },  
 { id: 'REV', label: 'Revised plan/proposal(Revised plan/proposal)' },  
 {  
 id: 'RNL',  
 label:  
 'License renewal or structural inspection for CCC(License renewal or structural inspection for CCC)',  
 },  
 { id: 'SCSAUD', label: 'Self-certification' },  
]  
export const BuildingType = [  
 { id: 'IND', value: 'IND', label: 'Industrial' },  
 { id: 'Commercial', value: 'Commercial', label: 'Commercial' },  
 {  
 id: 'COM',  
 value: 'COM',  
 label: 'Composite',  
 },  
 { id: 'SFH', value: 'SFH', label: 'Single Family House' },  
 {  
 id: 'NTEB',  
 value: 'NTEB',  
 label: 'New Territories "Exempted" Building (NTEH)',  
 },  
 {  
 id: 'SSB-NTEB',  
 label:  
 'Single staircase building other than single-family domestic building & New Territories "Exempted" Building (NTEH)',  
 },  
 { id: 'INST', value: 'INST', label: 'Institution' },  
]  
export const PremiseType = [  
 {  
 id: 'SCH',  
 value: 'SCH',  
 label: 'School',  
 },  
 {  
 id: 'KG',  
 value: 'KG',  
 label: 'Kindergarten',  
 },  
 { id: 'CCC', value: 'CCC', label: 'Child Care Centre' },  
 { id: 'NLHE', value: 'NLHE', label: 'Non-local Higher Education' },  
]  
export const FloorLevels = [  
 {  
 id: 'G',  
 label: 'G/F',  
 },  
 {  
 id: '1',  
 label: '1/F',  
 },  
 {  
 id: '2',  
 label: '2/F',  
 },  
 {  
 id: '3',  
 label: '3/F',  
 },  
 {  
 id: 'WB',  
 label: 'Whole Building',  
 },  
]  
export const Region = [  
 { id: 'HK', label: 'Hong Kong(香港)' },  
 { id: 'K', label: 'Kowloon(九龍)' },  
 { id: 'NT', label: 'New Territories(新界)' },  
]  
export const ThreeTierReqt = [  
 { id: '0', label: 'Without requirement(Without requirement)' },  
 { id: '2', label: 'Non-Cat. 3 requirements(Non-Cat. 3 requirements)' },  
 { id: '3', label: 'Cat. 3 requirements(Cat. 3 requirements)' },  
]  
export const ObjectiontoLR = [  
 { id: 'Y', label: 'Y' },  
 { id: 'N', label: 'N' },  
]  
export const AuditResult = [  
 { id: 'ACC', label: 'Accepted(Accepted)' },  
 { id: 'C', label: 'Clarification required(Clarification required)' },  
 { id: 'P', label: 'Pass(Pass)' },  
 { id: 'REJ', label: 'Rejected(Rejected)' },  
]  
export const ReferrerGovDepartment = [  
 { id: 'AFCD', label: 'Agriculture, Fisheries and Conservation Department' },  
 { id: 'ASD', label: 'Architectural Services Department' },  
 { id: 'CSO', label: `Chief Secretary for Administration's Office` },  
 { id: 'CU', label: 'Buildings Department - canopy Unit' },  
 { id: 'DO', label: 'District Office' },  
 { id: 'DOH', label: 'Department of Health' },  
 { id: 'DOJ', label: 'Department of Justice' },  
 { id: 'DSD', label: 'Drainage Services Department' },  
 { id: 'EBD', label: 'Buildings Department - Existing Building Division' },  
 { id: 'ED', label: 'Education Department' },  
 { id: 'EMSD', label: 'Electrical and Mechanical Services Department' },  
 { id: 'EPD', label: 'Enviromental Protection Department' },  
 { id: 'FEHD', label: 'Food and Enviromental Hygiene Department' },  
 { id: 'FEHDJ', label: 'FEHD - Jaint Office' },  
 {  
 id: 'FSCBP',  
 label:  
 'Fire Services Department(Commercial Buildings and Premises Division)',  
 },  
 { id: 'FSD', label: 'Fire Services Department' },  
 { id: 'FSS', label: 'Buildings Department - Fire Safety Section' },  
 { id: 'GEO', label: 'Geotechnical Engineering Office' },  
 { id: 'HAD', label: 'Home Affairs Department' },  
 { id: 'HD', label: 'Housing Department' },  
 { id: 'HKPF', label: 'HK Police Force' },  
 { id: 'HPLB', label: 'Housing, Planning and Lands Bureau' },  
 { id: 'HYD', label: 'Highways Department' },  
 { id: 'IU', label: 'Buildings Department - Information Unit' },  
 { id: 'LAND', label: 'Lands Department' },  
 { id: 'LCSD', label: 'Leisure and Cultural Service Department' },  
 { id: 'LD', label: 'Labour Department' },  
 { id: 'LR', label: 'Land Registrar' },  
 { id: 'LU', label: 'Buildings Department - Licensing Unit' },  
 { id: 'NBD', label: 'Buildings Department - New Buildings Division' },  
 { id: 'PD', label: 'Planning Department' },  
 { id: 'SAU', label: 'Buildings Department - Special Action Unit' },  
 { id: 'SCU', label: 'Lands Department - Squatter Control' },  
 { id: 'SSS', label: 'Buildings Department - Slope Safety Section' },  
 { id: 'SWD', label: 'Social Welfare Department' },  
 { id: 'TD', label: 'Transport Department' },  
 { id: 'TELA', label: 'Television & Entertainment Licensing Authority' },  
 { id: 'WSD', label: 'Water Supplies Department' },  
]  
export const ApplicationType = [  
 { label: 'School', id: 'SCH' },  
 { label: 'Child Care Centre', id: 'CCC' },  
 { label: 'Non Local Higher Education Course', id: 'NLHE' },  
]

## File: bd-scs-backend-web/src/components/ApplicationForm/index.jsx

import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import React from 'react'  
import Stack from '@mui/material/Stack'  
import Grid from '@mui/material/Grid'  
import Divider from '@mui/material/Divider'  
import Box from '@mui/material/Box'  
import Button from 'components/basic/Button'  
import Typography from 'components/basic/Typography'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Autocomplete from 'components/basic/Autocomplete'  
import RelatedPremisesTable from './PremisesTable'  
import AddressDialog from 'components/Dialog/AddressDialog'  
import ClearOutlinedIcon from '@mui/icons-material/ClearOutlined'  
import {  
 SchoolDeacriptionList,  
 AgeOfStudentList,  
 ApplicationType,  
} from 'components/ApplicationForm/constants'  
import CommonForm from './CommonForm'  
import FileReferenceDialog from 'components/Dialog/FileReferenceDialog'  
function ApplicationForm({ control, setValue, disabled, type = '' }) {  
 const { fields, append, remove } = useFieldArray({  
 control,  
 name: 'APP13',  
 })  
 const [addressDialogConfig, setAddressDialogConfig] = React.useState({  
 open: false,  
 index: null,  
 })  
 const watchApplicationType = useWatch({ control, name: 'ApplicationType' })  
 const [fileReferenceDiaglogOpen, setFileReferenceDiaglogOpen] =  
 React.useState(false)  
 return (  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <>  
 <Typography variant="title2" color="text.secondary">  
 Licensing Premises Information  
 </Typography>  
 {type != 'edit' && (  
 <ResponsiveGrid isDesktop={isDesktop} spacing={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ApplicationType"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Application Type"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.id?.message}  
 fullWidth  
 required  
 >  
 <Autocomplete  
 options={ApplicationType}  
 getOptionLabel={(option) => option.label}  
 {...field}  
 value={!field?.value?.id ? null : field?.value}  
 onChange={(e, value) => field.onChange(value)}  
 disableClearable  
 disabled={disabled}  
 />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicationNo"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Application Number"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled fullWidth />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 )}  
 {type != 'edit' && (  
 <ResponsiveGrid isDesktop={isDesktop} spacing={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="FileReferenceDisplay"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="File Reference"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth  
 required  
 >  
 <Stack direction="row" gap={2}>  
 <TextField fullWidth {...field} disabled />  
 <Button  
 onClick={() => {  
 setFileReferenceDiaglogOpen(true)  
 }}  
 >  
 Search  
 </Button>  
 </Stack>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="Region"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Region"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField disabled {...field} fullWidth />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 )}  
 {type != 'edit' && (  
 <ResponsiveGrid isDesktop={isDesktop} spacing={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="District"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="District"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField disabled {...field} fullWidth />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="Area"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Area"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField disabled {...field} fullWidth />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 )}  
 <ResponsiveGrid container isDesktop={isDesktop} spacing={3} mb={4}>  
 <Grid item md={8}>  
 <Controller  
 name="AddressOfPremiseEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Applicant Provided Licensing Address (English)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} fullWidth />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item md={2}>  
 <Controller  
 name="AddressOfPremiseENFloor"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Floor(s)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Box display="flex" alignItems="center">  
 <TextField  
 {...field}  
 fullWidth  
 placeholder="E.g. 1-3, UG, G, B1"  
 />  
 <Typography sx={{ ml: 1 }}>/F</Typography>  
 </Box>  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item md={2}>  
 <Controller  
 name="AddressOfPremiseENUnit"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Unit / Room(s)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 placeholder="E.g. 1001-1003, A, B"  
 fullWidth  
 />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} spacing={3} mb={4}>  
 <Grid item md={8}>  
 <Controller  
 name="AddressOfPremiseCN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Applicant Provided Licensing Address (Chinese)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} fullWidth />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item md={2}>  
 <Controller  
 name="AddressOfPremiseCNFloor"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Floor(s)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Box display="flex" alignItems="center">  
 <TextField  
 {...field}  
 placeholder="E.g. 1-3, 地下, 地庫1"  
 fullWidth  
 />  
 <Typography sx={{ ml: 1 }}>樓</Typography>  
 </Box>  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item md={2}>  
 <Controller  
 name="AddressOfPremiseCNUnit"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Unit / Room(s)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 placeholder="E.g. 1001-1003, A, B"  
 fullWidth  
 />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 {  
}  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="NameOfSchoolEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="School Name / Centre Name (English)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField {...field} disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="NameOfSchoolCN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="School Name / Centre Name (Chinese)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {(watchApplicationType?.id === 'SCH' || type == 'edit') && (  
 <>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="DescriptionOfSchool"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Description of School"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth={true}  
 error={invalid}  
 helperText={error?.message}  
 >  
 <Autocomplete  
 options={SchoolDeacriptionList}  
 getOptionLabel={(option) => option.label}  
 {...field}  
 value={!field?.value?.id ? null : field?.value}  
 onChange={(e, value) => field.onChange(value)}  
 disableClearable  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="AgeOfStudent"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Age of Student"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth={true}  
 error={invalid}  
 helperText={error?.message}  
 >  
 <Autocomplete  
 options={AgeOfStudentList}  
 getOptionLabel={(option) => option.label}  
 {...field}  
 value={!field?.value?.id ? null : field?.value}  
 onChange={(e, value) => field.onChange(value)}  
 disableClearable  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="EstimatedNoOfStudent"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Estimated Number of Students"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth={true}  
 error={invalid}  
 helperText={error?.message}  
 >  
 <TextField {...field} disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {isDesktop && <Grid item xs />}  
 </ResponsiveGrid>  
 </>  
 )}  
 <RelatedPremisesTable  
 control={control}  
 name="RelatedPremises"  
 setValue={setValue}  
 disabled={disabled}  
 />  
 <Divider sx={{ my: 4 }} />  
 <Typography variant="title2" color="text.secondary">  
 Applicant  
 </Typography>  
 <Grid container direction="column" gap={3} mb={4}>  
 <ResponsiveGrid isDesktop={isDesktop} gap={3}>  
 <Grid item xs>  
 <Controller  
 name="ApplicantNameEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Applicant Name"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} disabled={disabled} />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantNameCN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Applicant Name(Chinese)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} disabled={disabled} />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <Controller  
 name="ApplicantAddress"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Correspondence Address"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} disabled={disabled} />  
 </FormControl>  
 )  
 }}  
 />  
 <ResponsiveGrid isDesktop={isDesktop} gap={3}>  
 <Grid item xs>  
 <Controller  
 name="ApplicantTel"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Telephone Number(School / Centre)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled={disabled} type="tel" />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantFax"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Fax Number(School / Centre)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled={disabled} type="tel" />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid isDesktop={isDesktop} gap={3}>  
 <Grid item xs>  
 <Controller  
 name="ContactPersonEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Contact Person"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled={disabled} />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ContactPersonCN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormControl  
 title="Contact Person(Chinese)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled={disabled} />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid isDesktop={isDesktop} gap={3}>  
 <Grid item xs>  
 <Controller  
 name="ApplicantEmail"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Email Address(Applicant)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} type="email" disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ContactPersonEmail"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Email Address(Contact Person)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} type="email" disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid isDesktop={isDesktop} gap={3}>  
 <Grid item xs>  
 <Controller  
 name="ApplicantMobile"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Mobile Number(Applicant)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} type="tel" disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ContactPersonTel"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Telephone Number(Contact Person)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} type="tel" disabled={disabled} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </Grid>  
 <Typography variant="title2" color="text.secondary">  
 Information of Appointed Authorized Person (AP) / Registered  
 Structural Engineer (RSE) for submitting supporting documents for  
 the application :  
 </Typography>  
 {fields.map((item, index) => (  
 <Box display="flex" flexDirection="row">  
 <CommonForm  
 control={control}  
 name={`APP13.${index}`}  
 isDesktop={isDesktop}  
 disabled={disabled}  
 />  
 <ClearOutlinedIcon  
 sx={{  
 ml: '10px',  
 cursor: index ? 'pointer' : 'not-allowed',  
 }}  
 onClick={() => {  
 if (!index) return  
 remove(index)  
 }}  
 />  
 </Box>  
 ))}  
 <Button  
 onClick={() =>  
 append({  
 NameEN: '',  
 NameCN: '',  
 RegistrationNumberPrefix: '',  
 RegistrationNumber: '',  
 Email: '',  
 Mobile: '',  
 })  
 }  
 >  
 Add  
 </Button>  
 <Divider sx={{ my: 4 }} />  
 <Typography variant="title2" color="text.secondary">  
 Information of Appointed Authorized Person (AP) / Registered  
 Structural Engineer (RSE) for Self-Certification System  
 </Typography>  
 <CommonForm  
 control={control}  
 name="SelfCertification"  
 isDesktop={isDesktop}  
 disabled={disabled}  
 />  
 <Divider sx={{ my: 4 }} />  
 {/\* <Divider sx={{ my: 4 }} /> \*/}  
 <AddressDialog  
 open={addressDialogConfig?.open}  
 onClose={() => setAddressDialogConfig({ open: false })}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 setValue(`address`, address)  
 }  
 // if (inputManually) {  
 // setValue('isEngAddressReadOnly', false)  
 }}  
 />  
 <FileReferenceDialog  
 open={fileReferenceDiaglogOpen}  
 onClose={() => setFileReferenceDiaglogOpen(false)}  
 onChange={(newFileRef) => {  
 console.log('newFileRef', newFileRef.fileRef)  
 setValue('FileReference', newFileRef.sysFileRefId)  
 setValue('FileReferenceDisplay', newFileRef.display)  
 }}  
 />  
 </>  
 )}  
 </ResponsiveForm>  
 )  
}  
export default ApplicationForm

## File: bd-scs-backend-web/src/components/ApplicationForm/OldCommonForm.jsx

import { Controller } from 'react-hook-form'  
import Grid from '@mui/material/Grid'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
const CommonForm = ({ control, name, isDesktop, disabled }) => {  
 return (  
 <>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name={`${name}.RegistrationNumber`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title="Registration Number"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name={`${name}.Name`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title="Name"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name={`${name}.Email`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title="Email"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 type="email"  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name={`${name}.Mobile`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title="Mobile Number"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 type="tel"  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </>  
 )  
}  
export default CommonForm

## File: bd-scs-backend-web/src/components/ApplicationForm/PremisesTable.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled'  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import Grid from '@mui/material/Grid'  
import Divider from '@mui/material/Divider'  
import IconButton from '@mui/material/IconButton'  
import Typography from 'components/basic/Typography'  
import TextField from 'components/basic/TextField'  
import DatePicker from 'components/basic/DatePicker'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Button from 'components/basic/Button'  
import FormControl from 'components/basic/FormControl'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import AddressDialog from 'components/Dialog/AddressDialog'  
import AddCircleOutlinedIcon from '@mui/icons-material/AddCircleOutlined'  
import DeleteOutlineIcon from '@mui/icons-material/DeleteOutline'  
import colors from 'src/constants/colors'  
const Container = styled(Box)`  
 display: flex;  
 flex-direction: column;  
 flex-wrap: nowrap;  
`  
const Header = styled(Grid)`  
 width: ${(props) => (props.w ? `${props.w}px` : 'none')};  
 height: 60px;  
 color: #fff;  
`  
const ContentContiner = styled(Grid)`  
 height: 60px;  
 align-items: center;  
 margin-top: 0.5px;  
`  
function buildAddressEN(ad) {  
 const parts = [  
 ad.osadR\_E1,  
 ad.osadR\_E2,  
 ad.osadR\_E3,  
 ad.osadR\_E4,  
 ad.osadR\_E5,  
 ad.bldG\_NAME\_E1,  
 ad.osadR\_LOT\_E1,  
 ad.osadR\_LOT\_E2,  
 ad.osadR\_LOT\_E3,  
 ad.osadR\_LOT\_E4,  
 ad.osadR\_ST\_E1,  
 ad.osadR\_ST\_E2,  
 ad.district\_e,  
 ]  
 return parts.filter((part) => part && part.trim()).join(', ')  
}  
function buildAddressZH(ad) {  
 const parts = [  
 ad.district\_c,  
 ad.bldG\_NAME\_C1,  
 ad.osadR\_ST\_C,  
 ad.osadR\_LOT\_C1,  
 ad.osadR\_LOT\_C2,  
 ad.osadR\_LOT\_C3,  
 ad.osadR\_LOT\_C4,  
 ad.osadR\_C1,  
 ad.osadR\_C2,  
 ad.osadR\_C3,  
 ad.osadR\_C4,  
 ad.osadR\_C5,  
 ]  
 return parts.filter((part) => part && part.trim()).join('')  
}  
function RelatedPremisesTable({ control, t, sx, name, setValue, disabled }) {  
 const [addressDialogConfig, setAddressDialogConfig] = React.useState({  
 open: false,  
 index: null,  
 })  
 const [isDisabled, setIsDisabled] = React.useState(disabled)  
 const { fields, append, remove } = useFieldArray({  
 control,  
 name: `${name}`,  
 })  
 const onClickAdd = () => {  
 append({  
 ID: '',  
 Type: '',  
 AdditionalInfo: '',  
 OPYear: '',  
 Address: '',  
 })  
 }  
 const onClickDel = (index) => {  
 if (!index) return  
 remove(index)  
 }  
 return (  
 <Container sx={sx}>  
 <Grid  
 container  
 direction="row"  
 justifyContent="space-between"  
 alignItems="center"  
 my={1}  
 >  
 <Grid item>  
 <Typography sx={{ fontWeight: 'bold' }} color="text.secondary">  
 Related Premises  
 </Typography>  
 </Grid>  
 {/\* {fields.length <= 0 && (  
 <Grid item>  
 <Button onClick={onClickAdd}>Add</Button>  
 </Grid>  
 )} \*/}  
 </Grid>  
 <Box borderRadius={4} bgcolor="#fff">  
 <Grid  
 container  
 direction="row"  
 sx={{  
 backgroundColor: colors.main,  
 borderTopLeftRadius: 4,  
 borderTopRightRadius: 4,  
 }}  
 >  
 <Grid item>  
 <Header item container alignItems="center" px={2} w={100}>  
 <Typography>ID</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header item container alignItems="center" px={2} w={150}>  
 <Typography>Type</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header item container alignItems="center" px={2} w={200}>  
 <Typography>Additional Info.</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header item container alignItems="center" px={2} w={200}>  
 <Typography>{`OP Year(Age)`}</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header item container alignItems="center" px={2} xs>  
 <Typography>Address</Typography>  
 </Header>  
 </Grid>  
 </Grid>  
 <Grid container direction="row">  
 {/\* {fields.map((item, index) => {  
 return (  
 <Grid item container direction="row" key={`fields-${index}`}>  
 <ContentContiner item container px={2} sx={{ width: 100 }}>  
 <Controller  
 name={`${name}.${index}.ID`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} disabled />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} sx={{ width: 150 }}>  
 <Controller  
 name={`${name}.${index}.Type`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} disabled />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} sx={{ width: 200 }}>  
 <Controller  
 name={`${name}.${index}.AdditionalInfo`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} disabled />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} sx={{ width: 200 }}>  
 <Controller  
 name={`${name}.${index}.OPYear`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} disabled />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} xs>  
 <Controller  
 name={`${name}.${index}.Address`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} fullWidth disabled />  
 )}  
 />  
 </ContentContiner>  
 {!disabled && (  
 <Box  
 display="flex"  
 alignItems="center"  
 justifyContent="center"  
 mx={2}  
 >  
 <Button  
 onClick={() => {  
 setAddressDialogConfig({  
 open: true,  
 index: index,  
 })  
 setIsDisabled(true)  
 }}  
 >  
 Set Addr  
 </Button>  
 </Box>  
 )}  
 </Grid>  
 )  
 })} \*/}  
 <Grid item container direction="row" key={`related-premise`}>  
 <ContentContiner item container px={2} sx={{ width: 100 }}>  
 <Controller  
 name={`RelatedPremise`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} disabled />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} sx={{ width: 150 }}>  
 <Controller  
 name={`RelatedPremiseType`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} disabled />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} sx={{ width: 200 }}>  
 <Controller  
 name={`RelatedPremiseAdditionalInfo`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} disabled />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} sx={{ width: 200 }}>  
 <Controller  
 name={`RelatedPremiseOPYear`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} disabled />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} xs>  
 <Controller  
 name={`RelatedPremiseAddress`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} fullWidth disabled />  
 )}  
 />  
 </ContentContiner>  
 {!disabled && (  
 <Box  
 display="flex"  
 alignItems="center"  
 justifyContent="center"  
 mx={2}  
 >  
 <Button  
 onClick={() => {  
 setAddressDialogConfig({  
 open: true,  
 })  
 setIsDisabled(true)  
 }}  
 >  
 Set Addr  
 </Button>  
 </Box>  
 )}  
 </Grid>  
 </Grid>  
 </Box>  
 <AddressDialog  
 open={addressDialogConfig?.open}  
 onClose={() => setAddressDialogConfig({ open: false, index: null })}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 if (addressDialogConfig.index === null) return  
 const id = address.adR\_BLK\_ID  
 setValue('Area', address.area\_e)  
 setValue('Region', address.region\_e)  
 setValue('District', address.district\_e)  
 setValue('RelatedPremise', id)  
 setValue('RelatedPremiseType', address.bt\_e)  
 setValue('RelatedPremiseAddress', buildAddressEN(address))  
 setValue('AddressOfPremiseEN', buildAddressEN(address))  
 setValue('AddressOfPremiseCN', buildAddressZH(address))  
 }  
 }}  
 />  
 </Container>  
 )  
}  
export default RelatedPremisesTable

## File: bd-scs-backend-web/src/components/ApplicationForm/RelatedPremisesTable.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled'  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import Grid from '@mui/material/Grid'  
import Divider from '@mui/material/Divider'  
import IconButton from '@mui/material/IconButton'  
import Typography from 'components/basic/Typography'  
import TextField from 'components/basic/TextField'  
import DatePicker from 'components/basic/DatePicker'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Button from 'components/basic/Button'  
import FormControl from 'components/basic/FormControl'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import AddressDialog from 'components/Dialog/AddressDialog'  
import AddCircleOutlinedIcon from '@mui/icons-material/AddCircleOutlined'  
import DeleteOutlineIcon from '@mui/icons-material/DeleteOutline'  
import colors from 'src/constants/colors'  
const Container = styled(Box)`  
 display: flex;  
 flex-direction: column;  
 flex-wrap: nowrap;  
`  
const Header = styled(Grid)`  
 width: ${(props) => (props.w ? `${props.w}px` : 'none')};  
 height: 60px;  
 color: #fff;  
`  
const ContentContiner = styled(Grid)`  
 height: 60px;  
 align-items: center;  
 margin-top: 0.5px;  
`  
function RelatedPremisesTable({ control, t, sx, name, setValue, disabled }) {  
 const [addressDialogConfig, setAddressDialogConfig] = React.useState({  
 open: false,  
 index: null,  
 })  
 const { fields, append, remove } = useFieldArray({  
 control,  
 name: `${name}`,  
 })  
 const onClickAdd = () => {  
 append({  
 ID: '',  
 Type: '',  
 AdditionalInfo: '',  
 OPYear: '',  
 Address: '',  
 })  
 }  
 const onClickDel = (index) => {  
 if (!index) return  
 remove(index)  
 }  
 return (  
 <Container sx={sx}>  
 <Grid  
 container  
 direction="row"  
 justifyContent="space-between"  
 alignItems="center"  
 my={1}  
 >  
 <Grid item>  
 <Typography  
 sx={{ fontWeight: 'bold' }}  
 color="text.secondary"  
 >  
 Related Premises  
 </Typography>  
 </Grid>  
 {!disabled && (  
 <Grid item>  
 <Button onClick={onClickAdd}>Add</Button>  
 </Grid>  
 )}  
 </Grid>  
 <Box borderRadius={4} bgcolor="#fff">  
 <Grid  
 container  
 direction="row"  
 sx={{  
 backgroundColor: colors.main,  
 borderTopLeftRadius: 4,  
 borderTopRightRadius: 4,  
 }}  
 >  
 <Grid item>  
 <Header  
 item  
 container  
 alignItems="center"  
 px={2}  
 w={100}  
 >  
 <Typography>ID</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header  
 item  
 container  
 alignItems="center"  
 px={2}  
 w={150}  
 >  
 <Typography>Type</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header  
 item  
 container  
 alignItems="center"  
 px={2}  
 w={200}  
 >  
 <Typography>Additional Info.</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header  
 item  
 container  
 alignItems="center"  
 px={2}  
 w={200}  
 >  
 <Typography>{`OP Year(Age)`}</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header item container alignItems="center" px={2} xs>  
 <Typography>Address</Typography>  
 </Header>  
 </Grid>  
 </Grid>  
 <Grid container direction="row">  
 {fields.map((item, index) => {  
 return (  
 <Grid  
 item  
 container  
 direction="row"  
 key={`fields-${index}`}  
 >  
 <ContentContiner  
 item  
 container  
 px={2}  
 sx={{ width: 100 }}  
 >  
 <Controller  
 name={`${name}.${index}.ID`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner  
 item  
 container  
 px={2}  
 sx={{ width: 150 }}  
 >  
 <Controller  
 name={`${name}.${index}.Type`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner  
 item  
 container  
 px={2}  
 sx={{ width: 200 }}  
 >  
 <Controller  
 name={`${name}.${index}.AdditionalInfo`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner  
 item  
 container  
 px={2}  
 sx={{ width: 200 }}  
 >  
 <Controller  
 name={`${name}.${index}.OPYear`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} xs>  
 <Controller  
 name={`${name}.${index}.Address`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 {...field}  
 fullWidth  
 disabled={disabled}  
 />  
 )}  
 />  
 </ContentContiner>  
 {!disabled && (  
 <Box  
 display="flex"  
 alignItems="center"  
 justifyContent="center"  
 >  
 <Button  
 onClick={() =>  
 setAddressDialogConfig({  
 open: true,  
 index: index,  
 })  
 }  
 >  
 Set Addr  
 </Button>  
 </Box>  
 )}  
 {!disabled && (  
 <Box  
 display="flex"  
 alignItems="center"  
 justifyContent="center"  
 sx={{  
 cursor: index  
 ? 'pointer'  
 : 'not-allowed',  
 mx: 1,  
 }}  
 onClick={() => onClickDel(index)}  
 >  
 <DeleteOutlineIcon  
 sx={{  
 color: index  
 ? 'red'  
 : 'lightGrey',  
 }}  
 />  
 </Box>  
 )}  
 </Grid>  
 )  
 })}  
 </Grid>  
 </Box>  
 <AddressDialog  
 open={addressDialogConfig?.open}  
 onClose={() =>  
 setAddressDialogConfig({ open: false, index: null })  
 }  
 onChange={({ address, inputManually }) => {  
 console.log('address', address)  
 if (address) {  
 if (addressDialogConfig.index === null) return  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue(  
 `${name}.${addressDialogConfig.index}.Address`,  
 en  
 )  
 }  
 }}  
 />  
 </Container>  
 )  
}  
export default RelatedPremisesTable

## File: bd-scs-backend-web/src/components/Case/BuildingInformation/FloorLevel/Form.jsx

import { Controller } from 'react-hook-form'  
import dayjs from 'dayjs'  
import { GridActionsCellItem, GridToolbarContainer } from '@mui/x-data-grid'  
import MuiStyled from '@mui/material/styles/styled'  
import { useFormControl } from '@mui/material/FormControl'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Grid from '@mui/material/Grid'  
import InputLabel from '@mui/material/InputLabel'  
import Stack from '@mui/material/Stack'  
import { FloorLevels } from 'components/ApplicationForm/constants'  
import Autocomplete from 'components/basic/Autocomplete'  
import GradientButton from 'components/basic/Button/GradientButton'  
import FormControl from 'components/basic/FormControl'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import TextField from 'components/basic/TextField'  
import Typography from 'components/basic/Typography'  
import DataTable from 'components/basic/DataGrid'  
import { useTranslation } from 'react-i18next'  
import { Button, IconButton } from '@mui/material'  
import RemoveCircleIcon from '@mui/icons-material/RemoveCircle'  
import colors from 'src/constants/colors'  
import Chip from '@mui/material/Chip'  
import Link from '@mui/material/Link'  
import constants from 'src/constants'  
import { type } from '@testing-library/user-event/dist/type'  
import { useMemo, useState } from 'react'  
const StyledDataTable = MuiStyled(DataTable)(({ theme }) => {  
 return {  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: colors.main,  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: colors.main,  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 color: '#fff',  
 },  
 '& .MuiDataGrid-columnHeaderCheckbox': {  
 color: '#fff',  
 },  
 '& .MuiButtonBase-root': {  
 color: '#000',  
 },  
 '& .MuiDataGrid-main': {  
 borderTopLeftRadius: 0,  
 borderTopRightRadius: 0,  
 },  
 '.MuiDataGrid-row': {  
 alignItems: 'center',  
 },  
 '.MuiDataGrid-toolbarContainer': {  
 backgroundColor: colors.main,  
 borderTopLeftRadius: '0.75rem',  
 borderTopRightRadius: '0.75rem',  
 },  
 }  
})  
function FloorLevelForm({ control }) {  
 const { t, i18n } = useTranslation(['common'])  
 const floorData = useMemo(() => {  
 return [  
 {  
 id: 'dd8a8ac9-c8dc-489b-b68f-54d7d5b0c065',  
 indexPlan: '1/F Index Plan (1/3)',  
 remark: {  
 fileReference: '2/4161/80',  
 drawingNo: '2/4GE',  
 drawingTitle: '1ST FLOOR PLAN (1/3)',  
 },  
 createdAt: new Date(),  
 updatedAt: new Date(),  
 premises: [  
 {  
 id: '3b1cfc4f-afdf-449a-a954-9b8cf3300b5e',  
 name: 'Shop 101',  
 type: 'School',  
 file: '',  
 },  
 {  
 id: 'b6358cd9-66b3-4109-ae59-a55c7d32379e',  
 name: 'Shop 102',  
 type: 'Child Care Centre',  
 file: '',  
 },  
 ],  
 },  
 ]  
 }, [])  
 const [floor, setFloor] = useState(null)  
 const onSubmit = (data) => {}  
 return (  
 <Stack direction="column">  
 <ResponsiveForm sx={{ m: 3 }}>  
 {(isDesktop) => (  
 <Stack direction="column">  
 {/\* <Box py={3}>  
 <Controller  
 name={'siteID'}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 title="Site ID"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Box>  
 <Box py={3}>  
 <Controller  
 name={'notionalSiteID'}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 title="Notional Site ID"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Box>  
 <Box py={3}>  
 <Controller  
 name={'blockID'}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 title="Block ID"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Box> \*/}  
 <Box py={3}>  
 {  
}  
 <InputLabel>Remark</InputLabel>  
 <Stack direction="row" spacing={1} alignItems={'center'}>  
 <Controller  
 name="prefix"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl error={invalid} helperText={error?.message}>  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 <Typography variant="title">1234/56</Typography>  
 <Controller  
 name="suffix"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 error={invalid}  
 helperText={error?.message}  
 sx={{}}  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Stack>  
 </Box>  
 <Box py={3}>  
 <Controller  
 name="floor"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Floor"  
 labelPlacement="top"  
 fullWidth={true}  
 error={invalid}  
 helperText={error?.message}  
 >  
 <Autocomplete  
 options={FloorLevels}  
 getOptionLabel={(option) => option.label}  
 {...field}  
 value={!field?.value?.id ? null : field?.value}  
 onChange={(e, value) => field.onChange(value)}  
 placeholder="Please Select"  
 disableClearable  
 />  
 </FormControl>  
 )}  
 />  
 </Box>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="drawingNo"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={'Drawing No'}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="drawingTitle"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={'Drawing Title'}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <Divider sx={{ my: 3 }} />  
 <Box>  
 <Button  
 variant="contained"  
 >  
 Upload  
 </Button>  
 </Box>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 {}  
 {floorTable(floorData)}  
 {premiseTable(floorData[0].premises)}  
 </Stack>  
 )  
}  
function floorTable(data) {  
 return (  
 <div  
 style={{  
 display: 'flex',  
 flexDirection: 'column',  
 minHeight: '12.5rem',  
 maxHeight: '50rem',  
 }}  
 >  
 <StyledDataTable  
 aria-label="Floor"  
 api={`${import.meta.env.VITE\_API\_ROOT}/building`}  
 staticData={data}  
 disableRowSelectionOnClick={true}  
 slots={{ toolbar: FloorSelectorToolbar }}  
 getRowHeight={() => 'auto'}  
 columns={[  
 {  
 field: 'no',  
 headerName: '#',  
 headerAlign: 'center',  
 align: 'center',  
 valueGetter: (field, \_data, \_\_\_) => {  
 const index = data.findIndex(({ id }) => {  
 return id === \_data.id  
 })  
 return index + 1  
 },  
 },  
 {  
 field: 'indexPlan',  
 headerName: 'Index Plan',  
 headerAlign: 'center',  
 flex: 1,  
 align: 'center',  
 },  
 {  
 field: 'remark',  
 headerName: 'Remark',  
 headerAlign: 'center',  
 flex: 2,  
 align: 'justify',  
 renderCell: ({ value }) => {  
 return (  
 <Stack>  
 <Typography variant="subtitle">  
 {value.fileReference}  
 </Typography>  
 <Typography variant="subtitle">{value.drawingNo}</Typography>  
 <Typography variant="subtitle">  
 {value.drawingTitle}  
 </Typography>  
 </Stack>  
 )  
 },  
 },  
 {  
 field: 'createdAt',  
 headerName: 'Upload Date',  
 headerAlign: 'center',  
 align: 'center',  
 valueGetter: (field) => {  
 return dayjs(field).format('DD/MM/YYYY')  
 },  
 },  
 {  
 field: 'updatedAt',  
 headerName: 'Last Updated Date',  
 headerAlign: 'center',  
 align: 'center',  
 valueGetter: (field) => {  
 return dayjs(field).format('DD/MM/YYYY')  
 },  
 },  
 {  
 field: 'isDeleted',  
 type: 'actions',  
 headerName: 'Delete',  
 cellClassName: 'actions',  
 headerAlign: 'center',  
 align: 'center',  
 getActions: ({ id, row }) => {  
 return [  
 <GridActionsCellItem  
 icon={<RemoveCircleIcon />}  
 onClick={(\_id) => () => {}}  
 />,  
 ]  
 },  
 },  
 ]}  
 sx={{}}  
 />  
 </div>  
 )  
}  
function premiseTable(data) {  
 return (  
 <div  
 style={{  
 display: 'flex',  
 flexDirection: 'column',  
 minHeight: '12.5rem',  
 maxHeight: '50rem',  
 }}  
 >  
 <StyledDataTable  
 aria-label="Floor"  
 api={`${import.meta.env.VITE\_API\_ROOT}/building`}  
 staticData={data}  
 disableRowSelectionOnClick={true}  
 columns={[  
 {  
 headerName: '#',  
 headerAlign: 'center',  
 align: 'center',  
 valueGetter: (field, \_data, \_\_\_) => {  
 const index = data.findIndex(({ id }) => {  
 return id === \_data.id  
 })  
 return index + 1  
 },  
 },  
 {  
 field: 'name',  
 headerName: 'Premise Name',  
 headerAlign: 'center',  
 flex: 1,  
 align: 'center',  
 renderCell: ({ field, row }) => {  
 if (field === 'name') {  
 return <Link href={row.file}>{row.name}</Link>  
 }  
 },  
 },  
 {  
 field: 'type',  
 headerName: 'Type of Premises',  
 headerAlign: 'center',  
 flex: 1,  
 align: 'center',  
 },  
 ]}  
 sx={{}}  
 />  
 </div>  
 )  
}  
function FloorSelectorToolbar() {  
 return (  
 <GridToolbarContainer>  
 {  
}  
 <Box sx={{ m: '0.75rem', flexGrow: 1, backgroundColor: 'transparent' }}>  
 <Grid  
 container  
 direction="row"  
 spacing={2}  
 sx={{ alignItems: 'center', backgroundColor: 'transparent' }}  
 >  
 <Grid item size={1} sx={{ backgroundColor: 'transparent' }}>  
 <InputLabel sx={{ color: '#fff', backgroundColor: 'transparent' }}>  
 Floor  
 </InputLabel>  
 </Grid>  
 <Grid item size={1} sx={{ backgroundColor: 'transparent' }}>  
 <Autocomplete  
 options={FloorLevels}  
 getOptionLabel={(option) => option.label}  
 placeholder="Please Select"  
 disableClearable  
 sx={{ width: '100%' }}  
 />  
 </Grid>  
 </Grid>  
 </Box>  
 </GridToolbarContainer>  
 )  
}  
export default FloorLevelForm

## File: bd-scs-backend-web/src/components/Case/BuildingInformation/FloorLevel/index.jsx

import FloorLevelForm from "Form";  
function FloorLevel({}) {  
 return <>  
 <FloorLevelForm></FloorLevelForm>  
 </>  
}  
export default FloorLevel

## File: bd-scs-backend-web/src/components/Case/BuildingInformation/form.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled'  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import Grid from '@mui/material/Grid'  
import RadioGroup from '@mui/material/RadioGroup'  
import FormGroup from '@mui/material/FormGroup'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import Radio from '@mui/material/Radio'  
import Dialog from '@mui/material/Dialog'  
import DialogActions from '@mui/material/DialogActions'  
import DialogContent from '@mui/material/DialogContent'  
import DialogContentText from '@mui/material/DialogContentText'  
import DialogTitle from '@mui/material/DialogTitle'  
import Button from 'components/basic/Button'  
import Checkbox from 'components/basic/Checkbox'  
import RadioCheckbox from 'components/Case/RadioCheckbox'  
import Typography from 'components/basic/Typography'  
import TextField from 'components/basic/TextField'  
import GradientButton from 'components/basic/Button/GradientButton'  
import FormControl from 'components/basic/FormControl'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable'  
import SubmissionTable from 'components/Case/SubmissionTable'  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import CommonFormHeader from 'components/Case/FormHeader'  
import QuestionHeader from 'components/Case/QuestionHeader'  
import { Form } from 'react-router-dom'  
import colors from 'src/constants/colors'  
import { createTheme, alpha, getContrastRatio } from '@mui/material/styles'  
import { BuildingType } from "components/ApplicationForm/constants";  
function BuildingInformationForm({ control }) {  
 return (  
 <Stack direction="column">  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column">  
 <Box>  
 <Controller  
 name={'address'}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 title="Address"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Box>  
 <Box py={3}>  
 <Controller  
 name={'lotNo'}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 title="Lot No."  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Box>  
 {  
}  
 <Box py={3}>  
 <Controller  
 name={'fileReference'}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 title="File Reference"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Box>  
 <Box py={3}>  
 <Controller  
 name={'typeOfBuilding'}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 title="Type of Building"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {BuildingType.map((item) => (  
 <FormControlLabel  
 key={`${field.id}-${item.value}`}  
 control={  
 <Checkbox  
 checked={field.value === item.value}  
 onChange={(e) => {  
 const isChecked = e.target.checked  
 if (isChecked) {  
 field.onChange(item.value)  
 } else {  
 field.onChange('')  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 ))}  
 </FormGroup>  
 </FormControl>  
 )}  
 />  
 </Box>  
 {/\* <Box py={3}>  
 <Controller  
 name={'occupationPermit'}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 title="Occupation Permit (Link to Bravo)"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Box> \*/}  
 {}  
 <Box py={3}>  
 <Stack>  
 <Typography  
 variant="subtitle1"  
 sx={{ color: alpha('#000', 0.6) }}  
 >  
 Approved Plan Record  
 </Typography>  
 <Box sx={{ my: '0.5rem' }}></Box>  
 <Stack direction={'row'} spacing={1}>  
 <Button>Building Plan</Button>  
 <Button>Structural Plan</Button>  
 </Stack>  
 </Stack>  
 </Box>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default BuildingInformationForm

## File: bd-scs-backend-web/src/components/Case/BuildingInformation/index.jsx

import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useMutation, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import Stack from '@mui/material/Stack'  
import Dialog from '@mui/material/Dialog'  
import DialogActions from '@mui/material/DialogActions'  
import DialogContent from '@mui/material/DialogContent'  
import DialogContentText from '@mui/material/DialogContentText'  
import DialogTitle from '@mui/material/DialogTitle'  
import Button from 'components/basic/Button'  
import Form from 'components/Case/BuildingInformation/form'  
import FloorLevelForm from './FloorLevel/Form'  
import IconButton from '@mui/material/IconButton'  
import CloseIcon from '@mui/icons-material/Close'  
import UnderlineTabs from 'components/basic/UnderTabs'  
import TabContext from '@mui/lab/TabContext'  
import TabList from '@mui/lab/TabList'  
import TabPanel from '@mui/lab/TabPanel'  
import Tab from '@mui/material/Tab'  
import colors from 'src/constants/colors'  
import { updateCase } from 'src/apis/case'  
import { DVTableTaskPage } from 'routes/case/DvTable'  
function BuildingInformationDialog({ open, onClose, data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const [tab, setTab] = React.useState(0)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 values: {  
 address: data?.address,  
 lotNo: data?.lotNo,  
 siteID: data?.siteID,  
 notionalSiteID: data?.notionalSiteID,  
 blockID: data?.blockID,  
 typeOfBuilding: data?.typeOfBuilding,  
 occupationPermit: data?.occupationPermit,  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 onClose && onClose()  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const payload = {  
 building\_information: formData,  
 }  
 mutation.mutate(payload)  
 }  
 return (  
 <TabContext value={tab}>  
 <Dialog  
 open={open}  
 onClose={onClose}  
 aria-labelledby="alert-dialog-title"  
 aria-describedby="alert-dialog-description"  
 fullWidth  
 maxWidth="xxxl"  
 >  
 <Stack  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 >  
 <DialogTitle id="alert-dialog-title">  
 Building Information  
 </DialogTitle>  
 <IconButton  
 aria-label="close"  
 onClick={onClose}  
 sx={(theme) => ({  
 position: 'absolute',  
 right: 20,  
 top: 8,  
 color: theme.palette.grey[500],  
 })}  
 >  
 <CloseIcon />  
 </IconButton>  
 </Stack>  
 {  
}  
 <TabList  
 onChange={(event, newValue) => {  
 setTab(newValue)  
 }}  
 aria-label=""  
 >  
 <Tab label="Building Level" value={0} />  
 <Tab label="Floor Level" value={1} />  
 <Tab label="Room Use and DV" value={2} />  
 </TabList>  
 <DialogContent sx={{ p: 0 }}>  
 <TabPanel value={0}>  
 <Form control={control} setValue={setValue} />  
 </TabPanel>  
 <TabPanel value={1} sx={{ p: 0 }}>  
 <FloorLevelForm  
 control={control}  
 setValue={setValue}  
 ></FloorLevelForm>  
 </TabPanel>  
 <TabPanel value={2}>  
 <DVTableTaskPage />  
 </TabPanel>  
 </DialogContent>  
 <DialogActions>  
 <Button onClick={onClose}>Cancel</Button>  
 <Button onClick={() => handleSubmit(onSubmit)()} autoFocus>  
 Submit  
 </Button>  
 </DialogActions>  
 </Dialog>  
 </TabContext>  
 )  
}  
export default BuildingInformationDialog

## File: bd-scs-backend-web/src/components/Case/DV/AdequacyOfExitsFromStoreysTable.jsx

import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import styled from '@emotion/styled'  
import Box from '@mui/material/Box'  
import Paper from '@mui/material/Paper'  
import Table from '@mui/material/Table'  
import TableBody from '@mui/material/TableBody'  
import TableCell from '@mui/material/TableCell'  
import TableContainer from '@mui/material/TableContainer'  
import TableHead from '@mui/material/TableHead'  
import TableRow from '@mui/material/TableRow'  
import Button from 'components/basic/Button'  
import Typography from 'components/basic/Typography'  
import Checkbox from 'components/basic/Checkbox'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import { defAESValue } from 'src/routes/case/DvTable'  
const StyledTableCell = styled(TableCell)`  
 && {  
 max-width: ${props => props.w ? `${props.w}px` : `100px`};  
 width: ${props => props.w ? `${props.w}px` : `100px`};  
 min-width: ${props => props.w ? `${props.w}px` : `100px`};  
 }  
`  
function TableA({ control, name, setValue }) {  
 const { fields, append, prepend, remove, swap, move, insert } = useFieldArray({  
 control,  
 name: name,  
 });  
 const watchFields = useWatch({ control, name: name })  
 return (  
 <Box sx={{ width: '100%' }}>  
 <TableContainer component={Paper}>  
 <Table sx={{ minWidth: 650 }} aria-label="simple table">  
 <TableHead>  
 <TableRow>  
 <StyledTableCell>Floor</StyledTableCell>  
 <StyledTableCell>Compartment/ Total Capacity of the Storey</StyledTableCell>  
 <StyledTableCell>Capacity</StyledTableCell>  
 <StyledTableCell w={200} align="center">  
 Minimum No. of Exit Door/Route  
 <TableRow>  
 <StyledTableCell w={80} sx={{ borderBottom: "none" }}>Required</StyledTableCell>  
 <StyledTableCell w={80} sx={{ borderBottom: "none" }}>Provided</StyledTableCell>  
 </TableRow>  
 </StyledTableCell>  
 <StyledTableCell w={200} align="center">  
 Minimum Width of Exit Door (Total)  
 <TableRow>  
 <StyledTableCell w={80} sx={{ borderBottom: "none" }}>Required</StyledTableCell>  
 <StyledTableCell w={80} sx={{ borderBottom: "none" }}>Provided</StyledTableCell>  
 </TableRow>  
 </StyledTableCell>  
 <StyledTableCell w={200} align="center">  
 Minimum Width of Exit Route (Total)  
 <TableRow>  
 <StyledTableCell w={80} sx={{ borderBottom: "none" }}>Required</StyledTableCell>  
 <StyledTableCell w={80} sx={{ borderBottom: "none" }}>Provided</StyledTableCell>  
 </TableRow>  
 </StyledTableCell>  
 <StyledTableCell w={200} align="center">  
 Minimum Width of Exit Door (Each)  
 <TableRow>  
 <StyledTableCell w={80} sx={{ borderBottom: "none" }}>Required</StyledTableCell>  
 <StyledTableCell w={80} sx={{ borderBottom: "none" }}>Provided</StyledTableCell>  
 </TableRow>  
 </StyledTableCell>  
 <StyledTableCell w={200} align="center">  
 Minimum Width of Exit Route (Each)  
 <TableRow>  
 <StyledTableCell w={80} sx={{ borderBottom: "none" }}>Required</StyledTableCell>  
 <StyledTableCell w={80} sx={{ borderBottom: "none" }}>Provided</StyledTableCell>  
 </TableRow>  
 </StyledTableCell>  
 <StyledTableCell align="center">Checking Result</StyledTableCell>  
 <StyledTableCell>Action</StyledTableCell>  
 </TableRow>  
 </TableHead>  
 <TableBody>  
 {  
 fields.map((item, index) => {  
 return (  
 <TableRow>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.floor`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.compartment`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.capacity`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <StyledTableCell w={200}>  
 <TableRow>  
 <StyledTableCell w={80} sx={{ borderBottom: 0 }}>  
 <Controller  
 name={`${name}.${index}.minimumNoOfExitDoorRouteRequired`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </StyledTableCell>  
 <StyledTableCell w={80} sx={{ borderBottom: 0 }}>  
 <Controller  
 name={`${name}.${index}.minimumNoOfExitDoorRouteProvided`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </StyledTableCell>  
 </TableRow>  
 </StyledTableCell>  
 <StyledTableCell w={200}>  
 <TableRow>  
 <StyledTableCell w={80} sx={{ borderBottom: 0 }}>  
 <Controller  
 name={`${name}.${index}.minimumWidthOfExitDoorTotalRequired`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </StyledTableCell>  
 <StyledTableCell w={80} sx={{ borderBottom: 0 }}>  
 <Controller  
 name={`${name}.${index}.minimumWidthOfExitDoorTotalProvided`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </StyledTableCell>  
 </TableRow>  
 </StyledTableCell>  
 <StyledTableCell w={200}>  
 <TableRow>  
 <StyledTableCell w={80} sx={{ borderBottom: 0 }}>  
 <Controller  
 name={`${name}.${index}.minimumWidthOfExiRouteTotalRequired`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </StyledTableCell>  
 <StyledTableCell w={80} sx={{ borderBottom: 0 }}>  
 <Controller  
 name={`${name}.${index}.minimumWidthOfExiRouteTotalProvided`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </StyledTableCell>  
 </TableRow>  
 </StyledTableCell>  
 <StyledTableCell w={200}>  
 <TableRow>  
 <StyledTableCell w={80} sx={{ borderBottom: 0 }}>  
 <Controller  
 name={`${name}.${index}.minimumWidthOfExitDoorEachRequired`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </StyledTableCell>  
 <StyledTableCell w={80} sx={{ borderBottom: 0 }}>  
 <Controller  
 name={`${name}.${index}.minimumWidthOfExitDoorEachProvided`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </StyledTableCell>  
 </TableRow>  
 </StyledTableCell>  
 <StyledTableCell w={200}>  
 <TableRow>  
 <StyledTableCell w={80} sx={{ borderBottom: 0 }}>  
 <Controller  
 name={`${name}.${index}.minimumWidthOfExiRouteEachRequired`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </StyledTableCell>  
 <StyledTableCell w={80} sx={{ borderBottom: 0 }}>  
 <Controller  
 name={`${name}.${index}.minimumWidthOfExiRouteEachProvided`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </StyledTableCell>  
 </TableRow>  
 </StyledTableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.checkingResult`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Checkbox  
 checked={field?.value === 'yes'}  
 onChange={(e) => {  
 setValue(`${name}.${index}.checkingResult`, e.target.checked ? 'yes' : 'no')  
 }}  
 disableRipple  
 sx={{ padding: 0 }}  
 />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell sx={{ cursor: 'pointer' }} onClick={() => remove(index)}>  
 <Typography variant="warning">Delete</Typography>  
 </TableCell>  
 </TableRow>  
 )  
 })  
 }  
 </TableBody>  
 </Table>  
 </TableContainer>  
 <Button sx={{ m: 2 }} onClick={() => append(defAESValue)}>Add Row</Button>  
 </Box>  
 )  
}  
export default TableA

## File: bd-scs-backend-web/src/components/Case/DV/AdequacyOfStaircasesTable.jsx

import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import styled from '@emotion/styled'  
import { useEffect, useMemo, useState } from 'react'  
import Box from '@mui/material/Box'  
import Paper from '@mui/material/Paper'  
import Table from '@mui/material/Table'  
import TableBody from '@mui/material/TableBody'  
import TableCell from '@mui/material/TableCell'  
import TableContainer from '@mui/material/TableContainer'  
import TableHead from '@mui/material/TableHead'  
import TableRow from '@mui/material/TableRow'  
import Button from 'components/basic/Button'  
import Typography from 'components/basic/Typography'  
import Checkbox from 'components/basic/Checkbox'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import { defAOSValue } from 'src/routes/case/DvTable'  
const StyledTableCell = styled(TableCell)`  
 && {  
 max-width: ${(props) => (props.w ? `${props.w}px` : `100px`)};  
 width: ${(props) => (props.w ? `${props.w}px` : `100px`)};  
 min-width: ${(props) => (props.w ? `${props.w}px` : `100px`)};  
 }  
`  
const getTotalCount = (array, fieldName) => {  
 return array.reduce((acc, curr) => {  
 return Number(acc || 0) + Number(curr?.[fieldName] || 0)  
 }, 0)  
}  
function TableA({ control, name, setValue, data }) {  
 const watchFields = useWatch({ control, name: name })  
 const [cols, setCols] = useState([])  
 const { fields, append, prepend, remove, swap, move, insert } =  
 useFieldArray({  
 control,  
 name: name,  
 })  
 useEffect(() => {  
 if (data?.[0]) {  
 let tmpRow = { ...data?.[0] }  
 delete tmpRow.id  
 setCols(Object.keys(tmpRow))  
 }  
 if (fields?.[0]) {  
 let tmpRow = { ...fields?.[0] }  
 delete tmpRow.id  
 setCols(Object.keys(tmpRow))  
 } else {  
 setCols(Object.keys(defAOSValue))  
 }  
 }, [data])  
 const getColumns = useMemo(() => {  
 const tmp = [...cols]  
 return tmp.map((key) => {  
 const name = key.includes('ST')  
 ? key.toUpperCase()  
 : key  
 .replace(/([A-Z])/g, ' $1')  
 .replace(/^./, (str) => str.toUpperCase())  
 return {  
 id: key,  
 name: name,  
 }  
 })  
 }, [cols])  
 const row = useMemo(() => {  
 const tmp = [...cols]  
 const res = tmp.reduce((acc, curr) => ((acc[curr] = ''), acc), {})  
 return res  
 }, [cols])  
 const onClickAddColumn = () => {  
 const stColumnsCount = cols.filter((key) => key.includes('ST')).length  
 const newCols = [...cols, `ST${stColumnsCount + 1}`]  
 let tmp = [...watchFields]  
 tmp.forEach((item, index) => {  
 tmp[index] = { ...item, [newCols[newCols.length - 1]]: '' }  
 })  
 setValue(name, tmp)  
 setCols(newCols)  
 }  
 const onClickDeleteColumn = () => {  
 const stColumnsCount = cols.filter((key) => key.includes('ST')).length  
 if (stColumnsCount > 0) {  
 const newCols = cols.slice(0, cols.length - 1)  
 let tmp = [...watchFields]  
 tmp.forEach((item, index) => {  
 delete item[newCols[stColumnsCount.length - 1]]  
 })  
 setValue(name, tmp)  
 setCols(newCols)  
 }  
 }  
 return (  
 <Box sx={{ width: '100%' }}>  
 <TableContainer component={Paper}>  
 <Table sx={{ minWidth: 650 }} aria-label="simple table">  
 <TableHead>  
 <TableRow>  
 {getColumns.map((item, index) => {  
 return (  
 <StyledTableCell>  
 {item?.name}  
 </StyledTableCell>  
 )  
 })}  
 <StyledTableCell>Action</StyledTableCell>  
 </TableRow>  
 </TableHead>  
 <TableBody>  
 {fields.map((item, index) => {  
 return (  
 <TableRow key={item.id}>  
 {getColumns.map((item, colIndex) => {  
 return (  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.${item.id}`}  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type={  
 item.id ===  
 'floor'  
 ? 'text'  
 : 'number'  
 }  
 />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 )  
 })}  
 <TableCell  
 sx={{ cursor: 'pointer' }}  
 onClick={() => remove(index)}  
 >  
 <Typography variant="warning">  
 Delete  
 </Typography>  
 </TableCell>  
 </TableRow>  
 )  
 })}  
 <TableRow>  
 <TableCell  
 align="right"  
 sx={{ borderBottom: 'none' }}  
 >  
 Total occupant capacity  
 </TableCell>  
 {getColumns.map((item, index) => {  
 if (item.id === 'floor') return null  
 // return (  
 // <TableCell sx={{ borderBottom: "none" }}>{getTotalCount(watchFields, item.id)}</TableCell>  
 // )  
 const fieldName = item.id.includes('ST')  
 ? `staircasePopulation.${item.id}`  
 : item.id  
 return (  
 <TableCell sx={{ borderBottom: 'none' }}>  
 <Controller  
 name={`adequacyOfStaircasesResult.totalOccupantCapacity.${fieldName}`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="number"  
 />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 )  
 })}  
 </TableRow>  
 <TableRow>  
 <TableCell  
 align="right"  
 sx={{ borderBottom: 'none' }}  
 >  
 Total Discharge Value  
 </TableCell>  
 <TableCell sx={{ borderBottom: 'none' }} />  
 <TableCell sx={{ borderBottom: 'none' }}>  
 <Controller  
 name={`adequacyOfStaircasesResult.totalDischargeValue.totalAcceptedPopulation`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="number"  
 />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 {getColumns.map((item, index) => {  
 if (!item?.id?.includes('ST')) return null  
 return (  
 <TableCell sx={{ borderBottom: 'none' }}>  
 <Controller  
 name={`adequacyOfStaircasesResult.totalDischargeValue.staircasePopulation.${item.id}`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="number"  
 />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 )  
 })}  
 </TableRow>  
 <TableRow>  
 <TableCell  
 colSpan={3}  
 align="right"  
 sx={{ borderBottom: 'none' }}  
 >  
 Checking Result  
 </TableCell>  
 {getColumns.map((item, index) => {  
 if (!item?.id?.includes('ST')) return null  
 return (  
 <TableCell sx={{ borderBottom: 'none' }}>  
 <Controller  
 name={`adequacyOfStaircasesResult.checkingResult.${item.id}`}  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Checkbox  
 {...field}  
 disableRipple  
 checked={  
 field.value ===  
 'yes'  
 }  
 onChange={(e) =>  
 field.onChange(  
 e.target.checked  
 ? 'yes'  
 : ''  
 )  
 }  
 />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 )  
 })}  
 </TableRow>  
 </TableBody>  
 </Table>  
 </TableContainer>  
 <Button sx={{ m: 2 }} onClick={onClickAddColumn}>  
 Add Column  
 </Button>  
 <Button sx={{ m: 2 }} onClick={onClickDeleteColumn}>  
 Delete Column  
 </Button>  
 <Button  
 sx={{ m: 2 }}  
 onClick={() => setValue(name, [...watchFields, row])}  
 >  
 Add Row  
 </Button>  
 </Box>  
 )  
}  
export default TableA

## File: bd-scs-backend-web/src/components/Case/DV/OccupantCapacityOfRoomsTable.jsx

import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import styled from '@emotion/styled'  
import Box from '@mui/material/Box'  
import Paper from '@mui/material/Paper'  
import Table from '@mui/material/Table'  
import TableBody from '@mui/material/TableBody'  
import TableCell from '@mui/material/TableCell'  
import TableContainer from '@mui/material/TableContainer'  
import TableHead from '@mui/material/TableHead'  
import TableRow from '@mui/material/TableRow'  
import Button from 'components/basic/Button'  
import Typography from 'components/basic/Typography'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import { defOCRValue } from 'src/routes/case/DvTable'  
const StyledTableCell = styled(TableCell)`  
 && {  
 max-width: 100px;  
 width: 100px;  
 min-width: 100px;  
 }  
`  
const getTotalCount = (array, fieldName) => {  
 return array.reduce((acc, curr) => {  
 return Number(acc || 0) + Number(curr?.[fieldName] || 0)  
 }, 0)  
}  
function TableA({ control, name, setValue }) {  
 const { fields, append, prepend, remove, swap, move, insert } = useFieldArray({  
 control,  
 name: name,  
 });  
 const watchFields = useWatch({ control, name: name })  
 const exitArr = Array.from(Array(10).keys())  
 return (  
 <Box sx={{ width: '100%' }}>  
 <TableContainer component={Paper}>  
 <Table sx={{ minWidth: 650 }} aria-label="simple table">  
 <TableHead>  
 <TableRow>  
 <StyledTableCell>Floor</StyledTableCell>  
 <StyledTableCell>Compartment</StyledTableCell>  
 <StyledTableCell>Room Name</StyledTableCell>  
 <StyledTableCell>Room Use</StyledTableCell>  
 <StyledTableCell>Area ID</StyledTableCell>  
 <StyledTableCell>UFA</StyledTableCell>  
 <StyledTableCell>FS Factor</StyledTableCell>  
 <StyledTableCell>Approved Capacity</StyledTableCell>  
 <StyledTableCell>Approved Capacity/ Previous Accepted Capacity</StyledTableCell>  
 <StyledTableCell>New Occupant Capacity</StyledTableCell>  
 <StyledTableCell>Revised Occupant Capacity (By BS)</StyledTableCell>  
 <StyledTableCell>Change in Population</StyledTableCell>  
 <StyledTableCell>No. of exit</StyledTableCell>  
 {  
 exitArr.map((exitIndex) => {  
 return (  
 <StyledTableCell>Exit {exitIndex + 1} Capacity</StyledTableCell>  
 )  
 })  
 }  
 <StyledTableCell>Remark</StyledTableCell>  
 <StyledTableCell>Action</StyledTableCell>  
 </TableRow>  
 </TableHead>  
 <TableBody>  
 {  
 fields.map((item, index) => {  
 return (  
 <TableRow>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.floor`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.compartment`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.roomName`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.roomUse`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.areaId`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.ufa`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} type="number" />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.FSFactor`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} type="number" />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.approvedCapacity`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} type="number" />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.approvedCapacity\_previousApprovedCapacity`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="number"  
 onChange={(e) => {  
 const a = e.target.value ? Number(e.target.value) : 0  
 const n = watchFields[index]?.newOccupantCapacity || 0  
 const t = n - a  
 setValue(`${name}.${index}.changeInPopulation`, 0 > t ? "" : t)  
 field.onChange(e.target.value)  
 }}  
 />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.newOccupantCapacity`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="number"  
 onChange={(e) => {  
 const n = e.target.value ? Number(e.target.value) : 0  
 const a = watchFields[index]?.approvedCapacity\_previousApprovedCapacity || 0  
 const t = n - a  
 setValue(`${name}.${index}.changeInPopulation`, 0 > t ? "" : t)  
 field.onChange(e.target.value)  
 }}  
 />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.revisedOccupantCapacity`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.changeInPopulation`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled type="number" />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.noOfExit`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="number"  
 helperText="max: 10"  
 onChange={(e) => {  
 const a = e.target.value ? Number(e.target.value) : 0  
 if (a > 10 || a < 0) {  
 return  
 }  
 field.onChange(e.target.value)  
 }}  
 />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 {  
 exitArr.map((exitIndex) => {  
 return (  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.exitCapacity${exitIndex+1}`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} type="number" />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 )  
 })  
 }  
 <TableCell>  
 <Controller  
 name={`${name}.${index}.remark`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 <TableCell sx={{ cursor: 'pointer' }} onClick={() => remove(index)}>  
 <Typography variant="warning">Delete</Typography>  
 </TableCell>  
 </TableRow>  
 )  
 })  
 }  
 <TableRow>  
 <TableCell rowSpan={2} sx={{ borderBottom: "none" }} />  
 <TableCell colSpan={6} align="right" sx={{ borderBottom: "none" }}>Total Capacity</TableCell>  
 <TableCell>{getTotalCount(watchFields, 'approvedCapacity')}</TableCell>  
 <TableCell>{getTotalCount(watchFields, 'approvedCapacity\_previousApprovedCapacity')}</TableCell>  
 </TableRow>  
 <TableRow>  
 <TableCell colSpan={6} align="right" sx={{ borderBottom: "none" }}>Total Capacity of storey :</TableCell>  
 <TableCell>{getTotalCount(watchFields, 'approvedCapacity\_previousApprovedCapacity') + getTotalCount(watchFields, 'changeInPopulation')}</TableCell>  
 </TableRow>  
 </TableBody>  
 </Table>  
 </TableContainer>  
 <Button sx={{ m: 2 }} onClick={() => append(defOCRValue)}>Add Row</Button>  
 </Box>  
 )  
}  
export default TableA

## File: bd-scs-backend-web/src/components/Case/P3/index.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled'  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import Grid from '@mui/material/Grid'  
import RadioGroup from '@mui/material/RadioGroup'  
import FormGroup from '@mui/material/FormGroup'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import InputAdornment from '@mui/material/InputAdornment'  
import Radio from '@mui/material/Radio'  
import Checkbox from 'components/basic/Checkbox'  
import Typography from 'components/basic/Typography'  
import TextField from 'components/basic/TextField'  
import GradientButton from 'components/basic/Button/GradientButton'  
import FormControl from 'components/basic/FormControl'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable'  
import SubmissionTable from 'components/Case/SubmissionTable'  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import CommonFormHeader from 'components/Case/FormHeader'  
import QuestionHeader from 'components/Case/QuestionHeader'  
import Q1QuestionBody from 'components/Case/P3/Q1'  
import Q4QuestionBody from 'components/Case/P3/Q4'  
import CAT\_DESCRIPTIONS from 'src/config/cat';  
import FormControlLabelEditable from 'components/FormControls/FormControlLabelEditable'  
function P3Task({ control, onSubmit, setValue, getValues, nature }) {  
 const q1Ans = useWatch({ control, name: 'q1.answer' })  
 const q1Kpa = useWatch({ control, name: 'q1.kpa' })  
 const q3Ans = useWatch({ control, name: 'q3.answer' })  
 let optionsCat22 = {  
 a: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q3\_cat2\_2a.a,  
 b: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q3\_cat2\_2a.b,  
 }  
 let optionsCat1 = {  
 a: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q5\_options\_cat1.a,  
 b: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q5\_options\_cat1.b,  
 c: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q5\_options\_cat1.c,  
 }  
 let optionsQ6Cat1 = {  
 a: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q6\_options\_cat1.a,  
 }  
 if(getValues('q3.options\_cat2\_2a\_text')) {  
 Object.keys(getValues('q3.options\_cat2\_2a\_text')).forEach((key) => {  
 optionsCat22[key] = getValues('q3.options\_cat2\_2a\_text')[key];  
 });  
 }  
 if(getValues('q5.options\_cat1\_text')) {  
 Object.keys(getValues('q5.options\_cat1\_text')).forEach((key) => {  
 optionsCat1[key] = getValues('q5.options\_cat1\_text')[key];  
 });  
 }  
 if(getValues('q6.options\_cat1\_text')) {  
 Object.keys(getValues('q6.options\_cat1\_text')).forEach((key) => {  
 optionsQ6Cat1[key] = getValues('q6.options\_cat1\_text')[key];  
 });  
 }  
 return (  
 <Stack direction="column">  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Typography variant="title2" mx={3}>  
 {`Structural (Filled in by BS)`}  
 </Typography>  
 <Box px={4} py={3}>  
 <Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`1. Design live load adequate for the intended use?`}</Typography>  
 </Grid>  
 <Grid item>  
 <Controller  
 name="q1.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(  
 e.target.value  
 )  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={  
 <Radio disabled />  
 }  
 label={`On grade`}  
 />  
 <FormControlLabel  
 value={false}  
 control={  
 <Radio disabled />  
 }  
 label={`On suspended slab`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 {q1Ans === 'false' && (  
 <Box  
 sx={{  
 backgroundColor: '#fff',  
 p: 2,  
 mb: 2,  
 }}  
 >  
 <Controller  
 name="q1.kpa"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 InputProps={{  
 endAdornment: (  
 <InputAdornment position="end">  
 kpa  
 </InputAdornment>  
 ),  
 }}  
 disabled  
 {...field}  
 />  
 )}  
 />  
 </Box>  
 )}  
 {nature === 'SCH' &&  
 q1Ans === 'false' &&  
 q1Kpa < 3 && (  
 <Q1QuestionBody control={control} getValues={getValues} />  
 )}  
 {nature === 'KIND' &&  
 q1Ans === 'false' &&  
 q1Kpa < 2.5 && (  
 <Q1QuestionBody control={control} getValues={getValues} />  
 )}  
 </Box>  
 <Box px={4} py={3}>  
 <Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`2. No structural timber floors`}</Typography>  
 </Grid>  
 <Grid item>  
 <Controller  
 name="q2.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(  
 e.target.value  
 )  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`Issue Cert 2`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`Not Issue Cert 2`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`3. Significant structure/installation requiring Structural Justification (SJ)`}  
 control={control}  
 needAnswer  
 ansFieldName="q3.answer"  
 onChange={() => {  
 setValue('q3.options\_cat2\_2a', [])  
 setValue('q3.other\_cat2\_2a', '')  
 }}  
 />  
 {q3Ans == true && (  
 <Box sx={{ backgroundColor: '#fff', p: 2 }}>  
 <Typography>  
 {`(Logical question, appear if select "Yes")`}  
 </Typography>  
 <Typography  
 mt={2}  
 >{`[Cat. 2(ii)]`}</Typography>  
 <Typography>{`The AP/RSE is required to check and confirm the structural adequacy of the existing floor in respect of the additional loads due to the following :`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q3.options\_cat2\_2a"  
 control={control}  
 defaultValue={[]}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup sx={{ mt: 2 }}>  
 {['a', 'b'].map((key) => (  
 <FormControlLabelEditable  
 key={key}  
 isChecked={field.value?.includes(key)}  
 onChange={(isChecked) => {  
 if (isChecked) {  
 field.onChange([...field.value, key]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== key));  
 }  
 }}  
 label={optionsCat22[key]}  
 isLabelHighlight={optionsCat22[key] !== CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q3\_cat2\_2a[key]}  
 sx={{ mb: 1 }}  
 />  
 ))}  
 <Grid  
 container  
 direction="row"  
 >  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q3.other\_cat2\_2a"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 )}  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`4. Any minor works submission required (e.g. for structural defects / new MW structures)`}  
 control={control}  
 needAnswer  
 ansFieldName="q4.answer"  
 onChange={() => {  
 setValue('q4.options\_cat1\_1', [])  
 setValue('q4.options\_cat1\_2', [])  
 setValue('q4.options\_cat1\_3', [])  
 setValue('q4.options\_cat1\_4', [])  
 setValue('q4.other\_cat1\_4', '')  
 }}  
 />  
 <Q4QuestionBody  
 control={control}  
 setValue={setValue}  
 getValues={getValues}  
 />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`5. Other general conditions`}  
 control={control}  
 />  
 <Box sx={{ backgroundColor: '#fff', p: 2 }}>  
 <Typography mt={2}>{`[Cat. 1]`}</Typography>  
 <Controller  
 name="q5.options\_cat1"  
 control={control}  
 defaultValue={[]}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup>  
 {['a', 'b', 'c'].map((key) => (  
 <FormControlLabelEditable  
 key={key}  
 isChecked={field.value?.includes(key)}  
 onChange={(isChecked) => {  
 if (isChecked) {  
 field.onChange([...field.value, key]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== key));  
 }  
 }}  
 label={optionsCat1[key]}  
 isLabelHighlight={optionsCat1[key] !== CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q5\_options\_cat1[key]}  
 sx={{ mb: 1 }}  
 />  
 ))}  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(e) => {  
 const isChecked =  
 e.target  
 .checked  
 if (isChecked) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q5.other\_cat1"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`6. Additional comment from SE`}  
 control={control}  
 />  
 <Box sx={{ backgroundColor: '#fff', p: 2 }}>  
 <Typography mt={2}>{`[Cat. 1]`}</Typography>  
 <Box ml={4}>  
 <Controller  
 name="q6.options\_cat1"  
 control={control}  
 defaultValue={[]}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup>  
 {['a'].map((key) => (  
 <FormControlLabelEditable  
 key={key}  
 isChecked={field.value?.includes(key)}  
 onChange={(isChecked) => {  
 if (isChecked) {  
 field.onChange([...field.value, key]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== key));  
 }  
 }}  
 label={optionsQ6Cat1[key]}  
 isLabelHighlight={optionsQ6Cat1[key] !== CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q6\_options\_cat1[key]}  
 sx={{ mb: 1 }}  
 />  
 ))}  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e.target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q6.other\_cat1"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 </Box>  
 <GradientButton  
 sx={{ width: 100, alignSelf: 'flex-end', mr: 2 }}  
 onClick={onSubmit}  
 >  
 Save  
 </GradientButton>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default P3Task

## File: bd-scs-backend-web/src/components/Case/P3/Q1.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
import CAT\_DESCRIPTIONS from 'src/config/cat';  
function QuestionBody({ control, getValues }) {  
 let optionsCat3a = CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q1\_cat3.a;  
 if(getValues('q1.options\_cat3\_text')) {  
 Object.keys(getValues('q1.options\_cat3\_text')).forEach((key) => {  
 if(key === 'a') optionsCat3a = getValues('q1.options\_cat3\_text')[key];  
 });  
 }  
 return (  
 <>  
 <Typography>{`[Cat. 3]`}</Typography>  
 <Typography>{`The AP/RSE is required to demonstrate compliance of the following to the satisfaction of the BD`}</Typography>  
 <Controller  
 name="q1.options\_cat3"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 2 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={  
 <span style={{backgroundColor: (optionsCat3a === CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q1\_cat3.a ? '' : 'yellow')}}>  
 {optionsCat3a}  
 </span>  
 }  
 sx={{ mb: 2 }}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.other\_cat3"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3/Q4.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
import CAT\_DESCRIPTIONS from 'src/config/cat';  
import FormControlLabelEditable from 'components/FormControls/FormControlLabelEditable';  
import { options } from '../P6/Q1';  
function QuestionBody({ control, setValue, getValues }) {  
 const q4Ans = useWatch({ control, name: 'q4.answer' });  
 if (!q4Ans || q4Ans === "false") return null;  
 let optionsCat11 = {  
 a: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_1.a,  
 b: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_1.b,  
 c: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_1.c,  
 d: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_1.d,  
 e: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_1.e,  
 f: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_1.f  
 }  
 let optionsCat12 = {  
 a: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_2.a,  
 b: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_2.b,  
 c: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_2.c  
 }  
 let optionsCat13 = {  
 a: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_3.a,  
 b: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_3.b,  
 c: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_3.c,  
 d: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_3.d,  
 e: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_3.e,  
 f: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_3.f  
 }  
 let optionsCat14 = {  
 a: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_4.a,  
 b: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_4.b,  
 c: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_4.c,  
 d: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_4.d,  
 e: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_4.e,  
 f: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_4.f,  
 g: CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_4.g  
 }  
 if(getValues('q4.options\_cat1\_1\_text')) {  
 Object.keys(getValues('q4.options\_cat1\_1\_text')).forEach((key) => {  
 optionsCat11[key] = getValues('q4.options\_cat1\_1\_text')[key];  
 });  
 }  
 if(getValues('q4.options\_cat1\_2\_text')) {  
 Object.keys(getValues('q4.options\_cat1\_2\_text')).forEach((key) => {  
 optionsCat12[key] = getValues('q4.options\_cat1\_2\_text')[key];  
 });  
 }  
 if(getValues('q4.options\_cat1\_3\_text')) {  
 Object.keys(getValues('q4.options\_cat1\_3\_text')).forEach((key) => {  
 optionsCat13[key] = getValues('q4.options\_cat1\_3\_text')[key];  
 });  
 }  
 if(getValues('q4.options\_cat1\_4\_text')) {  
 Object.keys(getValues('q4.options\_cat1\_4\_text')).forEach((key) => {  
 optionsCat14[key] = getValues('q4.options\_cat1\_4\_text')[key];  
 });  
 }  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`(Logical question, appear if select "Yes")`}</Typography>  
 <Typography mt={2}>{`[Cat. 1]`}</Typography>  
 <Typography>{`The following MW are subjected to the control of the MWCS:`}</Typography>  
 <Stack gap={2} mt={2}>  
 <Box display="flex" direction="row" alignItems="center">  
 <Checkbox  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 setValue(`q4.options\_cat1\_1`, ['a', 'b', 'c', 'd', 'e', 'f']);  
 } else {  
 setValue(`q4.options\_cat1\_1`, []);  
 }  
 }}  
 />  
 <Typography variant="title2" sx={{ fontSize: 18 }}>{`Common Minor Works`}</Typography>  
 </Box>  
 <Box ml={4}>  
 <Controller  
 name="q4.options\_cat1\_1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup>  
 {['a', 'b', 'c', 'd', 'e', 'f'].map((key) => (  
 <FormControlLabelEditable  
 key={key}  
 isChecked={field.value?.includes(key)}  
 onChange={(isChecked) => {  
 if (isChecked) {  
 field.onChange([...field.value, key]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== key));  
 }  
 }}  
 label={optionsCat11[key]}  
 isLabelHighlight={optionsCat11[key] !== CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_1[key]}  
 sx={{ mb: 1 }}  
 />  
 ))}  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Stack>  
 <Stack gap={2} mt={2}>  
 <Box display="flex" direction="row" alignItems="center">  
 <Checkbox  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 setValue(`q4.options\_cat1\_2`, ['a', 'b', 'c']);  
 } else {  
 setValue(`q4.options\_cat1\_2`, []);  
 }  
 }}  
 />  
 <Typography variant="title2" sx={{ fontSize: 18 }}>{`Structural Alteration or Removal works`}</Typography>  
 </Box>  
 <Box ml={4}>  
 <Controller  
 name="q4.options\_cat1\_2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup>  
 {['a', 'b', 'c'].map((key) => (  
 <FormControlLabelEditable  
 key={key}  
 isChecked={field.value?.includes(key)}  
 onChange={(isChecked) => {  
 if (isChecked) {  
 field.onChange([...field.value, key]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== key));  
 }  
 }}  
 label={optionsCat12[key]}  
 isLabelHighlight={optionsCat12[key] !== CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_2[key]}  
 sx={{ mb: 1 }}  
 />  
 ))}  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Stack>  
 <Stack gap={2} mt={2}>  
 <Box display="flex" direction="row" alignItems="center">  
 <Checkbox  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 setValue(`q4.options\_cat1\_3`, ['a', 'b', 'c', 'd', 'e', 'f']);  
 } else {  
 setValue(`q4.options\_cat1\_3`, []);  
 }  
 }}  
 />  
 <Typography variant="title2" sx={{ fontSize: 18 }}>{`Roof or External Works`}</Typography>  
 </Box>  
 <Box ml={4}>  
 <Controller  
 name="q4.options\_cat1\_3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup>  
 {['a', 'b', 'c', 'd', 'e', 'f'].map((key) => (  
 <FormControlLabelEditable  
 key={key}  
 isChecked={field.value?.includes(key)}  
 onChange={(isChecked) => {  
 if (isChecked) {  
 field.onChange([...field.value, key]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== key));  
 }  
 }}  
 label={optionsCat13[key]}  
 isLabelHighlight={optionsCat13[key] !== CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_3[key]}  
 sx={{ mb: 1 }}  
 />  
 ))}  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Stack>  
 <Stack gap={2} mt={2}>  
 <Box display="flex" direction="row" alignItems="center">  
 <Checkbox  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 setValue(`q4.options\_cat1\_4`, ['a', 'b', 'c', 'd', 'e', 'f', 'other']);  
 } else {  
 setValue(`q4.options\_cat1\_4`, []);  
 }  
 }}  
 />  
 <Typography variant="title2" sx={{ fontSize: 18 }}>{`Subdivided Flat or Interior Works`}</Typography>  
 </Box>  
 <Box ml={4}>  
 <Controller  
 name="q4.options\_cat1\_4"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup>  
 {['a', 'b', 'c', 'd', 'e', 'f'].map((key) => (  
 <FormControlLabelEditable  
 key={key}  
 isChecked={field.value?.includes(key)}  
 onChange={(isChecked) => {  
 if (isChecked) {  
 field.onChange([...field.value, key]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== key));  
 }  
 }}  
 label={optionsCat14[key]}  
 isLabelHighlight={optionsCat14[key] !== CAT\_DESCRIPTIONS.structural\_schnlhkinds\_q4\_options\_cat1\_4[key]}  
 sx={{ mb: 1 }}  
 />  
 ))}  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q4.other\_cat1\_4"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Stack>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q6/q1.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t, setValue }) {  
 const q6\_1Ans = useWatch({ control, name: 'q6.q1\_answer' });  
 return (  
 <Box>  
 <QuestionHeader  
 question={`1. ${t('ccc.q6.q1.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q6.q1\_answer"  
 t={t}  
 onChange={() => {  
 setValue('q6.q1\_options\_cat1', [])  
 setValue('q6.q1\_options\_cat2', [])  
 setValue('q6.q1\_options\_cat3', [])  
 }}  
 />  
 {q6\_1Ans === "false" &&  
 <>  
 <Typography>{t('cat1')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q1\_options\_cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('ccc.q6.opt')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q1\_other\_cat1"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat2')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q1\_options\_cat2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('ccc.q6.opt')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q1\_other\_cat2"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat3')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q1\_options\_cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q1\_other\_cat3"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </>}  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q6/q2.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t, setValue }) {  
 const q6\_2Ans = useWatch({ control, name: 'q6.q2\_answer' });  
 return (  
 <Box>  
 <QuestionHeader  
 question={`2. ${t('ccc.q6.q2.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q6.q2\_answer"  
 t={t}  
 onChange={() => {  
 setValue('q6.q2\_options\_cat1', [])  
 setValue('q6.q2\_options\_cat2', [])  
 setValue('q6.q2\_options\_cat3', [])  
 }}  
 />  
 {q6\_2Ans === "false" &&  
 <>  
 <Typography>{t('cat1')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q2\_options\_cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('ccc.q6.opt')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q2\_other\_cat1"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat2')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q2\_options\_cat2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('ccc.q6.opt')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q2\_other\_cat2"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat3')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q2\_options\_cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q2\_other\_cat3"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </>}  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q6/q3.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t, setValue }) {  
 const q6\_3Ans = useWatch({ control, name: 'q6.q3\_answer' });  
 return (  
 <Box>  
 <QuestionHeader  
 question={`3. ${t('ccc.q6.q3.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q6.q3\_answer"  
 t={t}  
 onChange={() => {  
 setValue('q6.q3\_options\_cat1', [])  
 setValue('q6.q3\_options\_cat2', [])  
 setValue('q6.q3\_options\_cat3', [])  
 }}  
 />  
 {q6\_3Ans === "false" &&  
 <>  
 <Typography>{t('cat1')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q3\_options\_cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('ccc.q6.opt')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q3\_other\_cat1"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat2')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q3\_options\_cat2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('ccc.q6.opt')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q3\_other\_cat2"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat3')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q3\_options\_cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q3\_other\_cat3"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </>}  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q6/q4.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t, setValue }) {  
 const q6\_4Ans = useWatch({ control, name: 'q6.q4\_answer' });  
 return (  
 <Box>  
 <QuestionHeader  
 question={`4. ${t('ccc.q6.q4.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q6.q4\_answer"  
 t={t}  
 onChange={() => {  
 setValue('q6.q4\_options\_cat1', [])  
 setValue('q6.q4\_options\_cat2', [])  
 setValue('q6.q4\_options\_cat3', [])  
 }}  
 />  
 {q6\_4Ans === "false" &&  
 <>  
 <Typography>{t('cat1')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q4\_options\_cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('ccc.q6.opt')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q4\_other\_cat1"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat2')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q4\_options\_cat2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('ccc.q6.opt')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q4\_other\_cat2"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat3')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q4\_options\_cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q4\_other\_cat3"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </>}  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q6/q5.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t, setValue }) {  
 const q6\_5Ans = useWatch({ control, name: 'q6.q5\_answer' });  
 return (  
 <Box>  
 <QuestionHeader  
 question={`5. ${t('ccc.q6.q5.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q6.q5\_answer"  
 t={t}  
 onChange={() => {  
 setValue('q6.q5\_options\_cat1', [])  
 setValue('q6.q5\_options\_cat2', [])  
 setValue('q6.q5\_options\_cat3', [])  
 }}  
 />  
 {q6\_5Ans === "false" &&  
 <>  
 <Typography>{t('ccc.q6.q5.content')}</Typography>  
 <Typography>{t('cat1')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q5\_options\_cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q5\_other\_cat1"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat2')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q5\_options\_cat2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q5\_other\_cat2"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat3')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q5\_options\_cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q5\_other\_cat3"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </>}  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q6/q6a.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t, setValue }) {  
 const q6\_6aAns = useWatch({ control, name: 'q6.q6a\_answer' });  
 return (  
 <Box>  
 <QuestionHeader  
 question={`6a. ${t('ccc.q6.q6a.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q6.q6a\_answer"  
 t={t}  
 onChange={() => {  
 setValue('q6.q6a\_options\_cat1', [])  
 setValue('q6.q6a\_options\_cat2', [])  
 setValue('q6.q6a\_options\_cat3', [])  
 }}  
 />  
 <Typography>{t('ccc.q6.q6a.content')}</Typography>  
 <Typography>{t('cat1')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q6a\_options\_cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q6a\_other\_cat1"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat2')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q6a\_options\_cat2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q6a\_other\_cat2"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat3')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q6a\_options\_cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q6a\_other\_cat3"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box >  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q6/q6b.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t, setValue }) {  
 const q6\_6bAns = useWatch({ control, name: 'q6.q6b\_answer' });  
 return (  
 <Box>  
 <QuestionHeader  
 question={`6b. ${t('ccc.q6.q6b.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q6.q6b\_answer"  
 t={t}  
 onChange={() => {  
 setValue('q6.q6b\_options\_cat1', [])  
 setValue('q6.q6b\_options\_cat2', [])  
 setValue('q6.q6b\_options\_cat3', [])  
 }}  
 />  
 <Typography>{t('ccc.q6.q6b.content')}</Typography>  
 <Typography>{t('cat1')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q6b\_options\_cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q6b\_other\_cat1"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat2')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q6b\_options\_cat2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q6b\_other\_cat2"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography>{t('cat3')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q6.q6b\_options\_cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q6.q6b\_other\_cat3"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box >  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/index.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled'  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import Grid from '@mui/material/Grid'  
import RadioGroup from '@mui/material/RadioGroup'  
import FormGroup from '@mui/material/FormGroup'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import Radio from '@mui/material/Radio'  
import Checkbox from 'components/basic/Checkbox'  
import Typography from 'components/basic/Typography'  
import TextField from 'components/basic/TextField'  
import GradientButton from 'components/basic/Button/GradientButton'  
import FormControl from 'components/basic/FormControl'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable'  
import SubmissionTable from 'components/Case/SubmissionTable'  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import CommonFormHeader from 'components/Case/FormHeader'  
import QuestionHeader from 'components/Case/QuestionHeader'  
import Q1QuestionBody from 'components/Case/P3ForCCC/Q1'  
import Q3QuestionBody from 'components/Case/P3ForCCC/Q3'  
import Q4QuestionBody from 'components/Case/P3ForCCC/Q4'  
import Q5QuestionBody from 'components/Case/P3ForCCC/Q5'  
import Q6Dot1QuestionBody from 'components/Case/P3ForCCC/Q6/q1'  
import Q6Dot2QuestionBody from 'components/Case/P3ForCCC/Q6/q2'  
import Q6Dot3QuestionBody from 'components/Case/P3ForCCC/Q6/q3'  
import Q6Dot4QuestionBody from 'components/Case/P3ForCCC/Q6/q4'  
import Q6Dot5QuestionBody from 'components/Case/P3ForCCC/Q6/q5'  
import Q6Dot6aQuestionBody from 'components/Case/P3ForCCC/Q6/q6a'  
import Q6Dot6bQuestionBody from 'components/Case/P3ForCCC/Q6/q6b'  
function P3Task({ control, onSubmit, setValue }) {  
 const { t, i18n } = useTranslation(['P3Task'])  
 return (  
 <Stack direction="column">  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Typography variant="title2" mx={3}>  
 {t('structural')}  
 </Typography>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1. ${t('ccc.q1.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q1.answer"  
 t={t}  
 onChange={() => {  
 setValue('q1.options\_cat1', [])  
 setValue('q1.options\_cat2ia', [])  
 setValue('q1.options\_cat2ib', [])  
 setValue('q1.options\_cat2iia', [])  
 setValue('q1.options\_cat2iv', [])  
 setValue('q1.options\_cat3', [])  
 }}  
 />  
 <Q1QuestionBody control={control} t={t} />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`2. ${t('ccc.q2.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q2.answer"  
 t={t}  
 />  
 <Typography>{t('ccc.q2.content')}</Typography>  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`3. ${t('ccc.q3.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q3.answer"  
 t={t}  
 onChange={() => {  
 setValue('q3.options\_cat3', [])  
 }}  
 />  
 <Q3QuestionBody control={control} t={t} />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`4. ${t('ccc.q4.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q4.answer"  
 t={t}  
 onChange={() => {  
 setValue('q3.options\_cat2\_2a', [])  
 }}  
 />  
 <Q4QuestionBody control={control} t={t} />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`5. ${t('q5.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q5.answer"  
 t={t}  
 onChange={() => {  
 setValue('q5.options\_cat1\_1', [])  
 setValue('q5.options\_cat1\_2', [])  
 setValue('q5.options\_cat1\_3', [])  
 setValue('q5.options\_cat1\_4', [])  
 }}  
 />  
 <Q5QuestionBody control={control} t={t} />  
 </Box>  
 <Box px={4} py={3} >  
 <QuestionHeader  
 question={`6. ${t('ccc.q6.question')}`}  
 control={control}  
 needAnswer  
 ansFieldName="q6.answer"  
 t={t}  
 />  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Q6Dot1QuestionBody  
 control={control}  
 t={t}  
 setValue={setValue}  
 />  
 <Q6Dot2QuestionBody  
 control={control}  
 t={t}  
 setValue={setValue}  
 />  
 <Q6Dot3QuestionBody  
 control={control}  
 t={t}  
 setValue={setValue}  
 />  
 <Q6Dot4QuestionBody  
 control={control}  
 t={t}  
 setValue={setValue}  
 />  
 <Q6Dot5QuestionBody  
 control={control}  
 t={t}  
 setValue={setValue}  
 />  
 <Q6Dot6aQuestionBody  
 control={control}  
 t={t}  
 setValue={setValue}  
 />  
 <Q6Dot6bQuestionBody  
 control={control}  
 t={t}  
 setValue={setValue}  
 />  
 </Box>  
 </Box>  
 <GradientButton  
 sx={{ width: 100, alignSelf: 'flex-end', mr: 2 }}  
 onClick={onSubmit}  
 >  
 Save  
 </GradientButton>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default P3Task

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q1.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t }) {  
 const q1Ans = useWatch({ control, name: 'q1.answer' });  
 return (  
 <>  
 {  
 q1Ans === "false" && (  
 <Stack gap={2} sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{t('logicalQuestionYes', { ns: 'commonTask' })}</Typography>  
 <Typography>{t('cat1')}</Typography>  
 <Controller  
 name="q1.options\_cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 2 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q1.other\_cat1\_yes"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{t('cat2ia')}</Typography>  
 <Controller  
 name="q1.options\_cat2ia"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 2 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q1.other\_cat2ia\_yes"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{t('cat2ib')}</Typography>  
 <Controller  
 name="q1.options\_cat2ib"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 2 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q1.other\_cat2ib\_yes"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{t('cat2iia')}</Typography>  
 <Controller  
 name="q1.options\_cat2iia"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 2 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q1.other\_cat2iia\_yes"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{t('cat2ia')}</Typography>  
 <Controller  
 name="q1.options\_cat2iv"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 2 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q1.other\_cat2iv\_yes"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{t('cat3')}</Typography>  
 <Controller  
 name="q1.options\_cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 2 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q1.other\_cat3\_yes"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Stack>  
 )  
 }  
 </>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q3.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t }) {  
 const q3Ans = useWatch({ control, name: 'q3.answer' });  
 if (q3Ans !== 'true') return null  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{t('logicalQuestionYes', { ns: 'commonTask' })}</Typography>  
 <Typography>{t('cat3')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q3.options\_cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('ccc.q3.opt1')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('other', { ns: "commonTask" })}  
 />  
 <Controller  
 name="q3.other\_cat3"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q4.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t }) {  
 const q4Ans = useWatch({ control, name: 'q4.answer' });  
 if (q4Ans !== 'true') return null  
 return (  
 <Box sx={{ backgroundColor: '#fff', p: 2 }}>  
 <Typography>  
 {t('logicalQuestionYes', {  
 ns: 'commonTask',  
 })}  
 </Typography>  
 <Typography mt={2}>{t('cat2iia')}</Typography>  
 <Typography>{t('ccc.q4.content')}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q4.options\_cat2\_2a"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup sx={{ mt: 2 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'a'  
 )}  
 onChange={(e) => {  
 const isChecked =  
 e.target  
 .checked  
 if (isChecked) {  
 field.onChange(  
 [  
 ...field.value,  
 'a',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'a'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={t('ccc.q4.opt1')}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'b'  
 )}  
 onChange={(e) => {  
 const isChecked =  
 e.target  
 .checked  
 if (isChecked) {  
 field.onChange(  
 [  
 ...field.value,  
 'b',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'b'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={t('ccc.q4.opt2')}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'c'  
 )}  
 onChange={(e) => {  
 const isChecked =  
 e.target  
 .checked  
 if (isChecked) {  
 field.onChange(  
 [  
 ...field.value,  
 'c',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'c'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={t('ccc.q4.opt3')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e.target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={t('other', {  
 ns: 'commonTask',  
 })}  
 />  
 <Controller  
 name="q3.other\_cat2\_2a"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P3ForCCC/Q5.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t }) {  
 const q5Ans = useWatch({ control, name: 'q5.answer' });  
 if (q5Ans === "false") return null;  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{t('logicalQuestionYes', { ns: 'commonTask' })}</Typography>  
 <Typography mt={2}>{t('cat1')}</Typography>  
 <Typography>{t('q5.content')}</Typography>  
 <Stack gap={2} mt={2}>  
 <Typography variant="title2" sx={{ fontSize: 18 }}>{t('q5a.question')}</Typography>  
 <Box ml={4}>  
 <Controller  
 name="q5.options\_cat1\_1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('q5a.opt1')}  
 componentsProps={{  
 typography: {  
 sx: {  
 whiteSpace: 'break-spaces'  
 }  
 }  
 }}  
 sx={{ mb: 1 }}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={t('q5a.opt2')}  
 componentsProps={{  
 typography: {  
 sx: {  
 whiteSpace: 'break-spaces'  
 }  
 }  
 }}  
 sx={{ mb: 1 }}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('c')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'c']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'c'));  
 }  
 }}  
 />  
 }  
 label={t('q5a.opt3')}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('d')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'd']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'd'));  
 }  
 }}  
 />  
 }  
 label={t('q5a.opt4')}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('e')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'e']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'e'));  
 }  
 }}  
 />  
 }  
 label={t('q5a.opt5')}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('f')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'f']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'f'));  
 }  
 }}  
 />  
 }  
 label={t('q5a.opt6')}  
 />  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Stack>  
 <Stack gap={2} mt={2}>  
 <Typography variant="title2" sx={{ fontSize: 18 }}>{t('q5b.question')}</Typography>  
 <Box ml={4}>  
 <Controller  
 name="q5.options\_cat1\_2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('q5b.opt1')}  
 componentsProps={{  
 typography: {  
 sx: {  
 whiteSpace: 'break-spaces'  
 }  
 }  
 }}  
 sx={{ mb: 1 }}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={t('q5b.opt2')}  
 componentsProps={{  
 typography: {  
 sx: {  
 whiteSpace: 'break-spaces'  
 }  
 }  
 }}  
 sx={{ mb: 1 }}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('c')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'c']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'c'));  
 }  
 }}  
 />  
 }  
 label={t('q5b.opt3')}  
 />  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Stack>  
 <Stack gap={2} mt={2}>  
 <Typography variant="title2" sx={{ fontSize: 18 }}>{t('q5c.question')}</Typography>  
 <Box ml={4}>  
 <Controller  
 name="q5.options\_cat1\_3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('q5c.opt1')}  
 componentsProps={{  
 typography: {  
 sx: {  
 whiteSpace: 'break-spaces'  
 }  
 }  
 }}  
 sx={{ mb: 1 }}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={t('q5c.opt2')}  
 componentsProps={{  
 typography: {  
 sx: {  
 whiteSpace: 'break-spaces'  
 }  
 }  
 }}  
 sx={{ mb: 1 }}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('c')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'c']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'c'));  
 }  
 }}  
 />  
 }  
 label={t('q5c.opt3')}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('d')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'd']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'd'));  
 }  
 }}  
 />  
 }  
 label={t('q5c.opt4')}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('e')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'e']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'e'));  
 }  
 }}  
 />  
 }  
 label={t('q5c.opt5')}  
 />  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Stack>  
 <Stack gap={2} mt={2}>  
 <Typography variant="title2" sx={{ fontSize: 18 }}>{t('q5d.question')}</Typography>  
 <Box ml={4}>  
 <Controller  
 name="q5.options\_cat1\_4"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={t('q5d.opt1')}  
 componentsProps={{  
 typography: {  
 sx: {  
 whiteSpace: 'break-spaces'  
 }  
 }  
 }}  
 sx={{ mb: 1 }}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={t('q5d.opt2')}  
 componentsProps={{  
 typography: {  
 sx: {  
 whiteSpace: 'break-spaces'  
 }  
 }  
 }}  
 sx={{ mb: 1 }}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('c')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'c']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'c'));  
 }  
 }}  
 />  
 }  
 label={t('q5d.opt3')}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('d')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'd']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'd'));  
 }  
 }}  
 />  
 }  
 label={t('q5d.opt4')}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('e')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'e']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'e'));  
 }  
 }}  
 />  
 }  
 label={t('q5d.opt5')}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('f')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'f']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'f'));  
 }  
 }}  
 />  
 }  
 label={t('q5d.opt6')}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={t('q5d.opt7')}  
 />  
 <Controller  
 name="q5.other\_cat1\_4"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Stack>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P4/Q1/Q3.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control }) {  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`Are two independent means of escape provided for the exclusive use of the occupants of the kindergarten ? `}</Typography>  
 <Typography>{`[Cat. 3]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q1.q3.cat3"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`Two independent means of escape should be provided for the exclusive use of the occupants of the Kindergarten.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.q3.cat3\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P4/Q1/Q4.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control }) {  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`[Cat. 3]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q1.q4.cat3"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`Two independent means of escape should be provided for the exclusive use of the occupants of the Kindergarten.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.q4.cat3\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P4/Q1/Q5\_2.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control }) {  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`[Cat. 3]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q1.q5.q2.cat3"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`Addition and alteration works to comply with the requirements under FS Code 2011 and application for modification of Building (Planning) Regulations 49A to allow co-existence of the proposed kindergarten and the cinema in the building are required.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.q5.q2.cat3\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P4/Q1/Q9.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control }) {  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`[Cat. 3]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q1.q9.cat3"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`"The required staircases serving the storeys of the subject building above the ground storey should have a total discharge value of not less than the total occupant capacity of those storeys." `}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={`The no. and width of exit doors and exit routes of \_\_\_\_/F should be not less than those shown in Table B2 of FS Code according to the occupant capacity of the \_\_\_\_/F. `}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.q9.cat3\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 <Typography>[show below requirements in all case]</Typography>  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`[Cat.2(iv)]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q1.q9.cat2\_4"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`The opening of the required exit at the licensed premise should have a minimum clear width of \_\_\_mm (when the glass sliding door thereat is in an open position, if any).`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={`Minimum \_\_\_\_ exits should be provided to the proposed licensed premises. The \_\_\_\_ exits should have a minimum clear width of \_\_\_\_mm each and a minimum total clear width of \_\_\_mm. For double-leaf door, each leaf should have a clear width of not less than 600 mm. The lines of the travel distance between the two exits and any point in the premises should form an angle of not less than 30°. The exit doors should also open in the direction of exit and should not obstruct the exit route at any part of their swings.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.q9.cat2\_4\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P4/Q1/TextQuestion.jsx

import Stack from "@mui/material/Stack"  
import Typography from 'components/basic/Typography'  
import TextField from 'components/basic/TextField'  
import FormControl from 'components/basic/FormControl';  
import { Controller } from 'react-hook-form'  
export default function TextQuestion({ control, fieldName, question }) {  
 return (  
 <Stack direction="row" gap={2} sx={{ mt: 2, alignItems: 'center' }}>  
 <Typography sx={{ flex: 1 }}>{question}</Typography>  
 <Controller  
 name={fieldName}  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <FormControl labelPlacement="top" error={invalid} helperText={error?.message}>  
 <TextField {...otherField} />  
 </FormControl>  
 )}  
 />  
 </Stack>  
 )  
}

## File: bd-scs-backend-web/src/components/Case/P4/Q2/2a.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control }) {  
 const q2aAns = useWatch({ control, name: 'q2.q1.answer' });  
 if (q2aAns?.toString() !== "true") return null  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`(Logical question, appear if select "Yes")`}</Typography>  
 <Typography>{`[Cat.2(iv)]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q2.q1.cat2\_4"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`The clear width of door openings for Classroom \_\_\_ should not be less than \_\_\_ mm each.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={`Minimum \_\_\_\_ exits should be provided to Classroom \_\_\_\_. The \_\_\_\_ exits should have a minimum clear width of \_\_\_\_mm each and a minimum total clear width of \_\_\_\_mm. For double-leaf door, each leaf should have a clear width of not less than 600 mm. The lines of the travel distance between the two exits and any point in Classroom\_\_\_\_ should form an angle of not less than 30°. The exit doors should also open in the direction of exit and should not obstruct the exit route at any part of their swings.`}  
 componentsProps={{  
 typography: {  
 sx: {  
 whiteSpace: 'break-spaces',  
 my: 1,  
 }  
 },  
 }}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q2.q1.cat2\_4\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P4/Q2/2b.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control }) {  
 const q2bAns = useWatch({ control, name: 'q2.q2.answer' });  
 if (q2bAns?.toString() !== 'true') return null  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <>  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`(Logical question, appear if select "Yes")`}</Typography>  
 <Typography>{`[Cat.1]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q2.q2.cat1"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`The deadend travel distance should not exceed \_\_\_m from \_\_\_\_\_\_\_\_ to the protected exit/ discharge point of ultimate place of safety. Revised plan to demonstrate a satisfactory means of escape at \_\_\_\_\_ should be submitted for BD’s acceptance. Upon the receipt of the revised plan, additional building safety requirement may be imposed.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q2.q2.cat1\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 </>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P4/Q2/2c.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control }) {  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2, mt: 2 }}>  
 <Typography>{`[Cat.1]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q2.q3.cat1"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`The exit doors should be opened readily without the use of key from inside at all times and should not obstruct the common exit route at any part of their swing.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={`The roller shutter/ manual sliding door across the required exit should be kept fully open during operation hours.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('c')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'c']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'c'));  
 }  
 }}  
 />  
 }  
 label={`Automatic sliding door(s) across the required exit(s) should stay open to the full width in the event of power failure or fire.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('d')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'd']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'd'));  
 }  
 }}  
 />  
 }  
 label={`The area with headroom less than 2m should not be used for classroom/ teaching/ working area. `}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('e')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'e']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'e'));  
 }  
 }}  
 />  
 }  
 label={`Vision panel should be provided to the door or walls of \_\_\_\_\_\_ `}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('f')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'f']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'f'));  
 }  
 }}  
 />  
 }  
 label={`The lock installed at the exit door and furnitures/ fixtures obstructed the exit route should be removed. Photo records showing the completion of removal work should be submitted.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q2.q3.cat1\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Typography sx={{ mt: 2 }}>{`[Cat. 2(iv)]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q2.q3.cat2\_4"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`All passages leading to the exits with the licensed premises should have a minimum clear width of \_\_\_\_mm and minimum headroom of 2000mm. `}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={`The level difference across the required exit / exit route should be overcome by step(s) with a riser not higher than 175mm and treads of not less than 225mm wide.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('c')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'c']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'c'));  
 }  
 }}  
 />  
 }  
 label={`The exit door constructed to open both ways should have a transparent upper view panel.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('d')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'd']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'd'));  
 }  
 }}  
 />  
 }  
 label={`The gradient of the new ramp(s) forming part of exit route(s) should not be steeper than 1:12.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q2.q3.cat2\_4\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P4/index.jsx

import { Controller, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Box from '@mui/material/Box'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import FormGroup from '@mui/material/FormGroup'  
import FormControl from 'components/basic/FormControl'  
import Stack from '@mui/material/Stack'  
import Grid from '@mui/material/Grid'  
import TextField from 'components/basic/TextField'  
import Checkbox from 'components/basic/Checkbox'  
import GradientButton from 'components/basic/Button/GradientButton'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import Typography from 'components/basic/Typography'  
import CommonFormHeader from 'components/Case/FormHeader'  
import Q1\_3QuestionBody from 'components/Case/P4/Q1/Q3'  
import Q1\_4QuestionBody from 'components/Case/P4/Q1/Q4'  
import Q1\_5\_2QuestionBody from 'components/Case/P4/Q1/Q5\_2'  
import Q1\_9QuestionBody from 'components/Case/P4/Q1/Q9'  
import Q2aQuestionBody from 'components/Case/P4/Q2/2a'  
import Q2bQuestionBody from 'components/Case/P4/Q2/2b'  
import Q2cQuestionBody from 'components/Case/P4/Q2/2c'  
import Q3QuestionBody from 'components/Case/P4/Q3'  
import Q4QuestionBody from 'components/Case/P4/Q4'  
import QuestionHeader from 'components/Case/QuestionHeader'  
import TextQuestion from './Q1/TextQuestion'  
import Button from 'src/components/basic/Button'  
import DVTableTaskDialog from 'routes/case/DvTable'  
import { useState } from 'react'  
function P4Task({  
 deskStudy,  
 buildingInformation,  
 dv,  
 control,  
 onSubmit,  
 setValue,  
}) {  
 const q1\_3ans = useWatch({ control, name: 'q1.q3.answer' })  
 const q1\_4ans = useWatch({ control, name: 'q1.q4.answer' })  
 const q1\_5\_1ans = useWatch({ control, name: 'q1.q5.q1.answer' })  
 const q1\_5\_2ans = useWatch({ control, name: 'q1.q5.q2.answer' })  
 const a = buildingInformation?.typeOfBuilding === 'industrial'  
 const b = buildingInformation?.typeOfBuilding === 'NTEH2'  
 const c = buildingInformation?.typeOfBuilding === 'commercial'  
 const d = buildingInformation?.typeOfBuilding === 'composite'  
 const [dvDialog, setDvDialog] = useState({ open: false })  
 return (  
 <Stack direction="column">  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Typography variant="title2" mx={3}>{`MOE`}</Typography>  
 <Typography  
 variant="title2"  
 mx={3}  
 sx={{ fontSize: 18, mt: 1 }}  
 >{`MOE from room/compartment/storey`}</Typography>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1. Verify building information & update use & DV`}  
 />  
 </Box>  
 {a && (  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1.1 If (a) type of building = industrial, show`}  
 />  
 <Typography>{`[Cat. 3]`}</Typography>  
 <Typography>{`Kindergarten should not be located in an industrial building`}</Typography>  
 </Box>  
 )}  
 {b && (  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1.2 If (b) type of building = single staircase building other than a single-family domestic building not exceeding three main storeys or a New Territories "exempted" building , show`}  
 />  
 <Typography>{`[Cat. 3]`}</Typography>  
 <Typography>{`Two extra staircase must be provided to the premises to be licensed and the single staircase serving the whole building must be sealed off.`}</Typography>  
 </Box>  
 )}  
 {c && (  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1.3 If (c) type of building = commercial & premises above G/F, show`}  
 control={control}  
 needAnswer  
 ansFieldName="q1.q3.answer"  
 onChange={() => {  
 setValue('q1.q3.cat3', [])  
 setValue('q1.q3.cat3\_other', '')  
 }}  
 />  
 {q1\_3ans === 'false' && (  
 <Q1\_3QuestionBody control={control} />  
 )}  
 </Box>  
 )}  
 {d && (  
 <Box px={4} py={3}>  
 <Typography>  
 1.4 If (d) type of building = composite  
 building & premises above G/F, show  
 </Typography>  
 <TextQuestion  
 control={control}  
 fieldName="q1.q4.numberOfKindergartenFloor"  
 question="Nos. of entire Kindergarten floor"  
 />  
 <TextQuestion  
 control={control}  
 fieldName="q1.q4.numberOfCommercialFloor"  
 question="Total nos. of commercial floors above G/F"  
 />  
 <QuestionHeader  
 question={`Are two independent means of escape provided for the exclusive use of the occupants of the kindergarten ? `}  
 control={control}  
 needAnswer  
 ansFieldName="q1.q4.answer"  
 onChange={() => {  
 setValue('q1.q4.cat3', [])  
 setValue('q1.q4.cat3\_other', '')  
 }}  
 />  
 {q1\_4ans === 'true' && (  
 <Q1\_4QuestionBody control={control} />  
 )}  
 </Box>  
 )}  
 {deskStudy?.q7?.answer && (  
 <>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1.5.1 Exempted from Building Planning Regulation 49A granted ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q1.q5.q1.answer"  
 onChange={() => {  
 setValue('q1.q5.q2.answer', null)  
 setValue('q1.q5.cat3', [])  
 setValue('q1.q5.cat3\_other', '')  
 }}  
 />  
 </Box>  
 {q1\_5\_1ans === 'false' && (  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1.5.2 Adequate independent means of escape available for the cinema ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q1.q5.q2.answer"  
 onChange={() => {  
 setValue('q1.q5.cat3', [])  
 setValue('q1.q5.cat3\_other', '')  
 }}  
 />  
 {q1\_5\_2ans === 'false' && (  
 <Q1\_5\_2QuestionBody  
 control={control}  
 />  
 )}  
 </Box>  
 )}  
 </>  
 )}  
 {deskStudy?.q8?.answer &&  
 parseInt(deskStudy?.q8?.answer) > 24 && (  
 <Box px={4} py={3}>  
 <Typography>{`1.6 If (e) height of the premises above ground level > 24m, show`}</Typography>  
 <Typography>{`[Insert to standard letter]`}</Typography>  
 <Typography>  
 The height of the proposed school  
 premises is over 24m above ground level.  
 Education Bureau, with the advice of the  
 Director of Fire Services, may by notice  
 in writing authorize that any part of  
 such premises be situated at such height  
 greater than 24 m as may be specified in  
 the notice under Education Regulation 7.  
 </Typography>  
 </Box>  
 )}  
 {!(a || b || c || d) && (  
 <Box px={4} py={3}>  
 <Typography>{`1.7 If no non-compliance under (a) to (d), show`}</Typography>  
 <Typography>  
 There is no incompatible use.  
 </Typography>  
 </Box>  
 )}  
 {!(  
 deskStudy?.q8?.answer &&  
 parseInt(deskStudy?.q8?.answer) > 24  
 ) && (  
 <Box px={4} py={3}>  
 <Typography>{`1.8 If no non-compliance under (e), show`}</Typography>  
 <Typography>  
 There is no violation on 24m rule.  
 </Typography>  
 </Box>  
 )}  
 <Box px={4} py={3}>  
 {dv.acceptable ? (  
 <>  
 <Typography>{`1.9 If updated DV is acceptable, show`}</Typography>  
 <Typography>DV is not exceeded.</Typography>  
 </>  
 ) : (  
 <>  
 <Typography>{`1.9 If updated DV is not acceptable, show`}</Typography>  
 <Button  
 sx={{ my: 1 }}  
 onClick={() =>  
 setDvDialog({ open: true })  
 }  
 >  
 Revise capacity with agreement with  
 applicant  
 </Button>  
 <Typography>{`[show the following after update]`}</Typography>  
 <Typography>[Cat. 1]</Typography>  
 <Typography>  
 The maximum permitted capacity of the  
 Kindergarten should not exceed \_\_\_\_  
 persons including student and teacher.  
 Revised plan showing the number of  
 persons to be accommodated not exceeding  
 the maximum capacity should be submitted  
 for BD's acceptance. Upon the receipt of  
 the revised plan, additional building  
 safety requirement may be imposed.{' '}  
 </Typography>  
 <Q1\_9QuestionBody control={control} />  
 </>  
 )}  
 </Box>  
 <Typography variant="title2" mx={3}>  
 2. Layout Checking  
 </Typography>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`2.1 Any partitioned room within the school premises ? `}  
 control={control}  
 needAnswer  
 ansFieldName="q2.q1.answer"  
 onChange={() => {  
 setValue('q2.q1.cat2\_4', [])  
 setValue('q2.q1.cat2\_4\_other', '')  
 }}  
 />  
 <Q2aQuestionBody control={control} />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`2.2 [FS code Clause B11.2] Any deadend travel distance > 18m or >24m (balcony approach) or maximum travel distiance > 36m or >45m (balcony approach) ? `}  
 control={control}  
 needAnswer  
 ansFieldName="q2.q2.answer"  
 onChange={() => {  
 setValue('q2.q2.cat1', [])  
 setValue('q2.q2.cat1\_other', '')  
 }}  
 />  
 <Q2bQuestionBody control={control} />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`2.3 [FS code Sub-section B5, B13, B15, Clause B11.5] General requirement of exit routes/ exit doors`}  
 control={control}  
 />  
 <Q2cQuestionBody control={control} />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`3. Undertaking letter from management office/ applicant is required ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q3.answer"  
 onChange={() => {  
 setValue('q3.cat1', [])  
 setValue('q3.cat1\_other', '')  
 }}  
 />  
 <Q3QuestionBody control={control} />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`4. Fire engineering bounding conditions fulfilled ? `}  
 control={control}  
 needAnswer  
 ansFieldName="q4.answer"  
 onChange={() => {  
 setValue('q4.cat1', [])  
 setValue('q4.cat1\_other', '')  
 }}  
 />  
 <Q4QuestionBody control={control} />  
 </Box>  
 <GradientButton  
 sx={{ width: 100, alignSelf: 'flex-end', mr: 2 }}  
 onClick={onSubmit}  
 >  
 Save  
 </GradientButton>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <DVTableTaskDialog  
 open={dvDialog.open}  
 onClose={() => setDvDialog({ open: false })}  
 data={dv}  
 />  
 </Stack>  
 )  
}  
export default P4Task

## File: bd-scs-backend-web/src/components/Case/P4/Q3.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t }) {  
 const q3Ans = useWatch({ control, name: 'q3.answer' });  
 if (!q3Ans || q3Ans !== 'true') return null  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`(Logical question, appear if select "Yes")`}</Typography>  
 <Typography>{`[Cat.1]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q3.cat1"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`An undertaking letter from the owner/management company of the building confirming the following aspects should be submitted:  
(a) the \_\_\_\_\_\_\_mm wide common corridor (outside licensed premises) is under the management and control of the owner/management company; and  
(b) the aforesaid common corridor is maintained on site and is kept free from any obstruction at all times to serve as required exit route for the licensed premises.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={`An undertaking letter from the applicant confirming the following aspects should be submitted:  
The adjoining non-licence area is under same management and control by the applicant and the operation hours of the licensed premises should be the same or longer than the non-licence area. Revised plan with an annotation ""Licensed premsies and adjoining non-licence area are under the same control and management of the applicant"" should be submitted.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q3.cat1\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P4/Q4.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import CommonFormHeader from 'components/Case/FormHeader';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, t }) {  
 const q4Ans = useWatch({ control, name: 'q4.answer' });  
 if (q4Ans !== 'false') return null  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`(Logical question, appear if select "No")`}</Typography>  
 <Typography>{`[Cat. 1]`}</Typography>  
 <Box mx={2}>  
 <Controller  
 name="q4.cat1"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`The maximum number of occupants to be accommodated at \_\_\_\_\_\_\_\_\_\_\_ should be shown on the submitted plan to demonstrate the compliance with the bounding conditions of the latest accepted Fire Engineering Report. Applicant should provide clarification by submitting the relevant revised plan for BD's acceptance.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={`Name of rooms/areas should be indicated on the submitted licence plan and tally with the occupancy summary on the accepted building records and Fire Engineering Report. Applicant should provide clarification by submitting the relevant revised plan for BD's acceptance.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q4.cat1\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P5/index.jsx

import { useForm } from 'react-hook-form';  
import { useTranslation } from 'react-i18next';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import ResponsiveForm from 'components/basic/ResponsiveForm';  
import Typography from 'components/basic/Typography';  
import GradientButton from 'components/basic/Button/GradientButton';  
import CommonFormHeader from 'components/Case/FormHeader';  
import Q2QuestionBody from 'components/Case/P5/Q2';  
import Q3QuestionBody from 'components/Case/P5/Q3';  
import Q4QuestionBody from 'components/Case/P5/Q4';  
import Q5QuestionBody from 'components/Case/P5/Q5';  
import Q6QuestionBody from 'components/Case/P5/Q6';  
import QuestionHeader from 'components/Case/QuestionHeader';  
import Q1QuestionBody from 'components/Case/P5/Q1';  
function P5Task({ control, onSubmit, setValue }) {  
 return (  
 <Stack direction="column" >  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Typography variant="title2" mx={3}>FRC</Typography>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1. [FS Code Clause C7.1] Is the proposed school premises located inside a sub-divided floor (open plan design in approved plan)/ unit with the provision of common internal corridor ? `}  
 control={control}  
 needAnswer  
 ansFieldName="q1.answer"  
 onChange={() => {  
 setValue('q1.q1.answer', null)  
 setValue('q1.q2a.cat3', [])  
 setValue('q1.q2a.cat3\_other', null)  
 setValue('q1.q2b.answer', null)  
 setValue('q1.q2b.cat1', [])  
 setValue('q1.q2b.cat1\_other', null)  
 }}  
 />  
 <Q1QuestionBody  
 control={control}  
 setValue={setValue}  
 />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`2. [FS Code Clause C7.1] Is the proposed school premises sub-divided from an approved unit ? `}  
 control={control}  
 needAnswer  
 ansFieldName="q2.answer"  
 onChange={() => {  
 setValue('q2.q1.answer', null)  
 setValue('q2.q1.cat1', [])  
 setValue('q2.q1.cat1\_other', "")  
 setValue('q2.q2.answer', null)  
 setValue('q2.q2.cat2\_3b', [])  
 setValue('q2.q2.cat2\_3b\_other', "")  
 }}  
 />  
 <Q2QuestionBody  
 control={control}  
 setValue={setValue}  
 />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`3. [FS code Subsection C4, Clause C7.1, C13.1, C14.1, Table C1] Any alteration to approved compartment wall, approved fire rated wall or approved fire rated door is made ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q3.answer"  
 onChange={() => {  
 setValue('q3.cat2\_3b', [])  
 setValue('q3.cat2\_3b\_other', null)  
 }}  
 />  
 <Q3QuestionBody  
 control={control}  
 />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`4. [FS Code Subsection C9, Clause C16.5, FS Code Clause D11.1] Is FRP certification required for protected lobby/ staircase doors/ solid return on G/F ? `}  
 control={control}  
 needAnswer  
 ansFieldName="q4.answer"  
 onChange={() => {  
 setValue('q4.cat2\_3b', [])  
 setValue('q4.cat2\_3b\_other', "")  
 }}  
 />  
 <Q4QuestionBody  
 control={control}  
 />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`5. Fire engineering bounding conditions fulfilled ? `}  
 control={control}  
 needAnswer  
 ansFieldName="q5.answer"  
 onChange={() => {  
 setValue('q5.cat1', [])  
 setValue('q5.cat1\_other', null)  
 }}  
 />  
 <Q5QuestionBody  
 control={control}  
 />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`6. General Reminder`}  
 control={control}  
 />  
 <Q6QuestionBody  
 control={control}  
 />  
 </Box>  
 <GradientButton sx={{ width: 100, alignSelf: 'flex-end', mr: 2 }} onClick={onSubmit}>Save</GradientButton>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 );  
}  
export default P5Task;

## File: bd-scs-backend-web/src/components/Case/P5/Q1.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, setValue }) {  
 const q1Ans = useWatch({ control, name: 'q1.answer' });  
 const q11Ans = useWatch({ control, name: 'q1.q1.answer' });  
 const q12bAns = useWatch({ control, name: 'q1.q2b.answer' });  
 if (!q1Ans || q1Ans === "false") return null  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography mt={3}>{`(Logical question, appear if select "Yes")`}</Typography>  
 <QuestionHeader  
 question={`1.1 Is the common internal corridor fire rated ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q1.q1.answer"  
 onChange={() => {  
 setValue('q1.q2a.cat3', [])  
 setValue('q1.q2a.cat3\_other', "")  
 setValue('q1.q2b.answer', null)  
 setValue('q1.q2b.cat1', [])  
 setValue('q1.q2b.cat1\_other', "")  
 }}  
 />  
 {  
 q11Ans === 'false' && (  
 <>  
 <Typography sx={{ mt: 2 }}>{`1.2a [Cat.3]`}</Typography>  
 <Controller  
 name="q1.q2a.cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`Common internal corridor serving rooms or flats should be separated from rooms or flats by fire barriers having an fire resistance rating (FRR) of not less than that of \_\_\_\_\_ minutes.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.q2a.cat3\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes("other")} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </>  
 )  
 }  
 {  
 q11Ans === 'true' && (  
 <>  
 <QuestionHeader  
 question={`1.2b Is FRR certification required for the common internal corridor ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q1.q2b.answer"  
 onChange={() => {  
 setValue('q1.q2b.cat1', [])  
 setValue('q1.q2b.cat1\_other', "")  
 }}  
 />  
 {q12bAns === "true" &&  
 <>  
 <Typography>{`[Cat. 1]`}</Typography>  
 <Controller  
 name="q1.q2b.cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`An undertaking letter from the owner/management company of the building confirming the following aspects should be submitted:  
(a) The management company is to ensure that with the endorsement of an AP/RSE, the common internal corridors are constructed with materials having an FRR of not less than \_\_\_\_ minutes and the doors thereat are having an FRR of not less than \_\_\_\_ minutes and be self-closing. An AP/RSE is required to check and certify the fire resistance rating (FRR) and submit duly completed Appendix A to Practice Note for Authorized Persons and Registered Structural Engineers APP-13 by an AP/RSE with supporting test/assessment reports. (Such reports are not required for submission if Practice Note for Authorized Persons and Registered Structural Engineers ADM-20 is applicable.)`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.q2b.cat1\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes("other")} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </>}  
 </>  
 )  
 }  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P5/Q2.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, set, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import QuestionHeader from 'components/Case/QuestionHeader';  
function QuestionBody({ control, setValue }) {  
 const q2Ans = useWatch({ control, name: 'q2.answer' });  
 const q21Ans = useWatch({ control, name: 'q2.q1.answer' });  
 const q22Ans = useWatch({ control, name: 'q2.q2.answer' });  
 if (!q2Ans || q2Ans === "false") return null  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography mt={3}>{`(Logical question, appear if select "Yes")`}</Typography>  
 <QuestionHeader  
 question={`2.1 Are the propsoed school premises and sub-divided non-licence area under the same manamgenet and control of applicant ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q2.q1.answer"  
 onChange={() => {  
 setValue('q2.q1.cat1', [])  
 setValue('q2.q1.cat1\_other', "")  
 setValue('q2.q2.cat2\_3b', [])  
 setValue('q2.q2.cat2\_3b\_other', "")  
 }}  
 />  
 {  
 q21Ans === "true" && (  
 <>  
 <Typography>{`[Cat. 1]`}</Typography>  
 <Controller  
 name="q2.q1.cat1"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`An undertaking letter from the applicant confirming the following aspects should be submitted:  
The adjoining non-licence area is under same management and control by the applicant and the operation hours of the licensed premises should be the same or longer than the non-licence area. Revised plan with an annotation ""Licensed premsies and adjoining non-licence area are under the same control and management of the applicant"" should be submitted.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q2.q1.cat1\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes("other")} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </>  
 )  
 }  
 {  
 q21Ans === "false" && (  
 <>  
 <QuestionHeader  
 question={`2.2 Is FRR certification required for the separating wall between proposed school premises and sub-divided non-licence area ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q2.q2.answer"  
 onChange={() => {  
 setValue('q2.q2.cat2\_3b', [])  
 setValue('q2.q2.cat2\_3b\_other', "")  
 }}  
 />  
 {  
 q22Ans === "true" && (  
 <>  
 <Typography>{`[Cat. 2(iii)(b)]`}</Typography>  
 <Controller  
 name="q2.q2.cat2\_3b"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`The enclosure walls separating the licensed premises from adjoining units which is non-licence area, as highlighted on the attached plan, should be constructed up to structural ceiling and have an FRR of not less than \_\_\_\_\_ minutes.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q2.q2.cat2\_3b\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes("other")} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </>  
 )  
 }  
 </>  
 )  
 }  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P5/Q3.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
function QuestionBody({ control, t }) {  
 const q3Ans = useWatch({ control, name: 'q3.answer' });  
 if (!q3Ans || q3Ans === "false") return null  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`(Logical question, appear if select "Yes")`}</Typography>  
 <Typography>{`[Cat. 2(iii)(b)]`}</Typography>  
 <Controller  
 name="q3.cat2\_3b"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`The altered walls, as highlighted on the attached plan, should be constructed up to structural ceiling and have an FRR of not less than \_\_\_\_\_ minutes.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={`The entrance door of the licensed premises should have an FRR of not less than \_\_\_\_\_ minutes. Such door should be self-closing and provided with smoke seals.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('c')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'c']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'c'));  
 }  
 }}  
 />  
 }  
 label={`Fire stop for sealing up openings for passage of building services through the the enclosure wall of the licensed premises facing the common internal corridor having an FRR of not less than \_\_\_\_\_ minutes.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('d')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'd']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'd'));  
 }  
 }}  
 />  
 }  
 label={`Fire dampers for protecting openings for passage of ventilation ducts through the the enclosure wall of the licensed premises facing the common internal corridor having an FRR of not less than \_\_\_\_\_ minutes.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q3.cat2\_3b\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes("other")} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P5/Q4.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
function QuestionBody({ control, t }) {  
 const q4Ans = useWatch({ control, name: 'q4.answer' });  
 if (!q4Ans || q4Ans === "false") return null  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`(Logical question, appear if select "Yes")`}</Typography>  
 <Typography>{`[Cat. 2(iii)(b)]`}</Typography>  
 <Controller  
 name="q4.cat2\_3b"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`The new walls and doors for the fireman's lift lobby/ protected lobby of exit staircase should have an FRR of not less than \_\_\_ minutes. The walls should be constructed up to the structural ceiling and the doors should be self-closing and provided with vision panels and smoke seals.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={`The new doors of Male/ Female LAV. within the protected lobby of the exit staircase, as highlighted on the attached plan, should have an FRR of not less than \_\_\_ minutes. Such door should be self-closing and provided with smoke seals.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('c')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'c']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'c'));  
 }  
 }}  
 />  
 }  
 label={`The solid return having a minimum width of 450mm up to structural floor should be provided along the shopfront of the premises adjacent to the exit staircase of the building and should have an FRR of not less than 60 minutes.`}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q4.cat2\_3b\_other"  
 control={control}  
 render={({ field: reasonField, fieldState: { invalid, error } }) => (  
 <TextField {...reasonField} disabled={!field.value?.includes("other")} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P5/Q5.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
function QuestionBody({ control, t }) {  
 const q5Ans = useWatch({ control, name: 'q5.answer' });  
 if (!q5Ans || q5Ans === "true") return null  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`(Logical question, appear if select "No")`}</Typography>  
 <Typography>{`[Cat.1]`}</Typography>  
 <Controller  
 name="q5.cat1"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q5.cat1\_other"  
 control={control}  
 render={({ field: reasonField, fieldState: { invalid, error } }) => (  
 <TextField {...reasonField} disabled={!field.value?.includes("other")} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P5/Q6.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
function QuestionBody({ control, t }) {  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`[Cat. 1]`}</Typography>  
 <Controller  
 name="q6.cat1"  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('a')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'a']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'a'));  
 }  
 }}  
 />  
 }  
 label={`The existing approved self-closing fire resisting door should not be altered and should be maintained in good condition and working in order. `}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('b')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'b']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'b'));  
 }  
 }}  
 />  
 }  
 label={`The approved fire shutters for compartment purpose within the licensed premises should not be obstructed by furniture and fixtures.`}  
 />  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('c')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'c']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'c'));  
 }  
 }}  
 />  
 }  
 label={`The approved smoke vents / smoke extraction ducts should not be altered and obstructed at all times. `}  
 />  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q6.cat1\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P6/Q1/index.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
export const options = [  
 {  
 label: "Unauthorised rooftop/flat roof/lane/yard/open area structures forming part of the licensed premises.",  
 value: "a"  
 },  
 {  
 label: "Unauthorised structures on/suspended from balcony/verandah/canopy. ",  
 value: "b"  
 },  
 {  
 label: "Unauthorised canopies/retractable awnings/projections projecting from the external wall of building erected after the implementation of such MW items under the Minor Works Control System (MWCS). ",  
 value: "c"  
 },  
 {  
 label: "Unauthorised signboards (including shopfront signboards) not eligible for joining the Signboard Validation Scheme. ",  
 value: "d"  
 },  
 {  
 label: "Unauthorised supporting structures/frames for air-conditioning units/light fittings/mechanical plants (e.g. water cooling towers and associated supporting structures) projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof erected after the implementation of such MW items under the MWCS. ",  
 value: "e"  
 },  
 {  
 label: "Unauthorised external metal ventilation ducts and associated supporting frames projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof erected after the implementation of such MW items under the MWCS. ",  
 value: "f"  
 },  
 {  
 label: "Unauthorised obstructions to smoke vents",  
 value: "g"  
 },  
 {  
 label: "Unauthorised alteration/removal of compartment walls/fire resisting walls and doors.",  
 value: "h"  
 },  
 {  
 label: "Unauthorised openings/slab over existing floors for pipe ducts. ",  
 value: "i"  
 },  
 {  
 label: "Unauthorised installation of glass panels on external walls on upper floors. ",  
 value: "j"  
 },  
 {  
 label: "Unauthorised removal of protective barrier/external wall without providing a permanent protective barrier at a height of 1.1 m minimum resulting in a danger of falling persons or objects. ",  
 value: "k"  
 },  
 {  
 label: "Unauthorised alteration/removal of approved barrier free facilities for persons with a disability. ",  
 value: "l"  
 },  
 {  
 label: "Unauthorised hollow raised platforms within premises hindering the access for persons with a disability. ",  
 value: "m"  
 },  
 {  
 label: "Unauthorised removal of internal staircases not having been certified by AP/RSE as being structurally safe and accompanied with supporting calculations. ",  
 value: "n"  
 },  
 {  
 label: "Unauthorised supporting frames for suspending an air-conditioning/mechanical ventilation plants of weight more than 150 kg inside the licensed premises not having been certified by AP/RSE as being structurally safe and accompanied with supporting calculations. ",  
 value: "o"  
 },  
 {  
 label: "Unauthorised external solid fence walls/mesh fences/railings/poles on roof erected after the implementation of such MW items under the MWCS. ",  
 value: "p"  
 },  
 {  
 label: "Unauthorised slabs filling up approved cockloft/staircase voids. ",  
 value: "q"  
 },  
 {  
 label: "Unauthorised cocklofts, intermediate floors and floor extensions. ",  
 value: "r"  
 },  
 {  
 label: "Unauthorised staircases. ",  
 value: "s"  
 },  
 {  
 label: "Unauthorised openings through structural slabs and walls. ",  
 value: "t"  
 },  
 {  
 label: "Unauthorised removal, partial removal/major alteration of structural members. ",  
 value: "u"  
 },  
 {  
 label: "UBWs in common areas resulting in obstruction of means of escape from the licensed premises/the building.",  
 value: "v"  
 },  
]  
function QuestionBody({ control, setValue }) {  
 const watchQ1\_1Options = useWatch({ control, name: 'q1.q1.options' });  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`List of Objectionable UBWs`}</Typography>  
 <Controller  
 name="q1.options"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 options.map((item) => {  
 return (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 setValue('q1.answer', 'true')  
 } else {  
 const newV = [...field.value].filter((v) => v !== item.value);  
 const newOptions = [...watchQ1\_1Options].filter((v) => v.value !== item.value);  
 setValue('q1.q1.options', newOptions);  
 field.onChange(newV);  
 if (newV.length <= 0) {  
 setValue('q1.answer', null)  
 setValue('q1.other', "")  
 setValue('q1.q1.options', [])  
 }  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 )  
 })  
 }  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 setValue('q1.answer', 'true')  
 } else {  
 const newV = [...field.value].filter((v) => v !== 'other');  
 const newOptions = [...watchQ1\_1Options].filter((v) => v.value !== 'other');  
 field.onChange(newV);  
 setValue('q1.q1.options', newOptions);  
 if (newV.length <= 0) {  
 setValue('q1.answer', null)  
 setValue('q1.other', "")  
 setValue('q1.q1.options', [])  
 }  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P6/Q1/Q1.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import React from 'react';  
import { findIndex } from 'lodash';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
import { options } from './index'  
function QuestionBody({ control }) {  
 const q1Options = useWatch({ control, name: 'q1.options' });  
 const q1Other = useWatch({ control, name: 'q1.other' });  
 const opts = React.useMemo(() => {  
 return (  
 q1Options?.map((v) => {  
 if (v === "other") {  
 return {  
 label: "Other",  
 value: "other"  
 }  
 }  
 return options.find((o) => {  
 return o.value === v  
 })  
 })  
 )  
 }, [q1Options])  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`List of Objectionable UBWs`}</Typography>  
 <Controller  
 name={`q1.q1.options`}  
 control={control}  
 defaultValue={[]}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 opts.map((item, index) => {  
 const find = field.value?.find((v) => v.value === item.value);  
 if (item.value === "other") {  
 return (  
 <Grid container direction="row" alignItems="center" justifyContent="space-between" mt={2}>  
 <Grid item>  
 <Typography>{`${item.label}: ${q1Other}`}</Typography>  
 </Grid>  
 <Grid item>  
 <FormControl  
 labelPlacement="top"  
 fullWidth  
 >  
 <RadioGroup  
 row  
 value={find?.type}  
 onChange={(e) => {  
 const v = e.target.value;  
 const tmpIndex = findIndex(field.value, { value: 'other' });  
 if (tmpIndex >= 0) {  
 let tmpValue = [...field.value];  
 tmpValue[tmpIndex].type = v;  
 field.onChange([...tmpValue]);  
 } else {  
 const tmpValue = {  
 value: 'other',  
 type: v  
 }  
 field.onChange([...field.value, tmpValue]);  
 }  
 }}  
 >  
 <FormControlLabel value={true} control={<Radio />} label={`Yes`} />  
 <FormControlLabel value={false} control={<Radio />} label={`No`} />  
 </RadioGroup>  
 </FormControl>  
 </Grid>  
 </Grid>  
 )  
 }  
 return (  
 <Grid container direction="row" alignItems="center" justifyContent="space-between" mt={2}>  
 <Grid item>  
 <Typography>{item.label}</Typography>  
 </Grid>  
 <Grid item>  
 <FormControl  
 labelPlacement="top"  
 fullWidth  
 >  
 <RadioGroup  
 row  
 value={find?.type}  
 onChange={(e) => {  
 const v = e.target.value;  
 const tmpIndex = findIndex(field.value, { value: item.value });  
 if (tmpIndex >= 0) {  
 let tmpValue = [...field.value];  
 tmpValue[tmpIndex].type = v;  
 field.onChange([...tmpValue]);  
 } else {  
 const tmpValue = {  
 value: item.value,  
 type: v  
 }  
 field.onChange([...field.value, tmpValue]);  
 }  
 }}  
 >  
 <FormControlLabel value={true} control={<Radio />} label={`Yes`} />  
 <FormControlLabel value={false} control={<Radio />} label={`No`} />  
 </RadioGroup>  
 </FormControl>  
 </Grid>  
 </Grid>  
 )  
 })  
 }  
 </FormGroup>  
 )  
 }}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P6/Q1/Q2.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import React from 'react';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
const options = [  
 {  
 label: "Unauthorised rooftop/flat roof/lane/yard/open area structures should be removed. ",  
 value: "a"  
 },  
 {  
 label: "Unauthorised structures on/suspended from balcony/verandah/canopy should be removed.",  
 value: "b"  
 },  
 {  
 label: "Unauthorised canopies/retractable awnings/projections projecting from the external wall of building should be removed. ",  
 value: "c"  
 },  
 {  
 label: "Unauthorised signboards should be removed. ",  
 value: "d"  
 },  
 {  
 label: "Unauthorised supporting structures/frames for air-conditioning units/light fittings/mechanical plants projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof should be removed. ",  
 value: "e"  
 },  
 {  
 label: "Unauthorised external metal ventilation ducts and associated supporting frames projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof should be removed. ",  
 value: "f"  
 },  
 {  
 label: "Any obstruction to smoke vents is removed and smoke vents reinstated in accordance with the approved plans. The AP/RSE is required to certify the fire resistance rating (FRR) for the material used for reinstatement of the smoke vents and submit duly completed Appendix A to Practice Note for Authorized Persons and Registered Structural Engineers and Registered Geotechnical Engineers (PNAP) APP-13 with supporting test/assessment reports to BD (Such reports are not required for submission if PNAP ADM-20 is applicable). ",  
 value: "g"  
 },  
 {  
 label: "Unauthorised alteration/removal of compartment walls/fire resisting walls and doors should be reinstated in accordance with approved plan. ",  
 value: "h"  
 },  
 {  
 label: "Unauthorised openings/ slab over existing floors for pipe ducts should be reinstated in accordance with approved plan. ",  
 value: "i"  
 },  
 {  
 label: "Unauthorised installation of glass panels on external walls on upper floors should be removed and the configuration of external walls should be reinstated in accordance with approved plan. ",  
 value: "j"  
 },  
 {  
 label: "Protective barrier/external wall should be reinstated in accordance with approved plan. ",  
 value: "k"  
 },  
 {  
 label: "The access ramp for persons with a disability/ The accessible toilet for persons with a disability / The accessible urinal should be reinstated in accordance with approved plan. ",  
 value: "l"  
 },  
 {  
 label: "Unauthorised hollow raised platforms within premises hindering the access for persons with a disability should be removed. ",  
 value: "m"  
 },  
 {  
 label: "Internal staircase should be reinstated in accordance with approved plan. ",  
 value: "n"  
 },  
 {  
 label: "Unauthorised supporting frames for suspending an air-conditioning/ mechanical ventilation plants of weight more than 150 kg inside the premises should be removed. ",  
 value: "o"  
 },  
 {  
 label: "Unauthorised external solid fence walls/mesh fences/railings/poles on roof should be removed. ",  
 value: "p"  
 },  
 {  
 label: "Unauthorised slabs filling up approved cockloft/staircase voids should be removed and reinstated in accordance with approved plan. ",  
 value: "q"  
 },  
 {  
 label: "Unauthorised cocklofts/ intermediate floors/ floor extensions should be removed.",  
 value: "r"  
 },  
 {  
 label: "Unauthorised staircases should be removed.",  
 value: "s"  
 },  
 {  
 label: "Unauthorised openings through structural slabs / walls should be rectified and reinstated in accordance with approved plan. ",  
 value: "t"  
 },  
 {  
 label: "Unauthorised removal / partial removal / major alteration of structural members should be resinstated in accordance with approved plan. ",  
 value: "u"  
 },  
 {  
 label: "UBWs in common areas resulting in obstruction of means of escape from the licensed premises/the building should be removed. ",  
 value: "v"  
 },  
]  
function QuestionBody({ control }) {  
 const q1\_1Options = useWatch({ control, name: 'q1.q1.options' });  
 const q1Other = useWatch({ control, name: 'q1.other' });  
 const optList = React.useMemo(() => {  
 if (!q1\_1Options || !q1\_1Options.length) return []  
 return q1\_1Options.filter((o) => o.type === 'false').map((item) => {  
 if (item.value === "other") {  
 return {  
 label: `Other: ${q1Other}`,  
 value: "other"  
 }  
 }  
 return options.find((o) => o.value === item.value)  
 })  
 }, [q1\_1Options, q1Other])  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`[Cat. 3]`}</Typography>  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 optList.map((item) => {  
 return (  
 <Typography>{item.label}</Typography>  
 )  
 })  
 }  
 </FormGroup>  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P6/Q1/Q3.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import React from 'react';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
const options\_cat1\_1a = [  
 {  
 label: "Construction/Alteration/Repair/Replacement/Removal of window or window wall. (建造/改動/修葺/更換/拆除窗或玻璃外牆)",  
 value: "a"  
 },  
 {  
 label: "Erection/Alteration/Removal of supporting frames for air-conditioning unit projecting from an external wall of a building. (豎設/改動/拆除自建築物外牆伸出的，用於支承空調機的支架)",  
 value: "b"  
 },  
 {  
 label: "Erection/Alteration/Removal of external Ventilation duct works and associated supporting frames projecting from an external wall of a building. (豎設/改動/拆除自建築物外牆伸出的室外金屬通風管道或相關的承托支架)",  
 value: "c"  
 },  
 {  
 label: "Erection/Alteration/Removal of wall signboard. (豎設/改動/拆除 靠牆招牌)",  
 value: "d"  
 }  
]  
const options\_cat1\_1b = [  
 {  
 label: "Removal of unauthorised structures projecting from the external wall of a building. (拆除自建築物外牆伸出的違例構築物)",  
 value: "e"  
 },  
 {  
 label: "Removal of unauthorised suspended slab/unauthorised structure hung underneath the soffit of a balcony or canopy (拆除懸掛式違例樓板／懸掛或固定於露台或簷篷部分的違例構築物)",  
 value: "f"  
 },  
 {  
 label: "Removal of any unauthorized structure located on-grade or on a slab (other than a cantilevered slab) ((拆除位於地面或平板 ( 不包括懸臂式平板 ) 上的違例構築物)",  
 value: "g"  
 },  
 {  
 label: "Erection/Alteration/Removal of protective barrier. (豎設/改動/拆除防護欄障)",  
 value: "h"  
 }  
]  
const options\_cat1\_1c = [  
 {  
 label: "Erection/Alteration/Repair/Removal of retractable awnings. (豎設/改動/修葺/拆除可收合遮篷)",  
 value: "i"  
 },  
 {  
 label: "Erection/Alteration/Removal of external non-load bearing reinforced concrete wall of a building. (豎設/改動/拆除用鋼筋混凝土建造的非承重外牆)",  
 value: "j"  
 },  
 {  
 label: "Erection/Alteration/Removal of external non-load bearing block wall of a building. (豎設/改動/拆除用磚建造的非承重外牆)",  
 value: "k"  
 },  
 {  
 label: "Erection/Alteration/Repair/ Removal of mesh fences or metal railings on a roof (豎設／改動／ 修葺／拆除建築物屋頂上的網欄或金屬欄杆)",  
 value: "l"  
 }  
]  
const options\_cat1 = [  
 {  
 label: "Photo records showing the completion of removal/ rectification/ reinstatement works of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ should be submitted for BD’s acceptance.",  
 value: "a"  
 }  
]  
const options\_cat4 = [  
 {  
 label: "Unauthorised rooftop/flat roof/lane/yard/open area structures should be removed.",  
 value: "a"  
 },  
 {  
 label: "Unauthorised structures on/suspended from balcony/verandah/canopy should be removed.",  
 value: "b"  
 },  
 {  
 label: "Unauthorised canopies/retractable awnings/projections projecting from the external wall of building should be removed. ",  
 value: "c"  
 },  
 {  
 label: "Unauthorised signboards should be removed. ",  
 value: "d"  
 },  
 {  
 label: "Unauthorised supporting structures/frames for air-conditioning units/light fittings/mechanical plants projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof should be removed. ",  
 value: "e"  
 },  
 {  
 label: "Unauthorised external metal ventilation ducts and associated supporting frames projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof should be removed.",  
 value: "f"  
 },  
 {  
 label: "Any obstruction to smoke vents is removed and smoke vents reinstated in accordance with the approved plans. The AP/RSE is required to certify the fire resistance rating (FRR) for the material used for reinstatement of the smoke vents and submit duly completed Appendix A to Practice Note for Authorized Persons and Registered Structural Engineers and Registered Geotechnical Engineers (PNAP) APP-13 with supporting test/assessment reports to BD (Such reports are not required for submission if PNAP ADM-20 is applicable).",  
 value: "g"  
 },  
 {  
 label: "Unauthorised alteration/removal of compartment walls/fire resisting walls and doors should be reinstated in accordance with approved plan.",  
 value: "h"  
 },  
 {  
 label: "Unauthorised openings/ slab over existing floors for pipe ducts should be reinstated in accordance with approved plan.",  
 value: "i"  
 },  
 {  
 label: "Unauthorised installation of glass panels on external walls on upper floors should be removed and the configuration of external walls should be reinstated in accordance with approved plan.",  
 value: "j"  
 },  
 {  
 label: "Protective barrier/external wall should be reinstated in accordance with approved plan.",  
 value: "k"  
 },  
 {  
 label: "The access ramp for persons with a disability/ The accessible toilet for persons with a disability / The accessible urinal should be reinstated in accordance with approved plan.",  
 value: "l"  
 },  
 {  
 label: "Unauthorised hollow raised platforms within premises hindering the access for persons with a disability should be removed.",  
 value: "m"  
 },  
 {  
 label: "Internal staircase should be reinstated in accordance with approved plan.",  
 value: "n"  
 },  
 {  
 label: "Unauthorised supporting frames for suspending an air-conditioning/ mechanical ventilation plants of weight more than 150 kg inside the premises should be removed.",  
 value: "o"  
 },  
 {  
 label: "Unauthorised external solid fence walls/mesh fences/railings/poles on roof should be removed.",  
 value: "p"  
 },  
 {  
 label: "Unauthorised slabs filling up approved cockloft/staircase voids should be removed and reinstated in accordance with approved plan.",  
 value: "q"  
 },  
 {  
 label: "Unauthorised cocklofts/ intermediate floors/ floor extensions should be removed.",  
 value: "r"  
 },  
 {  
 label: "Unauthorised staircases should be removed.",  
 value: "s"  
 },  
 {  
 label: "Unauthorised openings through structural slabs / walls should be rectified and reinstated in accordance with approved plan.",  
 value: "t"  
 },  
 {  
 label: "Unauthorised removal / partial removal / major alteration of structural members should be resinstated in accordance with approved plan.",  
 value: "u"  
 },  
 {  
 label: "UBWs in common areas resulting in obstruction of means of escape from the licensed premises/the building should be removed.",  
 value: "v"  
 }  
]  
function QuestionBody({ control }) {  
 const q1\_1Options = useWatch({ control, name: 'q1.q1.options' });  
 const q1Other = useWatch({ control, name: 'q1.other' });  
 const trueOptList = React.useMemo(() => {  
 if (!q1\_1Options || !q1\_1Options.length) return []  
 return q1\_1Options.filter((o) => o.type === 'true').map((item) => {  
 if (item.value === "other") {  
 return {  
 label: `Other: ${q1Other}`,  
 value: "other"  
 }  
 }  
 return options\_cat4.find((o) => o.value === item.value)  
 })  
 }, [q1\_1Options, q1Other])  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`[Cat. 1(i)]`}</Typography>  
 <Controller  
 name="q1.q3.cat1\_1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 <Typography>Common Minor Works</Typography>  
 {  
 options\_cat1\_1a.map((item) => {  
 return (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== item.value));  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 )  
 })  
 }  
 <Typography>Structural Alteration or Removal works</Typography>  
 {  
 options\_cat1\_1b.map((item) => {  
 return (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== item.value));  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 )  
 })  
 }  
 <Typography>Roof or External Works</Typography>  
 {  
 options\_cat1\_1c.map((item) => {  
 return (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== item.value));  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 )  
 })  
 }  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.q3.cat1\_1\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{`[Cat. 1]`}</Typography>  
 <Controller  
 name="q1.q3.cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 options\_cat1.map((item) => {  
 return (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== item.value));  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 )  
 })  
 }  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.q3.cat1\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Box mb={4} mt={4}>  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 trueOptList.map((item) => {  
 return (  
 <Typography>{item.label}</Typography>  
 )  
 })  
 }  
 </FormGroup>  
 </Box>  
 <Typography>{`[Cat. 2(iv)]`}</Typography>  
 <Controller  
 name="q1.q3.cat2\_4"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 options\_cat4.map((item) => {  
 return (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== item.value));  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 )  
 })  
 }  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.q3.cat2\_4\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P6/Q2/Q1.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
const options = [  
 {  
 label: "Supporting structures for building services installation (BSI) located on-grade/on roof (other than cantilevered slab/inaccessible roof) with the height of the supporting structures not more than 1.5 m.",  
 value: "a"  
 },  
 {  
 label: "Shopfront side structures (SSS), projecting not more than 150 mm from the original building line. The wall area covered by any one side of SSS is not more than 5 m2, the distance between any part of the SSS and the ground is not more than 6 m and the SSS do not obstruct access to any public utilities. ",  
 value: "b"  
 },  
 {  
 label: "Shopfront overhead projections, which do not consist of wooden structural elements, projecting not more than 600 mm from the external wall of building having minimum vertical clearance of 2.5 m and not supporting other structures or equipment such as air-conditioning units. If the overhead projections consist of stone, the difference between any part of the projection and the ground should not be more than 6 m. ",  
 value: "c"  
 },  
 {  
 label: "Canopies, which do not consist of stone, tile, glass/cement mortar and are not constructed of concrete, projecting not more than 500 mm from the external wall of building. The distance between the highest point of the canopy and the roof/ground is more than 3 m.",  
 value: "d"  
 },  
 {  
 label: "Retractable awnings at a door opening not serving as an exit for an escape staircase/ leading to a balcony/verandah or a window opening for uses other than plant room/ lavatory/bathroom/kitchen, projecting not more than 2 m (over the roof)/2.5 m (any other cases) from the external wall of building when fully extended and not more than 500 mm when retracted. The distance between the highest point of the awning and the roof/ground is not more than 5.5 m. The width of awning is at most 500 mm wider than both sides of such opening.",  
 value: "e"  
 },  
 {  
 label: "Existing signboards validated through the Signboard Validation Scheme (SVS).",  
 value: "f"  
 },  
 {  
 label: "Supporting frames for air-conditioning units/lighting fittings projecting not more than 600 mm from the external wall of building. The distance between the highest point of the frame and the roof/ground is more than 3 m.",  
 value: "g"  
 },  
 {  
 label: "External metal ventilation ducts and the associated supporting frames located on-grade/on roof. The distance between the highest point of the duct/frame and the roof/ground is not more than 1.5 m.",  
 value: "h"  
 },  
 {  
 label: "External metal ventilation ducts and associated supporting frames projecting not more than 600 mm from the external wall of building. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m.",  
 value: "i"  
 },  
 {  
 label: "External metal ventilation ducts and the associated supporting frames hung underneath the soffit of a balcony/verandah/canopy (other than cantilevered slab) with largest cross-sectional dimension of the duct not more than 600 mm. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m.",  
 value: "j"  
 },  
 {  
 label: "External metal ventilation ducts and the associated supporting frames located on a balcony/verandah/canopy (other than cantilevered slab) with largest cross-sectional dimension of the duct not more than 600 mm. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m. The distance between the highest point of the duct/frame and the balcony/verandah/canopy is not more than 1.5 m.",  
 value: "k"  
 },  
 {  
 label: "External poles located on roof including any feature at its top with the height not more than 1.5 m.",  
 value: "l"  
 },  
 {  
 label: "External solid fence walls located on roof with the height not more than 1.1 m and thickness not more than 100 mm.",  
 value: "m"  
 },  
 {  
 label: "External mesh fences/railings located on roof including any feature at its top with the height not more than 1.5 m and is not used as a protective barrier. If lower part is a solid fence wall, the height and thickness are not more than 300 mm and 100 mm respectively.",  
 value: "n"  
 },  
 {  
 label: "Unauthorised installation of glass panels on external walls on upper floors having area of each glass panel not exceeding 6 m and the least dimension of such glass panel not exceeding 1.8 m and not overloading to cantilevered slab type balconies. The remaining external wall can provide adequate protection for openings to prevent the spread of fire between floors concerned. ",  
 value: "o"  
 },  
 {  
 label: "Supporting frames for suspending an air-conditioning/mechanical ventilation plant of more than 150 kg inside a building certified by AP/RSE as being structurally safe with supporting calculations. ",  
 value: "p"  
 },  
 {  
 label: "Unauthorised openings/slabs over existing floors for food hoists and pipe ducts having been certified by AP/RSE as being structurally safe with supporting calculations and should not result in additional gross floor area under the Building (Planning) Regulations.",  
 value: "q"  
 },  
 {  
 label: "Unauthorised alteration/ removal of approved access ramp for persons with a disability / accessible toilet for persons with a disability / accessible urinal subject to re-provision of access ramp for persons with a disability/ accessible toilet for persons with a disability / accessible urinal in compliance with Design Manual: Barrier Free Access.",  
 value: "r"  
 },  
 {  
 label: "Hollow raised platforms with a height not greater than 600 mm within premises and not hindering the access for persons with a disability.",  
 value: "s"  
 },  
 {  
 label: "Hollow raised platforms with a height between 600 mm to 2 000 mm within premises if such platforms having been certified by AP/RSE as being structurally safe with supporting calculations and not hindering the access for persons with a disability.",  
 value: "t"  
 },  
 {  
 label: "Removal of internal staircase having been certified by AP/RSE as being structurally safe with supporting calculations and the staircase void should be provided by protective barriers in accordance with Building (Construction) Regulation.",  
 value: "u"  
 },  
 {  
 label: "Kitchens and toilets within approved premises and drainage works certified as being in proper function and properly connected.",  
 value: "v"  
 },  
 {  
 label: "Small storage chamber for LPG cylinders with an aggregate capacity not exceeding 130 L and complying with “LPG Installation for Catering Purposes in Commercial Premises” issued by the Electrical and Mechanical Services Department.",  
 value: "w"  
 }  
]  
function QuestionBody({ control }) {  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`insert selected item to Cat. 2(i)(b) & Para. 3(a) of Standard Letter`}</Typography>  
 <Controller  
 name="q2.q1.option"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 options.map((item, index) => (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== item.value));  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 ))  
 }  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q2.q1.other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P6/Q2/Q2.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
const options = [  
 {  
 label: "Supporting frames for suspending an air-conditioning/mechanical ventilation plant of more than 150 kg inside a building",  
 value: "a"  
 },  
 {  
 label: "Unauthorised openings/slabs over existing floors for pipe ducts",  
 value: "b"  
 },  
 {  
 label: "Hollow raised platforms with a height between 600 mm to 2 000 mm within premises",  
 value: "c"  
 },  
 {  
 label: "Removal of internal staircase",  
 value: "d"  
 },  
]  
function QuestionBody({ control }) {  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`[Cat. 2(i)(a)]`}</Typography>  
 <Controller  
 name="q2.q2.cat2\_1a"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 options.map((item, index) => (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== item.value));  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 ))  
 }  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q2.q2.cat2\_1a\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P6/index.jsx

import { useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import { Controller } from 'react-hook-form'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import Grid from '@mui/material/Grid'  
import Radio from '@mui/material/Radio'  
import RadioGroup from '@mui/material/RadioGroup'  
import FormControl from 'components/basic/FormControl'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import Typography from 'components/basic/Typography'  
import GradientButton from 'components/basic/Button/GradientButton'  
import CommonFormHeader from 'components/Case/FormHeader'  
import Q1QuestionBody from 'components/Case/P6/Q1'  
import Q1\_1QuestionBody from 'components/Case/P6/Q1/Q1'  
import Q1\_2uestionBody from 'components/Case/P6/Q1/Q2'  
import Q1\_3uestionBody from 'components/Case/P6/Q1/Q3'  
import Q2\_1QuestionBody from 'components/Case/P6/Q2/Q1'  
import Q2\_2QuestionBody from 'components/Case/P6/Q2/Q2'  
import Q3QuestionBody from 'components/Case/P6/Q3'  
import Q4QuestionBody from 'components/Case/P6/Q4'  
import QuestionHeader from 'components/Case/QuestionHeader'  
function P6Task({ control, onSubmit, setValue }) {  
 const q1Ans = useWatch({ control, name: 'q1.answer' })  
 const q2\_2Ans = useWatch({ control, name: 'q2.q2.answer' })  
 const q3Ans = useWatch({ control, name: 'q3.answer' })  
 const q4Ans = useWatch({ control, name: 'q4.answer' })  
 const q1\_1\_ans = useWatch({ control, name: 'q1.q1.options' })  
 console.log('q1\_ans', q1Ans)  
 console.log('q1\_1\_ans', q1\_1\_ans)  
 return (  
 <Stack direction="column">  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Typography variant="title2" mx={3}>  
 {'UBW'}  
 </Typography>  
 <Box px={4} py={3}>  
 <Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`1. [LU Manual Section 1.8] Any objectionable UBW found ?`}</Typography>  
 </Grid>  
 <Grid item>  
 <Controller  
 name="q1.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(  
 e.target.value  
 )  
 setValue(  
 'q1.options',  
 []  
 )  
 setValue('q1.other', '')  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={  
 <Radio disabled />  
 }  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={false}  
 control={  
 <Radio  
 onClick={() => {  
 field.onChange(  
 e.target  
 .value  
 )  
 setValue(  
 'q1.options',  
 []  
 )  
 }}  
 />  
 }  
 label={`No`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 <Q1QuestionBody  
 control={control}  
 setValue={setValue}  
 />  
 </Box>  
 {q1Ans === 'true' && (  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1.1 Has the applicant advised that the objectionable UBW is under his/her control and to be removed or rectified ?`}  
 control={control}  
 onChange={() => {  
 setValue('q1.q2.cat3', [])  
 setValue('q1.q2.cat3\_other', '')  
 setValue('q1.q3.cat1\_1', [])  
 setValue('q1.q3.cat1\_1\_other', '')  
 setValue('q1.q3.cat1', [])  
 setValue('q1.q3.cat1\_other', '')  
 setValue('q1.q3.cat2\_4', [])  
 setValue('q1.q3.cat2\_4\_other', '')  
 }}  
 />  
 <Q1\_1QuestionBody control={control} />  
 </Box>  
 )}  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1.2 `}  
 control={control}  
 />  
 <Q1\_2uestionBody control={control} />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1.3 `}  
 control={control}  
 />  
 <Q1\_3uestionBody control={control} />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`2. [LU Manual Section 1.8] Any mentionable UBW found ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q2.answer"  
 />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`2.1 Select the mentionable UBW below:`}  
 control={control}  
 />  
 <Q2\_1QuestionBody control={control} />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`2.2 Select the mentionable UBW below:`}  
 control={control}  
 needAnswer  
 ansFieldName="q2.q2.answer"  
 />  
 {q2\_2Ans === 'true' && (  
 <Q2\_2QuestionBody control={control} />  
 )}  
 </Box>  
 <Box px={4} py={1}>  
 <QuestionHeader  
 question={`3. Is there any re-provision of barrier free facilities for persons with disability (e.g. ramp, accessible toilet etc.) ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q3.answer"  
 onChange={(e) => {  
 setValue('q3.cat2\_4', [])  
 setValue('q3.cat2\_4\_other', '')  
 }}  
 />  
 {q3Ans === 'true' && (  
 <Q3QuestionBody control={control} />  
 )}  
 </Box>  
 <Box px={4} py={1}>  
 <QuestionHeader  
 question={`4. Any Dedicated area / Public Passage to be affected ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q4.answer"  
 onChange={() => {  
 setValue('q4.cat1', [])  
 setValue('q4.cat1\_other', '')  
 setValue('q4.cat3', [])  
 setValue('q4.cat3\_other', '')  
 }}  
 />  
 {q4Ans === 'true' && (  
 <Q4QuestionBody control={control} />  
 )}  
 </Box>  
 <Box px={4} py={1}>  
 <QuestionHeader  
 question={`5. Is any objectionable UBW outside licence area required to be referred to Existing Buildings Division ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q5.answer"  
 />  
 <Box sx={{ p: 2 }}>  
 <Typography>{`SL23aL / SL23bL to be prepared`}</Typography>  
 </Box>  
 </Box>  
 <GradientButton  
 sx={{ width: 100, alignSelf: 'flex-end', mr: 2 }}  
 onClick={onSubmit}  
 >  
 Save  
 </GradientButton>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default P6Task

## File: bd-scs-backend-web/src/components/Case/P6/Q3.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
const options = [  
 {  
 label: "The accessible ramp/ accessible toilet for persons with a disability should be reinstated in accordance with the approved plan or constructed in accordance with “Design Manual Barrier Free Access 2008”.",  
 value: "a"  
 }  
]  
function QuestionBody({ control }) {  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`(Logical question, appear if select "Yes")`}</Typography>  
 <Typography>{`[Cat. 2(iv)]`}</Typography>  
 <Controller  
 name="q3.cat2\_4"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 options.map((item, index) => (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== item.value));  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 ))  
 }  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q3.cat2\_4\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/P6/Q4.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormGroup from '@mui/material/FormGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Radio from '@mui/material/Radio';  
import Checkbox from 'components/basic/Checkbox';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable';  
import SubmissionTable from 'components/Case/SubmissionTable';  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor';  
const options\_cat1 = [  
 {  
 label: "Revised plan showing the removal/ deleteion of \_\_\_\_\_\_\_ should be submitted for BD's acceptance. Upon the receipt of the revised plan, additional building safety requirement may be imposed.",  
 value: "a"  
 }  
]  
const options\_cat3 = [  
 {  
 label: "Structure within dedicated area / public passage which were excluded from GFA calculation in the original building design should be removed.",  
 value: "a"  
 }  
]  
function QuestionBody({ control }) {  
 return (  
 <Box sx={{ backgroundColor: "#fff", p: 2 }}>  
 <Typography>{`(Logical question, appear if select "Yes")`}</Typography>  
 <Typography>{`[Cat. 1]`}</Typography>  
 <Controller  
 name="q4.cat1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 options\_cat1.map((item, index) => (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== item.value));  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 ))  
 }  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q4.cat1\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{`[Cat. 3]`}</Typography>  
 <Controller  
 name="q4.cat3"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormGroup sx={{ mt: 1, ml: 4 }}>  
 {  
 options\_cat3.map((item, index) => (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(item.value)}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, item.value]);  
 } else {  
 field.onChange(field.value.filter((v) => v !== item.value));  
 }  
 }}  
 />  
 }  
 label={item.label}  
 />  
 ))  
 }  
 <Grid container direction="row">  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes('other')}  
 onChange={(e) => {  
 const isChecked = e.target.checked;  
 if (isChecked) {  
 field.onChange([...field.value, 'other']);  
 } else {  
 field.onChange(field.value.filter((v) => v !== 'other'));  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q4.cat3\_other"  
 control={control}  
 render={({ field: otherField, fieldState: { invalid, error } }) => (  
 <TextField {...otherField} disabled={!field.value?.includes('other')} />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 )  
}  
export default QuestionBody;

## File: bd-scs-backend-web/src/components/Case/SCH/Info.jsx

import { Controller } from 'react-hook-form'  
import Grid from '@mui/material/Grid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import DatePicker from 'components/basic/DatePicker'  
const RowItem = ({ control, fieldNameA, titleA, fieldNameB, titleB, component }) => {  
 const RenderComponent = component === "datePicker" ? DatePicker : TextField  
 return (  
 <Grid container direction="row" gap={2}>  
 <Grid item xs>  
 <Controller  
 name={fieldNameA}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={titleA}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <RenderComponent {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name={fieldNameB}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={titleB}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <RenderComponent {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 )  
}  
function InfoInput({ control }) {  
 return (  
 <Grid container direction="column" px={2} pt={4} gap={2}>  
 <RowItem  
 control={control}  
 fieldNameA={'completedBy'}  
 titleA={'Completed by:'}  
 fieldNameB={'checked'}  
 titleB={'Checked:'}  
 />  
 <RowItem  
 control={control}  
 fieldNameA={'nameA'}  
 titleA={'Name:'}  
 fieldNameB={'nameB'}  
 titleB={'Name:'}  
 />  
 <RowItem  
 control={control}  
 fieldNameA={'postA'}  
 titleA={'Post:'}  
 fieldNameB={'postB'}  
 titleB={'Post:'}  
 />  
 <RowItem  
 control={control}  
 fieldNameA={'dateA'}  
 titleA={'Date:'}  
 fieldNameB={'dateB'}  
 titleB={'Date:'}  
 component="datePicker"  
 />  
 </Grid>  
 )  
}  
export default InfoInput

## File: bd-scs-backend-web/src/components/Case/SCH/TableA.jsx

import { Controller } from 'react-hook-form'  
import Paper from '@mui/material/Paper'  
import Table from '@mui/material/Table'  
import TableBody from '@mui/material/TableBody'  
import TableCell from '@mui/material/TableCell'  
import TableContainer from '@mui/material/TableContainer'  
import TableHead from '@mui/material/TableHead'  
import TableRow from '@mui/material/TableRow'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
function TableA({ control }) {  
 return (  
 <TableContainer component={Paper}>  
 <Table sx={{ minWidth: 650 }} aria-label="simple table">  
 <TableHead>  
 <TableRow>  
 <TableCell>Item</TableCell>  
 <TableCell>Remarks</TableCell>  
 </TableRow>  
 </TableHead>  
 <TableBody>  
 <TableRow>  
 <TableCell><span style={{ marginRight: 30 }}>1</span>{`Latest submission for BD’s comments at f( ), LU’s latest Letter of Requirements (LoR) at f( )`}</TableCell>  
 <TableCell>  
 <Controller  
 name="q1.remarks"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 </TableRow>  
 <TableRow  
 sx={{ border: 0 }}  
 >  
 <TableCell><span style={{ marginRight: 30 }}>2</span>{`Prescribed form \*SC(CCC)-1/SC(CCC)-2 at f( )`}</TableCell>  
 <TableCell>  
 <Controller  
 name="q2.remarks"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 </TableRow>  
 </TableBody>  
 </Table>  
 </TableContainer>  
 )  
}  
export default TableA

## File: bd-scs-backend-web/src/components/Case/SCH/TableB.jsx

import { Controller } from 'react-hook-form'  
import Paper from '@mui/material/Paper'  
import Table from '@mui/material/Table'  
import TableBody from '@mui/material/TableBody'  
import TableCell from '@mui/material/TableCell'  
import TableContainer from '@mui/material/TableContainer'  
import TableHead from '@mui/material/TableHead'  
import TableRow from '@mui/material/TableRow'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Checkbox from 'components/basic/Checkbox'  
const TableRowItem = ({ control, no, item, yesNoFieldName, disabledNa, hideNa = "", remarksFieldName, remarksPlaceholder }) => {  
 return (  
 <TableRow>  
 <TableCell>{no}</TableCell>  
 <TableCell>{item}</TableCell>  
 <Controller  
 name={yesNoFieldName}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => {  
 return (  
 <>  
 <TableCell>  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Checkbox  
 {...field}  
 checked={field?.value === 'yes'}  
 disableRipple sx={{ padding: 0 }}  
 onChange={(e) => field.onChange(e.target.checked ? 'yes' : '')}  
 />  
 </FormControl>  
 </TableCell>  
 <TableCell>  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Checkbox  
 {...field}  
 checked={field?.value === 'no'}  
 disableRipple sx={{ padding: 0 }}  
 onChange={(e) => field.onChange(e.target.checked ? 'no' : '')}  
 />  
 </FormControl>  
 </TableCell>  
 {!hideNa &&  
 <TableCell>  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Checkbox  
 {...field}  
 checked={field?.value === 'na'}  
 disableRipple  
 sx={{ padding: 0 }}  
 disabled={disabledNa}  
 onChange={(e) => field.onChange(e.target.checked ? 'na' : '')}  
 />  
 </FormControl>  
 </TableCell>  
 }  
 </>  
 )  
 }}  
 />  
 <TableCell>  
 <Controller  
 name={remarksFieldName}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} helperText={remarksPlaceholder} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 </TableRow>  
 )  
}  
function TableB({ control }) {  
 return (  
 <TableContainer component={Paper}>  
 <Table sx={{ minWidth: 650 }} aria-label="simple table">  
 <TableHead>  
 <TableRow>  
 <TableCell>No.</TableCell>  
 <TableCell>Item</TableCell>  
 <TableCell align="center">Yes</TableCell>  
 <TableCell align="center">No</TableCell>  
 <TableCell align="center">N/A</TableCell>  
 <TableCell>Remarks (Give details in remarks)</TableCell>  
 </TableRow>  
 </TableHead>  
 <TableBody>  
 <TableRow>  
 <TableCell colspan="6">Prescribed form</TableCell>  
 </TableRow>  
 <TableRowItem  
 control={control}  
 no="3"  
 item="Prescribed form in order?"  
 yesNoFieldName="q3.answer"  
 disabledNa  
 remarksFieldName="q3.remarks"  
 />  
 <TableRow>  
 <TableCell colspan="6">Supporting Documents</TableCell>  
 </TableRow>  
 <TableRowItem  
 control={control}  
 no="4"  
 item={`All the required supporting documents (e.g. structural justification PNAP APP-13, photo records etc.) submitted?`}  
 yesNoFieldName="q4.answer"  
 remarksFieldName="q4.remarks"  
 />  
 <TableRow>  
 <TableCell colspan="6">Layout plan</TableCell>  
 </TableRow>  
 <TableRowItem  
 control={control}  
 no="5"  
 item={`Finalised layout plan incorporating BD’s comment submitted in order?`}  
 yesNoFieldName="q5.answer"  
 remarksFieldName="q5.remarks"  
 remarksPlaceholder="(e.g. If discrepancy was noted on the final layout plan, please select “No”.)"  
 />  
 <TableRowItem  
 control={control}  
 no="6"  
 item={`UBWs plan (if any) submitted?`}  
 yesNoFieldName="q6.answer"  
 remarksFieldName="q6.remarks"  
 />  
 </TableBody>  
 </Table>  
 </TableContainer>  
 )  
}  
export default TableB

## File: bd-scs-backend-web/src/components/Case/SCH/TableC.jsx

import { Controller } from 'react-hook-form'  
import Paper from '@mui/material/Paper'  
import Table from '@mui/material/Table'  
import TableBody from '@mui/material/TableBody'  
import TableCell from '@mui/material/TableCell'  
import TableContainer from '@mui/material/TableContainer'  
import TableHead from '@mui/material/TableHead'  
import TableRow from '@mui/material/TableRow'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Checkbox from 'components/basic/Checkbox'  
function TableC({ control }) {  
 return (  
 <TableContainer component={Paper}>  
 <Table sx={{ minWidth: 650 }} aria-label="simple table">  
 <TableHead>  
 <TableRow>  
 <TableCell>No.</TableCell>  
 <TableCell>Item</TableCell>  
 <TableCell align="center">Yes</TableCell>  
 <TableCell align="center">No</TableCell>  
 <TableCell>Remarks (Give details in remarks)</TableCell>  
 </TableRow>  
 </TableHead>  
 <TableBody>  
 <TableRow>  
 <TableCell>7</TableCell>  
 <TableCell>{`Issuing of Certificates/Compliance with building safety requirements in the LoR is recommended? `}</TableCell>  
 <Controller  
 name={'q7.answer'}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <>  
 <TableCell>  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Checkbox  
 {...field}  
 checked={field?.value === 'yes'}  
 disableRipple  
 sx={{ padding: 0 }}  
 onChange={(e) => field.onChange(e.target.checked ? 'yes' : '')}  
 />  
 </FormControl>  
 </TableCell>  
 <TableCell>  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Checkbox  
 {...field}  
 checked={field?.value === 'no'}  
 disableRipple  
 sx={{ padding: 0 }}  
 onChange={(e) => field.onChange(e.target.checked ? 'no' : '')}  
 />  
 </FormControl>  
 </TableCell>  
 </>  
 )}  
 />  
 <TableCell>  
 <Controller  
 name={'q7.remarks'}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </TableCell>  
 </TableRow>  
 </TableBody>  
 </Table>  
 </TableContainer>  
 )  
}  
export default TableC

## File: bd-scs-backend-web/src/components/Case/Eminutes.jsx

import { Controller, useForm } from 'react-hook-form'  
import React, { useContext, useState } from 'react'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Select from '@mui/material/Select'  
import MenuItem from '@mui/material/MenuItem'  
import GradientButton from 'components/basic/Button/GradientButton'  
import DataTable from 'components/basic/DataGrid'  
import FormControl from 'components/basic/FormControl'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import TextField from 'components/basic/TextField'  
import Typography from 'components/basic/Typography'  
import Button from 'components/basic/Button'  
import EminuteDialog from 'components/Dialog/EMinutesDialog'  
import KeyboardArrowDownIcon from '@mui/icons-material/KeyboardArrowDown'  
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import { useTranslation } from 'react-i18next'  
import { useParams } from 'react-router-dom'  
import { userContext } from 'src/context/UserProvider'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { ExpandLessOutlined, ExpandMoreOutlined } from '@mui/icons-material'  
import constants from 'src/constants'  
const EMinutes = ({ caseDetails }) => {  
 const { t, i18n } = useTranslation(['p1Task'])  
 const [snackbar, setSnackbar] = useContext(snackbarContext)  
 const [eminuteDialog, setEminuteDialog] = useState({  
 open: false,  
 id: null,  
 })  
 const [expanded, setExpanded] = useState(false)  
 const handleToggleExpand = () => {  
 setExpanded(!expanded)  
 }  
 const handleOnClickCreateEMinutes = () => {  
 setEminuteDialog({ open: true, id: null })  
 }  
 return (  
 <Box sx={{ position: 'relative' }}>  
 <Box  
 position="sticky"  
 py={2}  
 sx={{  
 cursor: 'pointer',  
 backgroundColor: 'white',  
 top: 0,  
 left: 0,  
 zIndex: 999,  
 }}  
 >  
 <Typography  
 variant="title2"  
 color="text.secondary"  
 position="sticky"  
 onClick={handleToggleExpand}  
 >  
 {t('eminutes')}{' '}  
 {expanded ? <ExpandMoreOutlined /> : <ExpandLessOutlined />}  
 </Typography>  
 </Box>  
 {expanded && (  
 <Stack direction="column" gap={2}>  
 <Button  
 sx={{ alignSelf: 'flex-end' }}  
 onClick={handleOnClickCreateEMinutes}  
 >  
 Create E-minutes  
 </Button>  
 <Box sx={{ height: 300, overflow: 'auto', mt: 3 }}>  
 <DataTable  
 api={`/cases/${caseDetails?.\_id}/eminutes`}  
 columns={[  
 {  
 field: 'from',  
 headerName: 'From',  
 width: 200,  
 },  
 {  
 field: 'to',  
 headerName: 'To',  
 width: 200,  
 },  
 {  
 field: 'efolio',  
 headerName: 'E-Minute',  
 width: 200,  
 },  
 {  
 field: 'subject',  
 headerName: 'Subject',  
 flex: 1,  
 },  
 {  
 field: 'status',  
 headerName: 'Status',  
 width: 200,  
 },  
 {  
 field: 'createdAt',  
 headerName: 'Date',  
 width: 200,  
 valueGetter: (params) => {  
 return dayjs(params).format(constants.DATE\_FORMAT)  
 },  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 color: '#fff',  
 },  
 '& .MuiDataGrid-columnHeaderCheckbox': {  
 color: '#fff',  
 },  
 '& .MuiButtonBase-root': {  
 color: '#000',  
 },  
 }}  
 onRowClick={(row) => {  
 setEminuteDialog({  
 open: true,  
 id: row.row.\_id,  
 })  
 }}  
 />  
 </Box>  
 {eminuteDialog.open ? (  
 <EminuteDialog  
 open={eminuteDialog.open}  
 onClose={() => setEminuteDialog({ open: false, id: null })}  
 caseDetails={caseDetails}  
 title={  
 eminuteDialog?.id ? 'Update E-Minutes' : 'Create E-Minutes'  
 }  
 eminId={eminuteDialog.id}  
 caseId={caseDetails?.\_id}  
 />  
 ) : null}  
 </Stack>  
 )}  
 </Box>  
 )  
}  
export default EMinutes

## File: bd-scs-backend-web/src/components/Case/FolioHistoryTable.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs';  
import { useParams } from 'react-router-dom';  
import React from 'react'  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import Divider from '@mui/material/Divider';  
import IconButton from '@mui/material/IconButton';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import Autocomplete from 'components/basic/Autocomplete';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import AddCircleOutlinedIcon from '@mui/icons-material/AddCircleOutlined';  
import { snackbarContext } from 'src/context/SnackbarProvider'  
const Container = styled(Box)`  
 border: 2px solid #00838f;  
 border-radius: 5px;  
 display: flex;  
 flex-direction: row;  
 justify-content: center;  
 background-color: #fff;  
`  
const Header = styled(Grid)`  
 width: 220px;  
 height: 60px;  
 background-color: #e0f7fa;  
 border-top-left-radius: 5px;  
 border-bottom-left-radius: 5px;  
`  
const ContentContiner = styled(Grid)`  
 width: 220px;  
 height: 60px;  
 align-items: center;  
 justify-content: center;  
 border-bottom: ${props => props.hideBorder ? 0 : 1}px solid #9e9e9e;  
 margin-top: 0.5px;  
`  
function FolioHistoryTable({ control, hideAddButton, sx, name }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId } = useParams();  
 const { fields, append } = useFieldArray({  
 control,  
 name: `${name}value`  
 });  
 return (  
 <Container sx={sx}>  
 <Grid container direction="row" flexWrap="nowrap" sx={{ width: 'fit-content' }}>  
 <Grid item container direction="column" sx={{ width: 'fit-content' }}>  
 <Grid item sx={{ borderBottom: `1px solid #9e9e9e` }}>  
 <Header item container alignItems="center" px={2}>  
 <Typography>{`Submission(s)`}</Typography>  
 </Header>  
 </Grid>  
 <Grid item sx={{ borderBottom: `1px solid #9e9e9e` }}>  
 <Header item container alignItems="center" px={2}>  
 <Typography>{`LU Reply`}</Typography>  
 </Header>  
 </Grid>  
 <Grid item sx={{ borderBottom: `1px solid #9e9e9e` }}>  
 <Header item container alignItems="center" px={2}>  
 <Typography>{`Nature`}</Typography>  
 </Header>  
 </Grid>  
 <Grid item sx={{ borderBottom: `1px solid #9e9e9e` }}>  
 <Header item container alignItems="center" px={2}>  
 <Typography>Inspection Report</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header item container alignItems="center" px={2}>  
 <Typography>Licence Plan</Typography>  
 </Header>  
 </Grid>  
 </Grid>  
 </Grid>  
 <Grid container direction="row" flexWrap="nowrap" sx={{ overflow: 'auto' }}>  
 {  
 fields.map((item, index) => {  
 return (  
 <Grid item container direction="column" key={`fields-${item.id}`} sx={{ width: 'fit-content', }}>  
 <ContentContiner item container>  
 <Controller  
 name={`${name}value.${index}.submission`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <Autocomplete  
 api={`/applications/${applicationId}/cases`}  
 getOptionLabel={(option) => option?.\_id}  
 onError={(error) => {  
 return setSnackbar({  
 type: 'error',  
 message: `Cannot fetch "/applications/${applicationId}/cases"`,  
 })  
 }}  
 {...field}  
 onChange={(e, newValue) => {  
 field.onChange(newValue)  
 }}  
 />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container>  
 <Controller  
 name={`${name}value.${index}.luReply`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <Typography sx={{ display: 'flex', flexDirection: 'row', alignItems: 'center' }}>  
 {`f(`}  
 <TextField {...field} sx={{ width: 40, mx: 1 }} />  
 {`)`}  
 </Typography>  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container hideBorder>  
 <TextField sx={{ width: 80, mx: 1 }} value={item?.submission?.Nature || ""} disabled />  
 </ContentContiner>  
 <ContentContiner item container hideBorder>  
 <TextField sx={{ width: 80, mx: 1 }} disabled />  
 </ContentContiner>  
 <ContentContiner item container hideBorder>  
 <TextField sx={{ width: 80, mx: 1 }} disabled />  
 </ContentContiner>  
 </Grid>  
 )  
 })  
 }  
 </Grid>  
 {!hideAddButton &&  
 <Box p={2} display="flex" justifyContent="center" alignItems="center">  
 <AddCircleOutlinedIcon  
 sx={{ fontSize: 80, cursor: 'pointer' }}  
 onClick={() => append({  
 submission: null,  
 luReply: ""  
 })}  
 />  
 </Box>  
 }  
 </Container>  
 )  
}  
export default FolioHistoryTable;

## File: bd-scs-backend-web/src/components/Case/FormHeader.jsx

import { Controller } from 'react-hook-form'  
import Grid from '@mui/material/Grid'  
import FormControl from 'components/basic/FormControl'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import TextField from 'components/basic/TextField'  
function FormHeader({ isDesktop, control, t }) {  
 return (  
 <>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={10} pb={4}>  
 <Grid item xs>  
 <Controller  
 name="applicationNumber"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicationNumber', {  
 ns: 'commonTask',  
 })}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="applicationType"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicationType', {  
 ns: 'commonTask',  
 })}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={10} pb={4}>  
 <Grid item xs>  
 <Controller  
 name="caseNumber"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('caseNumber', { ns: 'commonTask' })}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="submissionType"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('sunmissionType', {  
 ns: 'commonTask',  
 })}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </>  
 )  
}  
export default FormHeader

## File: bd-scs-backend-web/src/components/Case/P1.jsx

import { useTranslation } from 'react-i18next'  
import dayjs from 'dayjs'  
import { Controller } from 'react-hook-form'  
import { useParams } from 'react-router-dom'  
import { useQuery } from '@tanstack/react-query'  
import React from 'react'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Select from '@mui/material/Select'  
import MenuItem from '@mui/material/MenuItem'  
import GradientButton from 'components/basic/Button/GradientButton'  
import DataTable from 'components/basic/DataGrid'  
import FormControl from 'components/basic/FormControl'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import TextField from 'components/basic/TextField'  
import Typography from 'components/basic/Typography'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable'  
import CommonFormHeader from 'components/Case/FormHeader'  
import KeyboardArrowDownIcon from '@mui/icons-material/KeyboardArrowDown'  
import { getUsers } from 'apis/user'  
import { userContext } from 'src/context/UserProvider'  
import DocUploader from 'components/basic/DocUploader'  
import { TaskTitle } from 'src/constants/tasks'  
function P1Task({ control, onSubmit, setValue }) {  
 return (  
 <Stack direction="column">  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Box px={4} py={3}>  
 <Grid container direction="column" sx={{ pb: 2 }}>  
 <Typography  
 variant="title2"  
 color="text.secondary"  
 >{`${TaskTitle['INITIAL\_SITE\_INSPECTION']}`}</Typography>  
 </Grid>  
 <DocUploader  
 control={control}  
 label="Upload Documents"  
 name="initialSiteInspection"  
 limit={1}  
 setValue={setValue}  
 hideDocType  
 />  
 </Box>  
 <GradientButton sx={{ width: 100, alignSelf: 'flex-end', mr: 2 }} onClick={onSubmit}>Save</GradientButton>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default P1Task

## File: bd-scs-backend-web/src/components/Case/P2.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled'  
import { Controller, useWatch } from 'react-hook-form'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import Grid from '@mui/material/Grid'  
import Button from 'components/basic/Button'  
import RadioGroup from '@mui/material/RadioGroup'  
import FormGroup from '@mui/material/FormGroup'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import InputAdornment from '@mui/material/InputAdornment'  
import Radio from '@mui/material/Radio'  
import Checkbox from 'components/basic/Checkbox'  
import RadioCheckbox from 'components/Case/RadioCheckbox'  
import Typography from 'components/basic/Typography'  
import TextField from 'components/basic/TextField'  
import GradientButton from 'components/basic/Button/GradientButton'  
import FormControl from 'components/basic/FormControl'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable'  
import SubmissionTable from 'components/Case/SubmissionTable'  
import DataTable from 'components/basic/DataGrid'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import CommonFormHeader from 'components/Case/FormHeader'  
import QuestionHeader from 'components/Case/QuestionHeader'  
import CAT\_DESCRIPTIONS from 'src/config/cat';  
const committeeCaseRegisterUrl =  
 'https://sp.edkms.bd.hksarg/sites/BD/FC/Forms/Document%20Set/docsethomepage.aspx?ID=1735&FolderCTID=0x0120D52000CFEF80D851BFE043B759D02D2F95AD2A&List=6007db9f-0976-49b4-a0e6-5b58ebc7e782&RootFolder=%2Fsites%2FBD%2FFC%2FMeeting%20Notes%2FFS%20committee%2FFSC%20Case%20Register%20%281997%2D2024%29&RecSrc=%2Fsites%2FBD%2FFC%2FMeeting%20Notes%2FFS%20committee%2FFSC%20Case%20Register%20%281997%2D2024%29'  
function P2Task({  
 control,  
 onSubmit,  
 setValue,  
 getValues,  
 openBuildingInformationDialog,  
 openDvTableDialog,  
}) {  
 const q4Ans = useWatch({ control, name: 'q4.answer' })  
 const q5Ans = useWatch({ control, name: 'q5.answer' })  
 let optionsCat23a = `Aleration and Addition Works under BD ref.`;  
 let answerCat1 = `A revised application form with correct address should be submitted.`;  
 if(getValues('q4.options\_cat2\_3a\_text')) {  
 Object.keys(getValues('q4.options\_cat2\_3a\_text')).forEach((key) => {  
 if(key === 'a') optionsCat23a = getValues('q4.options\_cat2\_3a\_text')[key];  
 });  
 }  
 if(getValues('q5.answer\_cat1\_text')) {  
 answerCat1 = getValues('q5.answer\_cat1\_text');  
 }  
 return (  
 <Stack direction="column">  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Box px={4} py={3}>  
 <Stack direction="row" gap={2} alignItems="center">  
 <Typography variant="title2" sx={{ flex: 1 }}>  
 Desk study  
 </Typography>  
 <Button onClick={openBuildingInformationDialog}>  
 Building Information  
 </Button>  
 {  
}  
 </Stack>  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1. Background`}  
 control={control}  
 />  
 <FolioHistoryTable control={control} name="q1." />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`2. Any OP, plans / structural details available?`}  
 control={control}  
 needAnswer  
 ansFieldName="q2.answer"  
 />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`3. Submission for Alterations and Additions Works`}  
 />  
 <SubmissionTable control={control} name="q3." />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`4. Any A&A Works related to this application on going?`}  
 control={control}  
 needAnswer  
 ansFieldName="q4.answer"  
 onChange={() => {  
 setValue('q4.options\_cat2\_3a', [])  
 setValue('q4.value\_cat2\_3a', '')  
 }}  
 />  
 <Box sx={{ backgroundColor: '#fff', p: 2 }}>  
 <Typography  
 mt={2}  
 >{`[Cat.2(iii)(a)]`}</Typography>  
 {q4Ans === true ? (  
 <Controller  
 name="q4.options\_cat2\_3a"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => {  
 return (  
 <FormGroup sx={{ mt: 2 }}>  
 <Grid  
 container  
 direction="row"  
 >  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'a'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'a',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'a'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={  
 <span style={{backgroundColor: (optionsCat23a === `Aleration and Addition Works under BD ref.` ? '' : 'yellow')}}>  
 {optionsCat23a}  
 </span>}  
 />  
 <Controller  
 name="q4.value\_cat2\_3a"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'a'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )  
 }}  
 />  
 ) : (  
 <Typography sx={{ mt: 2 }}>NIL</Typography>  
 )}  
 </Box>  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`5. Address of permises correctly descripted`}  
 control={control}  
 needAnswer  
 ansFieldName="q5.answer"  
 onChange={() => {  
 setValue('q5.answer\_cat1', false)  
 }}  
 />  
 {q5Ans === 'false' && (  
 <Box sx={{ backgroundColor: '#fff', p: 2 }}>  
 <Typography>{`(Logical question, appear if select "No")`}</Typography>  
 <Typography mt={2}>{`[Cat.1]`}</Typography>  
 <Controller  
 name="q5.answer\_cat1"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => {  
 return (  
 <FormGroup sx={{ mt: 2 }}>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={  
 field.value  
 }  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e.target  
 .checked  
 field.onChange(  
 isChecked  
 )  
 }}  
 />  
 }  
 label={  
 <span style={{backgroundColor: (answerCat1 === `A revised application form with correct address should be submitted.` ? '' : 'yellow')}}>  
 {answerCat1}  
 </span>  
 }  
 sx={{ mb: 2 }}  
 />  
 </FormGroup>  
 )  
 }}  
 />  
 </Box>  
 )}  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`6. The fire safety provision was designed by fire engineering approach ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q6.answer"  
 />  
 <Button  
 onClick={() =>  
 window.open(committeeCaseRegisterUrl)  
 }  
 >  
 FS committee case register  
 </Button>  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`7. A cinema exists in the building ?`}  
 control={control}  
 needAnswer  
 ansFieldName="q7.answer"  
 />  
 </Box>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`8. The height of the proposed school above Ground Level`}  
 control={control}  
 />  
 <Box sx={{ backgroundColor: '#fff', p: 2, mt: 2 }}>  
 <Controller  
 name="q8.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 InputProps={{  
 endAdornment: (  
 <InputAdornment position="end">  
 m  
 </InputAdornment>  
 ),  
 }}  
 {...field}  
 />  
 )}  
 />  
 </Box>  
 </Box>  
 <GradientButton  
 sx={{  
 width: 'fit-content',  
 mr: 4,  
 alignSelf: 'flex-end',  
 }}  
 onClick={onSubmit}  
 >  
 Save  
 </GradientButton>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default P2Task

## File: bd-scs-backend-web/src/components/Case/P7.jsx

import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import { Controller, useWatch } from 'react-hook-form'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import Grid from '@mui/material/Grid'  
import FormGroup from '@mui/material/FormGroup'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import Checkbox from 'components/basic/Checkbox'  
import RadioCheckbox from 'components/Case/RadioCheckbox'  
import TextField from 'components/basic/TextField'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import Typography from 'components/basic/Typography'  
import GradientButton from 'components/basic/Button/GradientButton'  
import CommonFormHeader from 'components/Case/FormHeader'  
import QuestionHeader from 'components/Case/QuestionHeader'  
const options\_cat1 = [  
 {  
 label: 'The application date shown on the plan should be DATE’ in accordance with the submitted Form A2.',  
 value: 'a',  
 },  
]  
function P7Task({ control, setValue, onSubmit }) {  
 const q1Ans = useWatch({ control, name: 'q1.answer' })  
 return (  
 <Stack direction="column">  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Typography variant="title2" mx={3}>  
 {'MISC'}  
 </Typography>  
 <Box px={4} py={3}>  
 <QuestionHeader  
 question={`1. Other general conditions`}  
 control={control}  
 needAnswer  
 ansFieldName="q1.answer"  
 onChange={() => {  
 setValue('q1.cat1', [])  
 setValue('q1.cat1\_1', [])  
 setValue('q1.cat1\_1\_other', '')  
 setValue('q1.cat1\_other', '')  
 setValue('q1.cat2\_1a', [])  
 setValue('q1.cat2\_1a\_other', '')  
 setValue('q1.cat2\_1b', [])  
 setValue('q1.cat2\_1b\_other', '')  
 setValue('q1.cat2\_2a', [])  
 setValue('q1.cat2\_2a\_other', '')  
 setValue('q1.cat2\_3b', [])  
 setValue('q1.cat2\_3b\_other', '')  
 setValue('q1.cat2\_4', [])  
 setValue('q1.cat2\_4\_other', '')  
 setValue('q1.cat3', [])  
 setValue('q1.cat3\_other', '')  
 }}  
 />  
 {q1Ans === 'true' && (  
 <>  
 <Box sx={{ backgroundColor: '#fff', p: 2 }}>  
 <Typography>{`[Cat. 1]`}</Typography>  
 <Typography>{`4 sets of duly signed revised plans to incorporate the following matters should be submitted:`}</Typography>  
 <Controller  
 name="q1.cat1"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup  
 sx={{ mt: 1, ml: 4 }}  
 >  
 {options\_cat1.map(  
 (item) => {  
 return (  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 item.value  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 item.value,  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 item.value  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={  
 item.label  
 }  
 />  
 )  
 }  
 )}  
 <Grid  
 container  
 direction="row"  
 >  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.cat1\_other"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{`A revised application form with correct address should be submitted.`}</Typography>  
 <Controller  
 name="q1.cat1\_1"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup  
 sx={{ mt: 1, ml: 4 }}  
 >  
 <Grid  
 container  
 direction="row"  
 >  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.cat1\_1\_other"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Box  
 sx={{  
 backgroundColor: '#fff',  
 p: 2,  
 mt: 2,  
 }}  
 >  
 <Typography>{`[Cat. 2 (i)(a)]`}</Typography>  
 <Controller  
 name="q1.cat2\_1a"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup  
 sx={{ mt: 1, ml: 4 }}  
 >  
 <Grid  
 container  
 direction="row"  
 >  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.cat2\_1a\_other"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{`[Cat. 2 (i)(b)]`}</Typography>  
 <Controller  
 name="q1.cat2\_1b"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup  
 sx={{ mt: 1, ml: 4 }}  
 >  
 <Grid  
 container  
 direction="row"  
 >  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.cat2\_1b\_other"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{`[Cat. 2 (ii)(a)]`}</Typography>  
 <Controller  
 name="q1.cat2\_2a"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup  
 sx={{ mt: 1, ml: 4 }}  
 >  
 <Grid  
 container  
 direction="row"  
 >  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.cat2\_2a\_other"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{`[Cat. 2 (iii)(b)]`}</Typography>  
 <Controller  
 name="q1.cat2\_3b"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup  
 sx={{ mt: 1, ml: 4 }}  
 >  
 <Grid  
 container  
 direction="row"  
 >  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.cat2\_3b\_other"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 <Typography>{`[Cat. 2 (iv)]`}</Typography>  
 <Controller  
 name="q1.cat2\_4"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup  
 sx={{ mt: 1, ml: 4 }}  
 >  
 <Grid  
 container  
 direction="row"  
 >  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.cat2\_4\_other"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 <Box  
 sx={{  
 backgroundColor: '#fff',  
 p: 2,  
 mt: 2,  
 }}  
 >  
 <Typography>{`[Cat. 3]`}</Typography>  
 <Controller  
 name="q1.cat3"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormGroup  
 sx={{ mt: 1, ml: 4 }}  
 >  
 <Grid  
 container  
 direction="row"  
 >  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={field.value?.includes(  
 'other'  
 )}  
 onChange={(  
 e  
 ) => {  
 const isChecked =  
 e  
 .target  
 .checked  
 if (  
 isChecked  
 ) {  
 field.onChange(  
 [  
 ...field.value,  
 'other',  
 ]  
 )  
 } else {  
 field.onChange(  
 field.value.filter(  
 (  
 v  
 ) =>  
 v !==  
 'other'  
 )  
 )  
 }  
 }}  
 />  
 }  
 label={`Other`}  
 />  
 <Controller  
 name="q1.cat3\_other"  
 control={control}  
 render={({  
 field: otherField,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 {...otherField}  
 disabled={  
 !field.value?.includes(  
 'other'  
 )  
 }  
 />  
 )}  
 />  
 </Grid>  
 </FormGroup>  
 )}  
 />  
 </Box>  
 </>  
 )}  
 </Box>  
 <GradientButton  
 sx={{ width: 100, alignSelf: 'flex-end', mr: 2 }}  
 onClick={onSubmit}  
 >  
 Save  
 </GradientButton>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default P7Task

## File: bd-scs-backend-web/src/components/Case/P8.jsx

import { useTranslation } from 'react-i18next'  
import dayjs from 'dayjs'  
import { Controller } from 'react-hook-form'  
import { useParams } from 'react-router-dom'  
import { useQuery, useQueryClient } from '@tanstack/react-query'  
import React from 'react'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Select from '@mui/material/Select'  
import MenuItem from '@mui/material/MenuItem'  
import GradientButton from 'components/basic/Button/GradientButton'  
import DataTable from 'components/basic/DataGrid'  
import FormControl from 'components/basic/FormControl'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import TextField from 'components/basic/TextField'  
import Typography from 'components/basic/Typography'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import FolioHistoryTable from 'components/Case/FolioHistoryTable'  
import CommonFormHeader from 'components/Case/FormHeader'  
import KeyboardArrowDownIcon from '@mui/icons-material/KeyboardArrowDown'  
import { getUsers } from 'apis/user'  
import { userContext } from 'src/context/UserProvider'  
import DocUploader from 'components/basic/DocUploader'  
import { TaskTitle } from 'src/constants/tasks'  
import { LETTERS, SELFCERT\_COMPLETE\_LETTERS, SELFCERT\_FAIL\_LETTERS } from 'src/constants/letters'  
import { Button } from '@mui/material'  
import constants from 'src/constants'  
import AttachmentTable from 'components/AttachmentTable'  
function P8Task({  
 applicationId,  
 caseId,  
 control,  
 onSubmit,  
 setValue,  
 isScs,  
 isScsSuccess,  
 attachments,  
}) {  
 const queryClient = useQueryClient()  
 console.log('Attachments', attachments)  
 let letters = LETTERS;  
 if(isScs){  
 letters = SELFCERT\_COMPLETE\_LETTERS;  
 }  
 return (  
 <Stack direction="column">  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 {Object.keys(letters).map((key) => {  
 return (  
 <Stack direction="column" px={4} py={3}>  
 <Typography  
 variant="title2"  
 color="text.secondary"  
 >{`${letters[key].name}`}</Typography>  
 <a  
 variant="contained"  
 component="label"  
 href={`${constants.apiEndPoint}cases/${caseId}/letterTemplate?type=${key}`}  
 download  
 >  
 Download Template  
 </a>  
 <AttachmentTable  
 data={attachments?.filter((item) => item.subType == key)}  
 applicationId={applicationId}  
 submissionCaseId={caseId}  
 onSuccess={(caseId) => {  
 queryClient.invalidateQueries([  
 'fetchCaseDetail',  
 { caseId },  
 ])  
 }}  
 onDialogClose={() => {  
 queryClient.invalidateQueries([  
 'fetchCaseDetail',  
 { caseId },  
 ])  
 }}  
 disabled  
 type="PREPARE\_LETTER"  
 subType={key}  
 max={1}  
 min={1}  
 hide={{  
 filePartNo: true,  
 }}  
 />  
 {  
}  
 </Stack>  
 )  
 })}  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default P8Task

## File: bd-scs-backend-web/src/components/Case/QuestionHeader.jsx

import { Controller } from 'react-hook-form';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Grid from '@mui/material/Grid';  
import Radio from '@mui/material/Radio';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormControl from 'components/basic/FormControl';  
import Typography from 'components/basic/Typography';  
function QuestionHeader({ question, questionProps, control, needAnswer, ansFieldName, onChange, disabled }) {  
 return (  
 <Grid container direction="row" alignItems="center" justifyContent="space-between" mt={2}>  
 <Grid item>  
 <Typography {...questionProps}>{question}</Typography>  
 </Grid>  
 {needAnswer &&  
 <Grid item>  
 <Controller  
 name={ansFieldName}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value);  
 onChange && onChange(e.target.value);  
 }}  
 >  
 <FormControlLabel value={true} control={<Radio disabled={disabled} />} label={`Yes`} />  
 <FormControlLabel value={false} control={<Radio disabled={disabled} />} label={`No`} />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>}  
 </Grid>  
 )  
}  
export default QuestionHeader;

## File: bd-scs-backend-web/src/components/Case/RadioCheckbox.jsx

import Checkbox from 'components/basic/Checkbox';  
import RadioButtonUncheckedOutlinedIcon from '@mui/icons-material/RadioButtonUncheckedOutlined';  
import RadioButtonCheckedOutlinedIcon from '@mui/icons-material/RadioButtonCheckedOutlined';  
const RadioCheckbox = (props) => {  
 return (  
 <Checkbox  
 icon={<RadioButtonUncheckedOutlinedIcon />}  
 checkedIcon={<RadioButtonCheckedOutlinedIcon />}  
 {...props}  
 />  
 )  
}  
export default RadioCheckbox;

## File: bd-scs-backend-web/src/components/Case/ReminderReceivedTable.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled'  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Typography from 'components/basic/Typography'  
import TextField from 'components/basic/TextField'  
import Button from 'components/basic/Button'  
import DeleteOutlineIcon from '@mui/icons-material/DeleteOutline'  
import colors from 'src/constants/colors'  
const Container = styled(Box)`  
 display: flex;  
 flex-direction: column;  
 flex-wrap: nowrap;  
`  
const Header = styled(Grid)`  
 width: ${(props) => (props.w ? `${props.w}px` : 'none')};  
 height: 60px;  
 color: #fff;  
`  
const ContentContiner = styled(Grid)`  
 height: 60px;  
 align-items: center;  
 margin-top: 0.5px;  
`  
function ReminderReceivedTable({ control, t, sx, name, setValue, disabled }) {  
 const { fields, append, remove } = useFieldArray({  
 control,  
 name: `${name}`,  
 })  
 const onClickAdd = () => {  
 append({  
 No: '',  
 ReceivedDate: dayjs().format('YYYY-MM-DD'),  
 Remarks: '',  
 })  
 }  
 const onClickDel = (index) => {  
 // if (!index) return  
 remove(index)  
 }  
 return (  
 <Container sx={sx} mt={5}>  
 <Grid  
 container  
 direction="row"  
 justifyContent="space-between"  
 alignItems="center"  
 my={1}  
 >  
 <Grid item>  
 <Typography sx={{ fontWeight: 'bold' }} color="text.secondary">  
 Reminder Received:  
 </Typography>  
 </Grid>  
 {!disabled && (  
 <Grid item>  
 <Button onClick={onClickAdd}>Add</Button>  
 </Grid>  
 )}  
 </Grid>  
 <Box borderRadius={4} bgcolor="#fff">  
 <Grid  
 container  
 direction="row"  
 sx={{  
 backgroundColor: colors.main,  
 borderTopLeftRadius: 4,  
 borderTopRightRadius: 4,  
 }}  
 >  
 <Grid item>  
 <Header item container alignItems="center" px={2} w={200}>  
 <Typography>No.</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header item container alignItems="center" px={2} w={200}>  
 <Typography>Received Date</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header item container alignItems="center" px={2} xs>  
 <Typography>Remarks</Typography>  
 </Header>  
 </Grid>  
 </Grid>  
 <Grid container direction="row">  
 {fields.map((item, index) => {  
 return (  
 <Grid item container direction="row" key={`fields-${index}`}>  
 <ContentContiner item container px={2} sx={{ width: 200 }}>  
 <Controller  
 name={`${name}.${index}.No.`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} disabled={disabled} />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} sx={{ width: 200 }}>  
 <Controller  
 name={`${name}.${index}.ReceivedDate`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} disabled={disabled} />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container px={2} xs>  
 <Controller  
 name={`${name}.${index}.Remarks`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} fullWidth disabled={disabled} />  
 )}  
 />  
 </ContentContiner>  
 {!disabled && (  
 <Box  
 display="flex"  
 alignItems="center"  
 justifyContent="center"  
 sx={{  
 cursor: 'pointer',  
 mx: 1,  
 }}  
 onClick={() => onClickDel(index)}  
 >  
 <DeleteOutlineIcon  
 sx={{  
 color: 'red',  
 }}  
 />  
 </Box>  
 )}  
 </Grid>  
 )  
 })}  
 </Grid>  
 </Box>  
 </Container>  
 )  
}  
export default ReminderReceivedTable

## File: bd-scs-backend-web/src/components/Case/SubmissionTable.jsx

import { useTranslation } from 'react-i18next'  
import styled from '@emotion/styled';  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import dayjs from 'dayjs';  
import Box from '@mui/material/Box';  
import Stack from '@mui/material/Stack';  
import Grid from '@mui/material/Grid';  
import Divider from '@mui/material/Divider';  
import IconButton from '@mui/material/IconButton';  
import Typography from 'components/basic/Typography';  
import TextField from 'components/basic/TextField';  
import DatePicker from 'components/basic/DatePicker';  
import GradientButton from 'components/basic/Button/GradientButton';  
import FormControl from 'components/basic/FormControl';  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import AddCircleOutlinedIcon from '@mui/icons-material/AddCircleOutlined';  
const Container = styled(Box)`  
 border: 2px solid #00838f;  
 border-radius: 5px;  
 display: flex;  
 flex-direction: row;  
 background-color: #fff;  
`  
const Header = styled(Grid)`  
 width: 220px;  
 height: 60px;  
 background-color: #e0f7fa;  
 border-top-left-radius: 5px;  
 border-bottom-left-radius: 5px;  
`  
const ContentContiner = styled(Grid)`  
 width: 240px;  
 height: 60px;  
 align-items: center;  
 justify-content: center;  
 border-bottom: ${props => props.hideBorder ? 0 : 1}px solid #9e9e9e;  
 margin-top: 0.5px;  
`  
function SubmissionTable({ control, hideAddButton, sx, name }) {  
 const { fields, append } = useFieldArray({  
 control,  
 name: `${name}value`  
 });  
 return (  
 <Container sx={sx}>  
 <Grid container direction="row" flexWrap="nowrap" sx={{ width: 'fit-content' }}>  
 <Grid item container direction="column" sx={{ width: 'fit-content' }}>  
 <Grid item sx={{ borderBottom: `1px solid #9e9e9e` }}>  
 <Header item container alignItems="center" px={2}>  
 <Typography>{`File Ref`}</Typography>  
 </Header>  
 </Grid>  
 <Grid item sx={{ borderBottom: `1px solid #9e9e9e` }}>  
 <Header item container alignItems="center" px={2}>  
 <Typography>{`Approval date`}</Typography>  
 </Header>  
 </Grid>  
 <Grid item>  
 <Header item container alignItems="center" px={2}>  
 <Typography>{`Acknowledgement date`}</Typography>  
 </Header>  
 </Grid>  
 </Grid>  
 </Grid>  
 <Grid container direction="row" flexWrap="nowrap" sx={{ overflow: 'auto' }}>  
 {  
 fields.map((item, index) => {  
 return (  
 <Grid item container direction="column" key={`fields-${index}`} sx={{ width: 'fit-content'}} >  
 <ContentContiner item container sx={{ px: 2 }}>  
 <Controller  
 name={`${name}value.${index}.fileRef`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} sx={{ width: 200 }} />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container sx={{ px: 2 }}>  
 <Controller  
 name={`${name}value.${index}.approvedDate`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <DatePicker {...field} sx={{ width: 200 }} />  
 )}  
 />  
 </ContentContiner>  
 <ContentContiner item container hideBorder sx={{ px: 2 }}>  
 <Controller  
 name={`${name}value.${index}.acknowledgementDate`}  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <DatePicker {...field} sx={{ width: 200 }} />  
 )}  
 />  
 </ContentContiner>  
 </Grid>  
 )  
 })  
 }  
 </Grid>  
 {!hideAddButton &&  
 <Box p={2} display="flex" justifyContent="center" alignItems="center">  
 <AddCircleOutlinedIcon  
 sx={{ fontSize: 80, cursor: 'pointer' }}  
 onClick={() => append({  
 fileRef: "",  
 approvedDate: null,  
 acknowledgementDate: null,  
 })}  
 />  
 </Box>  
 }  
 </Container>  
 )  
}  
export default SubmissionTable;

## File: bd-scs-backend-web/src/components/Form/Type/a2.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import Stack from '@mui/material/Stack'  
import Divider from '@mui/material/Divider'  
import Grid from '@mui/material/Grid'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import Checkbox from 'components/basic/Checkbox'  
import Typography from 'src/components/basic/Typography'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import DelegateForm from 'components/Form/delegateForm'  
import BasicApplicantForm from 'components/Form/basicApplicantForm'  
import ContactForm2 from 'components/Form/contactForm2'  
import AddressForm from 'components/Form/addressForm'  
import SchoolForm from 'components/Form/schoolForm'  
import DocUploader from 'components/basic/DocUploader'  
function A2Form({ control, setValue = { setValue }, readOnly = false }) {  
 const { t } = useTranslation(['form'])  
 const [open, setOpen] = React.useState(false)  
 const watchApplicantName = useWatch({ control, name: 'ApplicantName' })  
 const watchApplicantEmail = useWatch({ control, name: 'ApplicantEmail' })  
 const watchApplicantMobile = useWatch({  
 control,  
 name: 'ApplicantMobile',  
 })  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <BasicApplicantForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <Grid container mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ApplicantAddress"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('address')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 <Grid container mb={4}>  
 <Grid item xs>  
 <FormControlLabel  
 sx={{  
 color: 'text.secondary',  
 '& .MuiTypography-root': {  
 fontWeight: 600,  
 },  
 }}  
 control={<Checkbox disabled={readOnly} />}  
 label={t('sameAsApplicant')}  
 onChange={(e) => {  
 const checked = e.target.checked  
 if (checked) {  
 setValue(  
 'ContactPName',  
 watchApplicantName  
 )  
 setValue(  
 'ContactPEmail',  
 watchApplicantEmail  
 )  
 setValue(  
 'ContactPMobile',  
 watchApplicantMobile  
 )  
 } else {  
 setValue('ContactPName', '')  
 setValue('ContactPEmail', '')  
 setValue('ContactPMobile', '')  
 }  
 }}  
 />  
 </Grid>  
 </Grid>  
 <ContactForm2  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <Typography  
 variant="title3"  
 color="text.secondary"  
 sx={{ mb: 2, mt: 3 }}  
 >  
 {t('locationSubmitInfo')}  
 </Typography>  
 <AddressForm  
 control={control}  
 isDesktop={isDesktop}  
 openDialog={setOpen}  
 readOnly={readOnly}  
 />  
 <SchoolForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <DelegateForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 {!readOnly && <DocUploader control={control} />}  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 }  
 }}  
 />  
 </>  
 )  
}  
export default A2Form

## File: bd-scs-backend-web/src/components/Form/Type/appointAp.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import Stack from '@mui/material/Stack'  
import Divider from '@mui/material/Divider'  
import Grid from '@mui/material/Grid'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import Checkbox from 'components/basic/Checkbox'  
import Typography from 'src/components/basic/Typography'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import DelegateForm from 'components/Form/delegateForm'  
import BasicApplicantForm from 'components/Form/basicApplicantForm'  
import ContactForm2 from 'components/Form/contactForm2'  
import AddressForm from 'components/Form/addressForm'  
import SchoolForm from 'components/Form/schoolForm'  
import DocUploader from 'components/basic/DocUploader'  
function AppointApForm({ control, setValue = { setValue }, readOnly = false }) {  
 const { t } = useTranslation(['form'])  
 const [open, setOpen] = React.useState(false)  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <DelegateForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 {}  
 {}  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 }}  
 />  
 </>  
 )  
}  
export default AppointApForm

## File: bd-scs-backend-web/src/components/Form/Type/b2.jsx

import React from 'react'  
import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import FormHelperText from '@mui/material/FormHelperText'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import DatePicker from 'src/components/basic/DatePicker'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Typography from 'src/components/basic/Typography'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import AddressForm from 'components/Form/addressForm'  
import BasicApplicantForm from 'components/Form/basicApplicantForm'  
import PlusIcon from 'assets/icon-plus.png'  
import FolderIcon from 'assets/icon-folder-open.png'  
import FileIcon from 'assets/icon-file.png'  
import DocUploader from 'components/basic/DocUploader'  
function B2Form({ control, setValue = { setValue }, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 const [open, setOpen] = React.useState(false)  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs={isDesktop ? 8 : true}>  
 <Controller  
 name="AmendmentOfLayoutPlan"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'amendmentOfLayoutPlanAsAdvisedBy'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 disabled={true}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicationDate"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('date')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <DatePicker  
 disabled={readOnly}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {  
}  
 {  
}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="BDLetterRefDate"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'buildingsDepartmentLetterDate'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <DatePicker  
 disabled={readOnly}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs mb={3}>  
 <Controller  
 name="BdRefNo"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'buildingsDepartmentLetterFileNumber'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 disabled={readOnly}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {!readOnly && <DocUploader control={control} />}  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 }  
 }}  
 />  
 </>  
 )  
}  
export default B2Form

## File: bd-scs-backend-web/src/components/Form/Type/c2.jsx

import React from 'react'  
import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import FormHelperText from '@mui/material/FormHelperText'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import DatePicker from 'src/components/basic/DatePicker'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Typography from 'src/components/basic/Typography'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import AddressForm from 'components/Form/addressForm'  
import BasicApplicantForm from 'components/Form/basicApplicantForm'  
import PlusIcon from 'assets/icon-plus.png'  
import FolderIcon from 'assets/icon-folder-open.png'  
import FileIcon from 'assets/icon-file.png'  
import DocUploader from 'components/basic/DocUploader'  
function C2Form({ control, setValue = { setValue }, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 const [open, setOpen] = React.useState(false)  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs={isDesktop ? 8 : true}>  
 <Controller  
 name="AmendmentOfLayoutPlan"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'amendmentOfLayoutPlanAsAdvisedBy'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 disabled={true}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicationDate"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('date')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <DatePicker  
 disabled={readOnly}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {  
}  
 {  
}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="BDLetterRefDate"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'buildingsDepartmentLetterDate'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <DatePicker  
 disabled={readOnly}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {!readOnly && <DocUploader control={control} />}  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 }}  
 />  
 </>  
 )  
}  
export default C2Form

## File: bd-scs-backend-web/src/components/Form/Type/cccaic2.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import AgencyForm from 'components/Form/agencyForm'  
import ApplicantForm from 'components/Form/applicantForm'  
import ContactForm from 'components/Form/contactForm'  
import DelegateForm from 'components/Form/delegateForm'  
import ProposedCenterInfoForm from 'components/Form/proposedCenterInfoForm'  
function Cccaic2Form({ control, setValue, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 const [open, setOpen] = React.useState(false)  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <ApplicantForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <AgencyForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <Divider />  
 <ProposedCenterInfoForm  
 control={control}  
 isDesktop={isDesktop}  
 openDialog={() => setOpen(true)}  
 readOnly={readOnly}  
 />  
 <Divider />  
 <ContactForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <DelegateForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 }  
 }}  
 />  
 </>  
 )  
}  
export default Cccaic2Form

## File: bd-scs-backend-web/src/components/Form/Type/cccaimh3.jsx

import React from 'react'  
import { Controller } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Divider from '@mui/material/Divider'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Autocomplete from 'components/basic/Autocomplete'  
import GradientButton from 'components/basic/Button/GradientButton'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import DatePicker from 'src/components/basic/DatePicker'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import ContactForm from 'components/Form/contactForm'  
import DelegateForm from 'components/Form/delegateForm'  
import ProposedCenterInfoForm from 'components/Form/proposedCenterInfoForm'  
function Cccaimh3Form({ control, setValue, readOnly = false }) {  
 const { t, i18n } = useTranslation(['cccaimh3', 'form'])  
 const [open, setOpen] = React.useState(false)  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="ApplicantTitle"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('title', { ns: 'form' })}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.id?.message}  
 fullWidth  
 required  
 >  
 <Autocomplete  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect',  
 { ns: 'form' }  
 ),  
 id: '',  
 },  
 {  
 label: t(  
 'titleOptions.mr',  
 { ns: 'form' }  
 ),  
 id: 'mr',  
 },  
 {  
 label: t(  
 'titleOptions.mrs',  
 { ns: 'form' }  
 ),  
 id: 'mrs',  
 },  
 {  
 label: t(  
 'titleOptions.miss',  
 { ns: 'form' }  
 ),  
 id: 'miss',  
 },  
 {  
 label: t(  
 'titleOptions.ms',  
 { ns: 'form' }  
 ),  
 id: 'ms',  
 },  
 ]}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disableClearable  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantNameEN"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'representativeFullNameEng'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantName"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'representativeFullNameChi'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="ApplicantEmail"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('representativeEmail')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField  
 {...field}  
 type="email"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantMobile"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('representativePhone')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField  
 {...field}  
 type="tel"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantHKIC"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('representativeIdNo')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="ApplicantPosition"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('representativePosition')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicationDate"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('date', { ns: 'form' })}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <DatePicker  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {isDesktop && <Grid item xs />}  
 </ResponsiveGrid>  
 <Grid item xs mb={4}>  
 <Controller  
 name="ApplicantRepOrgEN"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('organiserNameEng')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs mb={4}>  
 <Controller  
 name="ApplicantRepOrg"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('organiserNameChi')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs mb={4}>  
 <Controller  
 name="ApplicantAddressEN"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('organiserAddressEng')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs mb={4}>  
 <Controller  
 name="ApplicantAddress"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('organiserAddressChi')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={6} mb={4}>  
 <Controller  
 name="ApplicantTel"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('organiserPhone')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="tel"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <ProposedCenterInfoForm  
 control={control}  
 isDesktop={isDesktop}  
 openDialog={() => setOpen(true)}  
 readOnly={readOnly}  
 />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <ContactForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <DelegateForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 }  
 }}  
 />  
 </>  
 )  
}  
export default Cccaimh3Form

## File: bd-scs-backend-web/src/components/Form/Type/e2.jsx

import React from 'react'  
import { Controller, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Divider from '@mui/material/Divider'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Checkbox from 'components/basic/Checkbox'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Typography from 'src/components/basic/Typography'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import AddressForm from 'components/Form/addressForm'  
import BasicApplicantForm from 'components/Form/basicApplicantForm'  
import ContactForm2 from 'components/Form/contactForm2'  
import DelegateForm from 'components/Form/delegateForm'  
import SchoolForm from 'components/Form/schoolForm'  
function E2Form({ control, setValue, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 const [open, setOpen] = React.useState(false)  
 const watchApplicantName = useWatch({ control, name: 'ApplicantName' })  
 const watchApplicantEmail = useWatch({ control, name: 'ApplicantEmail' })  
 const watchApplicantMobile = useWatch({  
 control,  
 name: 'ApplicantMobile',  
 })  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <BasicApplicantForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <Grid container mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ApplicantAddress"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('address')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 <Grid container mb={4}>  
 <Grid item xs>  
 <FormControlLabel  
 sx={{  
 color: 'text.secondary',  
 '& .MuiTypography-root': {  
 fontWeight: 600,  
 },  
 }}  
 control={<Checkbox disabled={readOnly} />}  
 label={t('sameAsApplicant')}  
 onChange={(e) => {  
 const checked = e.target.checked  
 if (checked) {  
 setValue(  
 'ContactPName',  
 watchApplicantName  
 )  
 setValue(  
 'ContactPEmail',  
 watchApplicantEmail  
 )  
 setValue(  
 'ContactPMobile',  
 watchApplicantMobile  
 )  
 } else {  
 setValue('ContactPName', '')  
 setValue('ContactPEmail', '')  
 setValue('ContactPMobile', '')  
 }  
 }}  
 />  
 </Grid>  
 </Grid>  
 <ContactForm2  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <Typography  
 variant="title3"  
 color="text.secondary"  
 sx={{ mb: 2, mt: 3 }}  
 >  
 {t('locationSubmitInfo')}  
 </Typography>  
 <AddressForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 openDialog={setOpen}  
 />  
 <SchoolForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <DelegateForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 }  
 }}  
 />  
 </>  
 )  
}  
export default E2Form

## File: bd-scs-backend-web/src/components/Form/Type/f2.jsx

import React from 'react'  
import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import FormHelperText from '@mui/material/FormHelperText'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import DatePicker from 'src/components/basic/DatePicker'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Typography from 'src/components/basic/Typography'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import AddressForm from 'components/Form/addressForm'  
import BasicApplicantForm from 'components/Form/basicApplicantForm'  
import PlusIcon from 'assets/icon-plus.png'  
import FolderIcon from 'assets/icon-folder-open.png'  
import FileIcon from 'assets/icon-file.png'  
import DocUploader from 'components/basic/DocUploader'  
function F2Form({ control, setValue = { setValue }, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 const [open, setOpen] = React.useState(false)  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs={isDesktop ? 8 : true}>  
 <Controller  
 name="AmendmentOfLayoutPlan"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'amendmentOfLayoutPlanAsAdvisedBy'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 disabled={true}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicationDate"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('date')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <DatePicker  
 disabled={readOnly}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {  
}  
 {  
}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="BDLetterRefDate"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'buildingsDepartmentLetterDate'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <DatePicker  
 disabled={readOnly}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs mb={3}>  
 <Controller  
 name="BdRefNo"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'buildingsDepartmentLetterFileNumber'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 disabled={readOnly}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {!readOnly && <DocUploader control={control} />}  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 }}  
 />  
 </>  
 )  
}  
export default F2Form

## File: bd-scs-backend-web/src/components/Form/Type/g2.jsx

import React from 'react'  
import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import FormHelperText from '@mui/material/FormHelperText'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import DatePicker from 'src/components/basic/DatePicker'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Typography from 'src/components/basic/Typography'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import AddressForm from 'components/Form/addressForm'  
import BasicApplicantForm from 'components/Form/basicApplicantForm'  
import PlusIcon from 'assets/icon-plus.png'  
import FolderIcon from 'assets/icon-folder-open.png'  
import FileIcon from 'assets/icon-file.png'  
import DocUploader from 'components/basic/DocUploader'  
function G2Form({ control, setValue = { setValue }, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 const [open, setOpen] = React.useState(false)  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs={isDesktop ? 8 : true}>  
 <Controller  
 name="AmendmentOfLayoutPlan"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'amendmentOfLayoutPlanAsAdvisedBy'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 disabled={true}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicationDate"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('date')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <DatePicker  
 disabled={readOnly}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {  
}  
 {  
}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="BDLetterRefDate"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'buildingsDepartmentLetterDate'  
 )}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <DatePicker  
 disabled={readOnly}  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {!readOnly && <DocUploader control={control} />}  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 }}  
 />  
 </>  
 )  
}  
export default G2Form

## File: bd-scs-backend-web/src/components/Form/Type/jokc02.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import AgencyForm from 'components/Form/agencyForm'  
import ApplicantForm from 'components/Form/applicantForm'  
import ContactForm from 'components/Form/contactForm'  
import DelegateForm from 'components/Form/delegateForm'  
import ProposedCenterInfoForm from 'components/Form/proposedCenterInfoForm'  
function Jokc02Form({ control, setValue, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 const [open, setOpen] = React.useState(false)  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <ApplicantForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <AgencyForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <ProposedCenterInfoForm  
 control={control}  
 isDesktop={isDesktop}  
 openDialog={() => setOpen(true)}  
 readOnly={readOnly}  
 />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <ContactForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <DelegateForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 }  
 }}  
 />  
 </>  
 )  
}  
export default Jokc02Form

## File: bd-scs-backend-web/src/components/Form/Type/submitDoc.jsx

import React from 'react'  
import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Box from '@mui/material/Box'  
import FormHelperText from '@mui/material/FormHelperText'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import DatePicker from 'src/components/basic/DatePicker'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Typography from 'src/components/basic/Typography'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import Autocomplete from 'components/basic/Autocomplete'  
import AddressForm from 'components/Form/addressForm'  
import BasicApplicantForm from 'components/Form/basicApplicantForm'  
import ContactForm2 from 'components/Form/contactForm2'  
import DelegateForm from 'components/Form/delegateForm'  
import SquareButton from 'components/basic/Button/SquareIconButton'  
import PlusIcon from 'assets/icon-plus.png'  
import FolderIcon from 'assets/icon-folder-open.png'  
import FileIcon from 'assets/icon-file.png'  
import DocUploader from 'components/basic/DocUploader'  
function SubmitDocForm({ control, setValue = { setValue }, readOnly = false }) {  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 {  
}  
 {  
}  
 {!readOnly && <DocUploader control={control} />}  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </>  
 )  
}  
export default SubmitDocForm

## File: bd-scs-backend-web/src/components/Form/Type/updateInfo.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import Stack from '@mui/material/Stack'  
import Divider from '@mui/material/Divider'  
import Grid from '@mui/material/Grid'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import Checkbox from 'components/basic/Checkbox'  
import Typography from 'src/components/basic/Typography'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import DelegateForm from 'components/Form/delegateForm'  
import BasicApplicantForm from 'components/Form/basicApplicantForm'  
import ContactForm2 from 'components/Form/contactForm2'  
import AddressForm from 'components/Form/addressForm'  
import SchoolForm from 'components/Form/schoolForm'  
import DocUploader from 'components/basic/DocUploader'  
function UpdateInfoForm({  
 control,  
 setValue = { setValue },  
 readOnly = false,  
}) {  
 const { t } = useTranslation(['form'])  
 const [open, setOpen] = React.useState(false)  
 const watchApplicantName = useWatch({ control, name: 'ApplicantName' })  
 const watchApplicantEmail = useWatch({ control, name: 'ApplicantEmail' })  
 const watchApplicantMobile = useWatch({  
 control,  
 name: 'ApplicantMobile',  
 })  
 return (  
 <>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <BasicApplicantForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <Grid container mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ApplicantAddress"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('address')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 <Grid container mb={4}>  
 <Grid item xs>  
 <FormControlLabel  
 sx={{  
 color: 'text.secondary',  
 '& .MuiTypography-root': {  
 fontWeight: 600,  
 },  
 }}  
 control={<Checkbox disabled={readOnly} />}  
 label={t('sameAsApplicant')}  
 onChange={(e) => {  
 const checked = e.target.checked  
 if (checked) {  
 setValue(  
 'ContactPName',  
 watchApplicantName  
 )  
 setValue(  
 'ContactPEmail',  
 watchApplicantEmail  
 )  
 setValue(  
 'ContactPMobile',  
 watchApplicantMobile  
 )  
 } else {  
 setValue('ContactPName', '')  
 setValue('ContactPEmail', '')  
 setValue('ContactPMobile', '')  
 }  
 }}  
 />  
 </Grid>  
 </Grid>  
 <ContactForm2  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 <Typography  
 variant="title3"  
 color="text.secondary"  
 sx={{ mb: 2, mt: 3 }}  
 >  
 {t('locationSubmitInfo')}  
 </Typography>  
 <AddressForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 openDialog={setOpen}  
 />  
 <SchoolForm  
 control={control}  
 isDesktop={isDesktop}  
 readOnly={readOnly}  
 />  
 {}  
 {}  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <AddressDialog  
 open={open}  
 onClose={() => setOpen(false)}  
 onChange={({ address, inputManually }) => {  
 if (address) {  
 const en = `${address.bldG\_NAME\_E1} ${address.district\_e} ${address.osadR\_ST\_E1} ${address.osadR\_LOT\_E1} ${address.osadR\_LOT\_E2} ${address.osadR\_LOT\_E3} ${address.osadR\_LOT\_E4}`  
 const tc = `${address.district\_c}${address.bldG\_NAME\_C1} ${address.osadR\_ST\_C}${address.osadR\_LOT\_C1}${address.osadR\_LOT\_C2}${address.osadR\_LOT\_C3}${address.osadR\_LOT\_C4}`  
 setValue('BlockId', address.adR\_BLK\_ID)  
 setValue('AddressOfPremiseEN', en)  
 setValue('AddressOfPremiseCN', tc)  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 if (inputManually) {  
 setValue('AddressOfPremiseEN', '')  
 setValue('AddressOfPremiseCN', '')  
 setValue('isEngAddressReadOnly', false)  
 setValue('isChiAddressReadOnly', false)  
 }  
 }}  
 />  
 </>  
 )  
}  
export default UpdateInfoForm

## File: bd-scs-backend-web/src/components/Form/Type/withdrawApp.jsx

import React from 'react'  
import { Controller, useFieldArray, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Box from '@mui/material/Box'  
import FormHelperText from '@mui/material/FormHelperText'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import DatePicker from 'src/components/basic/DatePicker'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Typography from 'src/components/basic/Typography'  
import AddressDialog from 'src/components/Dialog/AddressDialog'  
import Autocomplete from 'components/basic/Autocomplete'  
import AddressForm from 'components/Form/addressForm'  
import BasicApplicantForm from 'components/Form/basicApplicantForm'  
import ContactForm2 from 'components/Form/contactForm2'  
import DelegateForm from 'components/Form/delegateForm'  
import SquareButton from 'components/basic/Button/SquareIconButton'  
import PlusIcon from 'assets/icon-plus.png'  
import FolderIcon from 'assets/icon-folder-open.png'  
import FileIcon from 'assets/icon-file.png'  
import DocUploader from 'components/basic/DocUploader'  
import Checkbox from 'components/basic/Checkbox'  
import FormControlLabel from '@mui/material/FormControlLabel'  
function WithdrawAppForm({  
 control,  
 setValue = { setValue },  
 readOnly = false,  
}) {  
 return (  
 <>  
 <Controller  
 control={control}  
 name={`confirmWithdrawApp`}  
 defaultValue={false}  
 render={({  
 field: { onChange, value },  
 fieldState: { invalid, error },  
 }) => (  
 <Stack>  
 <FormControlLabel  
 label="I confirm that I want to withdraw this application."  
 control={  
 <Checkbox checked={value} onChange={onChange} />  
 }  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error.message}  
 </FormHelperText>  
 )}  
 </Stack>  
 )}  
 />  
 </>  
 )  
}  
export default WithdrawAppForm

## File: bd-scs-backend-web/src/components/Form/addressForm.jsx

import { Controller, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Typography from 'components/basic/Typography'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Autocomplete from 'components/basic/Autocomplete'  
import Grid from '@mui/material/Grid'  
import FormHelperText from '@mui/material/FormHelperText'  
import Divider from '@mui/material/Divider'  
import DatePicker from 'src/components/basic/DatePicker'  
import OpenAddressDialogButton from 'components/OpenAddressDialogButton'  
function AddressForm({ control, isDesktop, openDialog, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 const watchIsEngAddressReadOnly = useWatch({  
 control,  
 name: 'isEngAddressReadOnly',  
 })  
 const watchIsChiAddressReadOnly = useWatch({  
 control,  
 name: 'isChiAddressReadOnly',  
 })  
 return (  
 <>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="AddressOfPremiseEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('addressEng')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <OpenAddressDialogButton  
 placeholder={t('placeholder.eng')}  
 onClick={openDialog}  
 readOnly={  
 watchIsEngAddressReadOnly || readOnly  
 }  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={1.5}>  
 <Controller  
 name="AddressOfPremiseENFloor"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('floor')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.floor')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={1.5}>  
 <Controller  
 name="AddressOfPremiseENUnit"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('unit')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.unit')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="AddressOfPremiseCN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('addressChi')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <OpenAddressDialogButton  
 placeholder={t('placeholder.chinese')}  
 onClick={openDialog}  
 readOnly={  
 watchIsChiAddressReadOnly || readOnly  
 }  
 {...field}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={1.5}>  
 <Controller  
 name="AddressOfPremiseCNFloor"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('floor')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.floor')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={1.5}>  
 <Controller  
 name="AddressOfPremiseCNUnit"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('unit')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.unit')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </>  
 )  
}  
export default AddressForm

## File: bd-scs-backend-web/src/components/Form/agencyForm.jsx

import { Controller, useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Typography from 'components/basic/Typography'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Autocomplete from 'components/basic/Autocomplete'  
import Grid from '@mui/material/Grid'  
import FormHelperText from '@mui/material/FormHelperText'  
import Divider from '@mui/material/Divider'  
import DatePicker from 'src/components/basic/DatePicker'  
function AgencyForm({ control, isDesktop, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 return (  
 <>  
 <Grid item xs mb={4}>  
 <Controller  
 name="ApplicantRepOrgEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('agencyNameEng')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs mb={4}>  
 <Controller  
 name="ApplicantRepOrg"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('agencyNameChi')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs mb={4}>  
 <Controller  
 name="ApplicantAddressEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('agencyAddressEng')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs mb={4}>  
 <Controller  
 name="ApplicantAddress"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('agencyAddressChi')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={6} mb={4}>  
 <Controller  
 name="ApplicantTel"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('agencyPhone')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 type="tel"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </>  
 )  
}  
export default AgencyForm

## File: bd-scs-backend-web/src/components/Form/applicantForm.jsx

import { Controller, useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Typography from 'components/basic/Typography'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Autocomplete from 'components/basic/Autocomplete'  
import Grid from '@mui/material/Grid'  
import FormHelperText from '@mui/material/FormHelperText'  
import Divider from '@mui/material/Divider'  
import DatePicker from 'src/components/basic/DatePicker'  
function ApplicantForm({ control, isDesktop, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 return (  
 <>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ApplicantTitle"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('title')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.id?.message}  
 fullWidth  
 required  
 >  
 <Autocomplete  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect'  
 ),  
 id: '',  
 },  
 {  
 label: t('titleOptions.mr'),  
 id: 'mr',  
 },  
 {  
 label: t('titleOptions.mrs'),  
 id: 'mrs',  
 },  
 {  
 label: t('titleOptions.miss'),  
 id: 'miss',  
 },  
 {  
 label: t('titleOptions.ms'),  
 id: 'ms',  
 },  
 ]}  
 getOptionLabel={(option) => option.label}  
 {...field}  
 value={  
 !field?.value?.id ? null : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disableClearable  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantNameEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicantFullNameEng')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantName"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicantFullNameChi')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ApplicantEmail"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicantAgencyEmail')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantMobile"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicantAgencyPhone')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.receiveSCS')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantHKIC"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicantIdCardNo')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.receiveSCS')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <Grid item xs={6} mb={4}>  
 <Controller  
 name="ApplicationDate"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('date')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <DatePicker {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </>  
 )  
}  
export default ApplicantForm

## File: bd-scs-backend-web/src/components/Form/basicApplicantForm.jsx

import { Controller, useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Typography from 'components/basic/Typography'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Autocomplete from 'components/basic/Autocomplete'  
import Grid from '@mui/material/Grid'  
import FormHelperText from '@mui/material/FormHelperText'  
import Divider from '@mui/material/Divider'  
import DatePicker from 'src/components/basic/DatePicker'  
import OpenAddressDialogButton from 'components/OpenAddressDialogButton'  
function BasicApplicantForm({ control, isDesktop, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 return (  
 <>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ApplicantName"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicantFullName')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField  
 {...field}  
 type="text"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantEmail"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicantEmail')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField  
 {...field}  
 type="email"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicantMobile"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicantPhone')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField  
 {...field}  
 type="tel"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </>  
 )  
}  
export default BasicApplicantForm

## File: bd-scs-backend-web/src/components/Form/contactForm.jsx

import { Controller, useForm, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Typography from 'components/basic/Typography'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Autocomplete from 'components/basic/Autocomplete'  
import Grid from '@mui/material/Grid'  
import FormHelperText from '@mui/material/FormHelperText'  
import Divider from '@mui/material/Divider'  
import DatePicker from 'src/components/basic/DatePicker'  
import OpenAddressDialogButton from 'components/OpenAddressDialogButton'  
function ContactForm({ control, isDesktop, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 return (  
 <>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ContactPTitle"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('title')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.id?.message}  
 fullWidth={true}  
 required  
 >  
 <Autocomplete  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect'  
 ),  
 id: '',  
 },  
 {  
 label: t('titleOptions.mr'),  
 id: 'mr',  
 },  
 {  
 label: t('titleOptions.mrs'),  
 id: 'mrs',  
 },  
 {  
 label: t('titleOptions.miss'),  
 id: 'miss',  
 },  
 {  
 label: t('titleOptions.ms'),  
 id: 'ms',  
 },  
 ]}  
 getOptionLabel={(option) => option.label}  
 {...field}  
 value={  
 !field?.value?.id ? null : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disableClearable  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ContactPName"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('contactName')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ContactPFax"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('contactFax')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField  
 {...field}  
 type="tel"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ContactPEmail"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('contactEmail')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 type="email"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ContactPMobile"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('contactPhone')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 type="tel"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {isDesktop && <Grid item xs />}  
 </ResponsiveGrid>  
 <Grid item xs mb={4}>  
 <Controller  
 name="ContactPAddress"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('contactAddressEng')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </>  
 )  
}  
export default ContactForm

## File: bd-scs-backend-web/src/components/Form/contactForm2.jsx

import { Controller, useForm, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Typography from 'components/basic/Typography'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Autocomplete from 'components/basic/Autocomplete'  
import Grid from '@mui/material/Grid'  
import FormHelperText from '@mui/material/FormHelperText'  
import Divider from '@mui/material/Divider'  
import DatePicker from 'src/components/basic/DatePicker'  
import OpenAddressDialogButton from 'components/OpenAddressDialogButton'  
function ContactForm({ control, isDesktop, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 return (  
 <>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ContactPName"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('contactName2')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ContactPEmail"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('contactEmail')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.receiveSCS')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ContactPMobile"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('contactMobilePhone')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.receiveSCS')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ContactPTel"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('contactPhone2')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ContactPFax"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('contactFax')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicationDate"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('date')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <DatePicker {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </>  
 )  
}  
export default ContactForm

## File: bd-scs-backend-web/src/components/Form/delegateForm.jsx

import { Controller, useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Typography from 'components/basic/Typography'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Autocomplete from 'components/basic/Autocomplete'  
import Grid from '@mui/material/Grid'  
import FormHelperText from '@mui/material/FormHelperText'  
import Divider from '@mui/material/Divider'  
import { RegTypes1, RegTypes2, RegTypes3 } from 'src/constants/options'  
function DelegateForm({ control, isDesktop, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 return (  
 <>  
 <Typography  
 variant="title3"  
 color="text.secondary"  
 sx={{ mb: 2, mt: 3 }}  
 >  
 {t('uploadPersonInfo', { ns: 'form' })}  
 </Typography>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="EnglishNameOfAp1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('fullNameEng', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ChineseNameOfAp1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('fullNameChi', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <FormControl  
 title={t('registerNo', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <Grid container direction="row" gap={2}>  
 <Controller  
 name="RegTypeOfAp1"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <Grid item>  
 <Autocomplete  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect',  
 { ns: 'form' }  
 ),  
 id: '',  
 },  
 ...RegTypes1,  
 ]}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 sx={{ minWidth: 120 }}  
 placeholder={t(  
 'placeholder.pleaseSelect',  
 { ns: 'form' }  
 )}  
 disableClearable  
 disabled={readOnly}  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error?.message}  
 </FormHelperText>  
 )}  
 </Grid>  
 )}  
 />  
 <Controller  
 name="CertRegNoOfAp1"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <Grid item xs>  
 <TextField  
 fullWidth  
 {...field}  
 placeholder={t(  
 'placeholder.registerNo',  
 { ns: 'form' }  
 )}  
 disabled={readOnly}  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error?.message}  
 </FormHelperText>  
 )}  
 </Grid>  
 )}  
 />  
 </Grid>  
 </FormControl>  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="EmailOfAp1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('email', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.receiveSCS', {  
 ns: 'form',  
 })}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="MobileOfAp1"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('phone', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.receiveSCS', {  
 ns: 'form',  
 })}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs />  
 </ResponsiveGrid>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Typography  
 variant="title3"  
 color="text.secondary"  
 sx={{ mb: 2, mt: 3 }}  
 >  
 {t('structuralEngineerInformation', { ns: 'form' })}  
 </Typography>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="EnglishNameOfAp2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('fullNameEng', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ChineseNameOfAp2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('fullNameChi', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <FormControl  
 title={t('registerNo', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <Grid container direction="row" gap={2}>  
 <Controller  
 name="RegTypeOfAp2"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <Grid item>  
 <Autocomplete  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect',  
 { ns: 'form' }  
 ),  
 id: '',  
 },  
 ...RegTypes2,  
 ]}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 sx={{ minWidth: 120 }}  
 placeholder={t(  
 'placeholder.pleaseSelect',  
 { ns: 'form' }  
 )}  
 disableClearable  
 disabled={readOnly}  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error?.message}  
 </FormHelperText>  
 )}  
 </Grid>  
 )}  
 />  
 <Controller  
 name="CertRegNoOfAp2"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <Grid item xs>  
 <TextField  
 fullWidth  
 {...field}  
 placeholder={t(  
 'placeholder.registerNo',  
 { ns: 'form' }  
 )}  
 disabled={readOnly}  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error?.message}  
 </FormHelperText>  
 )}  
 </Grid>  
 )}  
 />  
 </Grid>  
 </FormControl>  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="EmailOfAp2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('email', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.receiveSCS', {  
 ns: 'form',  
 })}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="MobileOfAp2"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('phone', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.receiveSCS', {  
 ns: 'form',  
 })}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs />  
 </ResponsiveGrid>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Typography  
 variant="title3"  
 color="text.secondary"  
 sx={{ mb: 2, mt: 3 }}  
 >  
 {t('engineerInformation', { ns: 'form' })}  
 </Typography>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="EnglishNameOfRse"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('fullNameEng', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ChineseNameOfRse"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('fullNameChi', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <FormControl  
 title={t('registerNo', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <Grid container direction="row" gap={2}>  
 <Controller  
 name="RegTypeOfRse"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <Grid item>  
 <Autocomplete  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect',  
 { ns: 'form' }  
 ),  
 id: '',  
 },  
 ...RegTypes3,  
 ]}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 sx={{ minWidth: 120 }}  
 placeholder={t(  
 'placeholder.pleaseSelect',  
 { ns: 'form' }  
 )}  
 disableClearable  
 disabled={readOnly}  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error?.message}  
 </FormHelperText>  
 )}  
 </Grid>  
 )}  
 />  
 <Controller  
 name="CertRegNoOfRse"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <Grid item xs>  
 <TextField  
 fullWidth  
 {...field}  
 placeholder={t(  
 'placeholder.registerNo',  
 { ns: 'form' }  
 )}  
 disabled={readOnly}  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error?.message}  
 </FormHelperText>  
 )}  
 </Grid>  
 )}  
 />  
 </Grid>  
 </FormControl>  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="EmailOfRse"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('email', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.receiveSCS', {  
 ns: 'form',  
 })}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="MobileOfRse"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('phone', { ns: 'form' })}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.receiveSCS', {  
 ns: 'form',  
 })}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs />  
 </ResponsiveGrid>  
 </>  
 )  
}  
export default DelegateForm

## File: bd-scs-backend-web/src/components/Form/proposedCenterInfoForm.jsx

import { Controller, useForm, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Typography from 'components/basic/Typography'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Autocomplete from 'components/basic/Autocomplete'  
import Grid from '@mui/material/Grid'  
import FormHelperText from '@mui/material/FormHelperText'  
import Divider from '@mui/material/Divider'  
import DatePicker from 'src/components/basic/DatePicker'  
import OpenAddressDialogButton from 'components/OpenAddressDialogButton'  
function ProposedCenterInfoForm({  
 control,  
 isDesktop,  
 openDialog,  
 readOnly = false,  
}) {  
 const { t, i18n } = useTranslation(['form'])  
 const watchIsEngAddressReadOnly = useWatch({  
 control,  
 name: 'isEngAddressReadOnly',  
 })  
 const watchIsChiAddressReadOnly = useWatch({  
 control,  
 name: 'isChiAddressReadOnly',  
 })  
 return (  
 <>  
 <Typography  
 variant="title3"  
 color="text.secondary"  
 sx={{ mb: 2, mt: 3 }}  
 >  
 {t('proposedCenterInformation')}  
 </Typography>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="AddressOfPremiseEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('addressEng')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <OpenAddressDialogButton  
 onClick={openDialog}  
 readOnly={  
 watchIsEngAddressReadOnly || readOnly  
 }  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={1.5}>  
 <Controller  
 name="AddressOfPremiseENFloor"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('floor')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.floor')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={1.5}>  
 <Controller  
 name="AddressOfPremiseENUnit"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('unit')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.unit')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="AddressOfPremiseCN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('addressChi')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <OpenAddressDialogButton  
 onClick={openDialog}  
 readOnly={  
 watchIsChiAddressReadOnly || readOnly  
 }  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={1.5}>  
 <Controller  
 name="AddressOfPremiseCNFloor"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('floor')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.floor')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={1.5}>  
 <Controller  
 name="AddressOfPremiseCNUnit"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('unit')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 placeholder={t('placeholder.unit')}  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <Grid item xs mb={4}>  
 <FormControl  
 title={t('proposedCenterName')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 required  
 >  
 <Controller  
 name="NameOfSchoolEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <>  
 <TextField  
 {...field}  
 sx={{ mb: invalid ? 0 : 1 }}  
 placeholder={t('placeholder.eng')}  
 disabled={readOnly}  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error?.message}  
 </FormHelperText>  
 )}  
 </>  
 )}  
 />  
 <Controller  
 name="NameOfSchoolCN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <>  
 <TextField  
 {...field}  
 placeholder={t('placeholder.chinese')}  
 disabled={readOnly}  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error?.message}  
 </FormHelperText>  
 )}  
 </>  
 )}  
 />  
 </FormControl>  
 </Grid>  
 <Grid item xs mb={4}>  
 <Controller  
 name="PhoneOfSchool"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('proposedCenterPhone')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <TextField  
 {...field}  
 type="tel"  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </>  
 )  
}  
export default ProposedCenterInfoForm

## File: bd-scs-backend-web/src/components/Form/schoolForm.jsx

import { Controller, useForm, useWatch } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import Typography from 'components/basic/Typography'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import Autocomplete from 'components/basic/Autocomplete'  
import Grid from '@mui/material/Grid'  
import FormHelperText from '@mui/material/FormHelperText'  
import Divider from '@mui/material/Divider'  
import DatePicker from 'src/components/basic/DatePicker'  
import OpenAddressDialogButton from 'components/OpenAddressDialogButton'  
import { SchoolCategory, StudentAge } from 'src/constants/options'  
function SchoolForm({ control, isDesktop, readOnly = false }) {  
 const { t, i18n } = useTranslation(['form'])  
 const watchSchoolCategory = useWatch({  
 control,  
 name: 'DescriptionOfSchool',  
 })  
 return (  
 <>  
 <Grid container direction="row" gap={3} mb={4}>  
 <FormControl  
 title={t('schoolName')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 required  
 >  
 <Controller  
 name="NameOfSchoolEN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <>  
 <TextField  
 {...field}  
 sx={{ mb: invalid ? 0 : 1 }}  
 placeholder={t('placeholder.eng')}  
 disabled={readOnly}  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error?.message}  
 </FormHelperText>  
 )}  
 </>  
 )}  
 />  
 <Controller  
 name="NameOfSchoolCN"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <>  
 <TextField  
 {...field}  
 placeholder={t('placeholder.chinese')}  
 disabled={readOnly}  
 />  
 {invalid && (  
 <FormHelperText error>  
 {error?.message}  
 </FormHelperText>  
 )}  
 </>  
 )}  
 />  
 </FormControl>  
 </Grid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="DescriptionOfSchool"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('schoolCategory')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 error={invalid}  
 helperText={error?.message}  
 required  
 >  
 <Autocomplete  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect'  
 ),  
 id: '',  
 },  
 ...SchoolCategory,  
 ]}  
 getOptionLabel={(option) => option.label}  
 {...field}  
 value={  
 !field?.value?.id ? null : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disableClearable  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {watchSchoolCategory?.id === 'other\_sch' ? (  
 <Grid item xs>  
 <Controller  
 name="OtherSchoolDesc"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('pleaseRemarks')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField {...field} disabled={readOnly} />  
 <FormHelperText error>  
 {t('warning.required')}  
 </FormHelperText>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 ) : isDesktop ? (  
 <Grid item xs />  
 ) : null}  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="AgeOfStudent"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('studentAge')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 error={invalid}  
 helperText={error?.message}  
 required  
 >  
 <Autocomplete  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect'  
 ),  
 id: '',  
 },  
 ...StudentAge,  
 ]}  
 getOptionLabel={(option) => option.label}  
 {...field}  
 value={  
 !field?.value?.id ? null : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disableClearable  
 disabled={readOnly}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="EstimatedNoOfStudent"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('estimateTheNumberOfStudents')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 required  
 >  
 <TextField {...field} disabled={readOnly} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </>  
 )  
}  
export default SchoolForm

## File: bd-scs-backend-web/src/config/cat.js

const CAT\_DESCRIPTIONS = {  
 ubw\_q1\_options: {  
 a: "Unauthorised rooftop/flat roof/lane/yard/open area structures forming part of the licensed premises.",  
 b: "Unauthorised structures on/suspended from balcony/verandah/canopy.",  
 c: "Unauthorised canopies/retractable awnings/projections projecting from the external wall of building erected after the implementation of such MW items under the Minor Works Control System (MWCS).",  
 d: "Unauthorised signboards (including shopfront signboards) not eligible for joining the Signboard Validation Scheme.",  
 e: "Unauthorised supporting structures/frames for air-conditioning units/light fittings/mechanical plants (e.g. water cooling towers and associated supporting structures) projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof erected after the implementation of such MW items under the MWCS.",  
 f: "Unauthorised external metal ventilation ducts and associated supporting frames projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof erected after the implementation of such MW items under the MWCS.",  
 g: "Unauthorised obstructions to smoke vents.",  
 h: "Unauthorised alteration/removal of compartment walls/fire resisting walls and doors.",  
 i: "Unauthorised openings/slab over existing floors for pipe ducts.",  
 j: "Unauthorised installation of glass panels on external walls on upper floors.",  
 k: "Unauthorised removal of protective barrier/external wall without providing a permanent protective barrier at a height of 1.1 m minimum resulting in a danger of falling persons or objects.",  
 l: "Unauthorised alteration/removal of approved barrier free facilities for persons with a disability.",  
 m: "Unauthorised hollow raised platforms within premises hindering the access for persons with a disability.",  
 n: "Unauthorised removal of internal staircases not having been certified by AP/RSE as being structurally safe and accompanied with supporting calculations.",  
 o: "Unauthorised supporting frames for suspending an air-conditioning/mechanical ventilation plants of weight more than 150 kg inside the licensed premises not having been certified by AP/RSE as being structurally safe and accompanied with supporting calculations.",  
 p: "Unauthorised external solid fence walls/mesh fences/railings/poles on roof erected after the implementation of such MW items under the MWCS.",  
 q: "Unauthorised slabs filling up approved cockloft/staircase voids.",  
 r: "Unauthorised cocklofts, intermediate floors and floor extensions.",  
 s: "Unauthorised staircases.",  
 t: "Unauthorised openings through structural slabs and walls.",  
 u: "Unauthorised removal, partial removal/major alteration of structural members.",  
 v: "UBWs in common areas resulting in obstruction of means of escape from the licensed premises/the building.",  
 },  
 ubw\_q2\_q1\_option: {  
 a: "Supporting structures for building services installation (BSI) located on-grade/on roof (other than cantilevered slab/inaccessible roof) with the height of the supporting structures not more than 1.5 m.",  
 b: "Shopfront side structures (SSS), projecting not more than 150 mm from the original building line. The wall area covered by any one side of SSS is not more than 5 m2, the distance between any part of the SSS and the ground is not more than 6 m and the SSS do not obstruct access to any public utilities.",  
 c: "Shopfront overhead projections, which do not consist of wooden structural elements, projecting not more than 600 mm from the external wall of building having minimum vertical clearance of 2.5 m and not supporting other structures or equipment such as air-conditioning units. If the overhead projections consist of stone, the difference between any part of the projection and the ground should not be more than 6 m.",  
 d: "Canopies, which do not consist of stone, tile, glass/cement mortar and are not constructed of concrete, projecting not more than 500 mm from the external wall of building. The distance between the highest point of the canopy and the roof/ground is more than 3 m.",  
 e: "Retractable awnings at a door opening not serving as an exit for an escape staircase/leading to a balcony/verandah or a window opening for uses other than plant room/lavatory/bathroom/kitchen, projecting not more than 2 m (over the roof)/2.5 m (any other cases) from the external wall of building when fully extended and not more than 500 mm when retracted. The distance between the highest point of the awning and the roof/ground is not more than 5.5 m. The width of awning is at most 500 mm wider than both sides of such opening.",  
 f: "Existing signboards validated through the Signboard Validation Scheme (SVS).",  
 g: "Supporting frames for air-conditioning units/lighting fittings projecting not more than 600 mm from the external wall of building. The distance between the highest point of the frame and the roof/ground is more than 3 m.",  
 h: "External metal ventilation ducts and the associated supporting frames located on-grade/on roof. The distance between the highest point of the duct/frame and the roof/ground is not more than 1.5 m.",  
 i: "External metal ventilation ducts and associated supporting frames projecting not more than 600 mm from the external wall of building. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m.",  
 j: "External metal ventilation ducts and the associated supporting frames hung underneath the soffit of a balcony/verandah/canopy (other than cantilevered slab) with largest cross-sectional dimension of the duct not more than 600 mm. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m.",  
 k: "External metal ventilation ducts and the associated supporting frames located on a balcony/verandah/canopy (other than cantilevered slab) with largest cross-sectional dimension of the duct not more than 600 mm. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m. The distance between the highest point of the duct/frame and the balcony/verandah/canopy is not more than 1.5 m.",  
 l: "External poles located on roof including any feature at its top with the height not more than 1.5 m.",  
 m: "External solid fence walls located on roof with the height not more than 1.1 m and thickness not more than 100 mm.",  
 n: "External mesh fences/railings located on roof including any feature at its top with the height not more than 1.5 m and is not used as a protective barrier. If lower part is a solid fence wall, the height and thickness are not more than 300 mm and 100 mm respectively.",  
 o: "Unauthorised installation of glass panels on external walls on upper floors having area of each glass panel not exceeding 6 m and the least dimension of such glass panel not exceeding 1.8 m and not overloading to cantilevered slab type balconies. The remaining external wall can provide adequate protection for openings to prevent the spread of fire between floors concerned.",  
 p: "Supporting frames for suspending an air-conditioning/mechanical ventilation plant of more than 150 kg inside a building certified by AP/RSE as being structurally safe with supporting calculations.",  
 q: "Unauthorised openings/slabs over existing floors for food hoists and pipe ducts having been certified by AP/RSE as being structurally safe with supporting calculations and should not result in additional gross floor area under the Building (Planning) Regulations.",  
 r: "Unauthorised alteration/removal of approved access ramp for persons with a disability / accessible toilet for persons with a disability / accessible urinal subject to re-provision of access ramp for persons with a disability/ accessible toilet for persons with a disability / accessible urinal in compliance with Design Manual: Barrier Free Access.",  
 s: "Hollow raised platforms with a height not greater than 600 mm within premises and not hindering the access for persons with a disability.",  
 t: "Hollow raised platforms with a height between 600 mm to 2000 mm within premises if such platforms having been certified by AP/RSE as being structurally safe with supporting calculations and not hindering the access for persons with a disability.",  
 u: "Removal of internal staircase having been certified by AP/RSE as being structurally safe with supporting calculations and the staircase void should be provided by protective barriers in accordance with Building (Construction) Regulation.",  
 v: "Kitchens and toilets within approved premises and drainage works certified as being in proper function and properly connected.",  
 w: "Small storage chamber for LPG cylinders with an aggregate capacity not exceeding 130 L and complying with 'LPG Installation for Catering Purposes in Commercial Premises' issued by the Electrical and Mechanical Services Department.",  
 },  
 desk\_study\_q5\_cat1:  
 "A revised application form with correct address should be submitted.",  
 structural\_schnlhkinds\_q4\_options\_cat1\_1: {  
 a: "Construction/Alteration/Repair/Replacement/Removal of window or window wall. (建造/改動/修葺/更換/拆除窗或玻璃外牆)",  
 b: "Erection/Alteration/Removal of supporting frames for air-conditioning unit projecting from an external wall of a building. (豎設/改動/拆除自建築物外牆伸出的，用於支承空調機的支架)",  
 c: `Erection/Alteration/Removal of external Ventilation duct works and associated supporting frames projecting from an external wall of a building. (豎設/改動/拆除自建築物外牆伸出的室外金屬通風管道或相關的承托支架)`,  
 d: "Repair of structural elements. (修葺結構構件)",  
 e: "Erection/Alteration/Removal of wall signboard. (豎設/改動/拆除靠牆招牌)",  
 f: "Erection, repair, alteration or removal of aboveground / underground drains (豎設/修葺/改動/拆除地底以上 / 地下的排水渠)",  
 },  
 structural\_schnlhkinds\_q4\_options\_cat1\_2: {  
 a: "Removal of unauthorised structures projecting from the external wall of a building. (拆除自建築物外牆伸出的違例構築物)",  
 b: "Removal of unauthorised floor slab. (拆除違例樓板)",  
 c: "Erection/Alteration/Removal of protective barrier. (豎設/改動/拆除防護欄障)",  
 },  
 structural\_schnlhkinds\_q4\_options\_cat1\_3: {  
 a: "Erection/Alteration/Repair/Removal of retractable awnings. (豎設/改動/修葺/拆除可收合遮篷)",  
 b: "Erection/Alteration/Removal of external non-load bearing reinforced concrete wall of a building. (豎設/改動/拆除用鋼筋混凝土建造的非承重外牆)",  
 c: "Erection/Alteration/Removal of external non-load bearing block wall of a building. (豎設/改動/拆除用磚建造的非承重外牆)",  
 d: "Erection or alteration of any solid fence wall on-grade (豎設/改動於地面上的實心圍牆)",  
 e: "Erection or alteration of any external pole on-grade (豎設/改動於地面上的室外支柱)",  
 f: `Erection/Alteration/Repair/ Removal of mesh fences or metal railings on a roof (豎設／改動／ 修葺／拆除建築物屋頂上的網欄或金屬欄杆)`,  
 },  
 structural\_schnlhkinds\_q4\_options\_cat1\_4: {  
 a: "Alteration of any opening to the enclosure of the staircase that is used as a means of access for firefighting and rescue or its protected lobby. (改動用作消防和救援進出途徑的樓梯或其防護門廊的圍封部分上的洞口)",  
 b: "Erection of non-load bearing block walls in a flat. (於樓宇單位內豎設用磚建造的非承重牆)",  
 c: "Thickening of floor slab in a flat by laying solid screeding. (舖設實心地台，以加厚樓宇單位內的樓板)",  
 d: "Erection/Alteration of supporting frames for suspending an air-conditioning plant or a mechanical ventilation plant inside a building. (豎設/改動於建築物內用作懸掛空調裝置或機械通風裝置的承托支)",  
 e: "Erection/Alteration of ventilation duct works or associated supporting frames inside a building. (豎設/改動 於建築物內豎設用作懸掛空調裝置或機械通風裝置的承托支架)",  
 f: "Erection/Alteration of fire damper in a ventilation system. (豎設/改動於通風系統中的防火閘)"  
 },  
 structural\_schnlhkinds\_q5\_options\_cat1: {  
 a: "Any false ceiling of the proposed licensed premises should be constructed of lightweight material and the void above false ceiling should be kept vacant and not to be used for any other purpose.",  
 b: "Only lightweight partitions of gypsum board or similar construction should be erected at the licensed premises.",  
 c: "No plant and equipment or the like i.e. Building Services Installation suspended from the structural ceiling with the supporting frames having a vertical clearance of less than 2000mm within the licensed premises weights more than 150kg.",  
 },  
 structural\_schnlhkinds\_q6\_options\_cat1: {  
 a: `There is no information as to the design and construction of the subject premises, and I am unable to certify that the premises are structural suitable for use as school. You should therefore required to appoint an Authorized Person/ Registered Structural Engineer (""AP/RSE"") in respect of the followings :-  
(a) to check that the premises are in sound structural condition;  
(b) to submit justification that the premises are suitable for the intended use as school, having regard to the loading for which the premises was designed and constructed; and  
(c) to check that the resistance of fire of the structural elements of the premises are satisfactory.  
The above structural justification should be submitted for BD's acceptance. Upon the receipt of the revised plan, additional building safety requirement may be imposed.`,  
 },  
 frc\_q1\_q2b\_cat1: {  
 a: `An undertaking letter from the owner/management company of the building confirming the following aspects should be submitted:  
(a) The management company is to ensure that with the endorsement of an AP/RSE, the common internal corridors are constructed with materials having an FRR of not less than \_\_\_\_ minutes and the doors thereat are having an FRR of not less than \_\_\_\_ minutes and be self-closing. An AP/RSE is required to check and certify the fire resistance rating (FRR) and submit duly completed Appendix A to Practice Note for Authorized Persons and Registered Structural Engineers APP-13 by an AP/RSE with supporting test/assessment reports. (Such reports are not required for submission if Practice Note for Authorized Persons and Registered Structural Engineers ADM-20 is applicable.)`,  
 },  
 frc\_q2\_q1\_cat1: {  
 a: `An undertaking letter from the applicant confirming the following aspects should be submitted:  
The adjoining non-licence area is under same management and control by the applicant and the operation hours of the licensed premises should be the same or longer than the non-licence area. Revised plan with an annotation "Licensed premsies and adjoining non-licence area are under the same control and management of the applicant" should be submitted.`,  
 },  
 ubw\_q1\_q3\_cat1\_1: {  
 a: "Construction/Alteration/Repair/Replacement/Removal of window or window wall. (建造/改動/修葺/更換/拆除窗或玻璃外牆)",  
 b: "Erection/Alteration/Removal of supporting frames for air-conditioning unit projecting from an external wall of a building. (豎設/改動/拆除自建築物外牆伸出的，用於支承空調機的支架)",  
 c: "Erection/Alteration/Removal of external Ventilation duct works and associated supporting frames projecting from an external wall of a building. (豎設/改動/拆除自建築物外牆伸出的室外金屬通風管道或相關的承托支架)",  
 d: "Erection/Alteration/Removal of wall signboard. (豎設/改動/拆除 靠牆招牌)",  
 e: "Removal of unauthorised structures projecting from the external wall of a building. (拆除自建築物外牆伸出的違例構築物)",  
 f: "Removal of unauthorised suspended slab/unauthorised structure hung underneath the soffit of a balcony or canopy (拆除懸掛式違例樓板／懸掛或固定於露台或簷篷部分的違例構築物)",  
 g: "Removal of any unauthorized structure located on-grade or on a slab (other than a cantilevered slab) ((拆除位於地面或平板 ( 不包括懸臂式平板 ) 上的違例構築物)",  
 h: "Erection/Alteration/Removal of protective barrier. (豎設/改動/拆除防護欄障)",  
 i: "Erection/Alteration/Repair/Removal of retractable awnings. (豎設/改動/修葺/拆除可收合遮篷)",  
 j: "Erection/Alteration/Removal of external non-load bearing reinforced concrete wall of a building. (豎設/改動/拆除用鋼筋混凝土建造的非承重外牆)",  
 k: "Erection/Alteration/Removal of external non-load bearing block wall of a building. (豎設/改動/拆除用磚建造的非承重外牆)",  
 l: "Erection/Alteration/Repair/ Removal of mesh fences or metal railings on a roof (豎設／改動／ 修葺／拆除建築物屋頂上的網欄或金屬欄杆)",  
 },  
 ubw\_q1\_q3\_cat1: {  
 a: "Photo records showing the completion of removal/ rectification/ reinstatement works of \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ should be submitted for BD’s acceptance.",  
 },  
 ubw\_q4\_cat1: {  
 a: "Revised plan showing the removal/ deleteion of \_\_\_\_\_\_\_ should be submitted for BD's acceptance. Upon the receipt of the revised plan, additional building safety requirement may be imposed.",  
 },  
 misc\_q1\_cat1: {  
 a: "The application date shown on the plan should be DATE’ in accordance with the submitted Form A2.",  
 },  
 desk\_study\_q4\_cat2\_3a: {  
 a: `Alteration and Addition Works under BD ref.: bd\_ref`,  
 },  
 ubw\_q2\_q2\_cat2\_1a: {  
 a: "Supporting frames for suspending an air-conditioning/mechanical ventilation plant of more than 150 kg inside a building",  
 b: "Unauthorised openings/slabs over existing floors for pipe ducts",  
 c: "Hollow raised platforms with a height between 600 mm to 2 000 mm within premises",  
 d: "Removal of internal staircase",  
 },  
 ubw\_q2\_q1\_cat2\_1b: {  
 a: "Supporting structures for building services installation (BSI) located on-grade/on roof (other than cantilevered slab/inaccessible roof) with the height of the supporting structures not more than 1.5 m.",  
 b: "Shopfront side structures (SSS), projecting not more than 150 mm from the original building line. The wall area covered by any one side of SSS is not more than 5 m2, the distance between any part of the SSS and the ground is not more than 6 m and the SSS do not obstruct access to any public utilities.",  
 c: "Shopfront overhead projections, which do not consist of wooden structural elements, projecting not more than 600 mm from the external wall of building having minimum vertical clearance of 2.5 m and not supporting other structures or equipment such as air-conditioning units. If the overhead projections consist of stone, the difference between any part of the projection and the ground should not be more than 6 m.",  
 d: "Canopies, which do not consist of stone, tile, glass/cement mortar and are not constructed of concrete, projecting not more than 500 mm from the external wall of building. The distance between the highest point of the canopy and the roof/ground is more than 3 m.",  
 e: "Retractable awnings at a door opening not serving as an exit for an escape staircase/ leading to a balcony/verandah or a window opening for uses other than plant room/ lavatory/bathroom/kitchen, projecting not more than 2 m (over the roof)/2.5 m (any other cases) from the external wall of building when fully extended and not more than 500 mm when retracted. The distance between the highest point of the awning and the roof/ground is not more than 5.5 m. The width of awning is at most 500 mm wider than both sides of such opening.",  
 f: "Existing signboards validated through the Signboard Validation Scheme (SVS).",  
 g: "Supporting frames for air-conditioning units/lighting fittings projecting not more than 600 mm from the external wall of building. The distance between the highest point of the frame and the roof/ground is more than 3 m.",  
 h: "External metal ventilation ducts and the associated supporting frames located on-grade/on roof. The distance between the highest point of the duct/frame and the roof/ground is not more than 1.5 m.",  
 i: "External metal ventilation ducts and associated supporting frames projecting not more than 600 mm from the external wall of building. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m.",  
 j: "External metal ventilation ducts and the associated supporting frames hung underneath the soffit of a balcony/verandah/canopy (other than cantilevered slab) with largest cross-sectional dimension of the duct not more than 600 mm. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m.",  
 k: "External metal ventilation ducts and the associated supporting frames located on a balcony/verandah/canopy (other than cantilevered slab) with largest cross-sectional dimension of the duct not more than 600 mm. The distance between the highest point of the duct/frame and the ground/roof is more than 3 m. The distance between the highest point of the duct/frame and the balcony/verandah/canopy is not more than 1.5 m.",  
 l: "External poles located on roof including any feature at its top with the height not more than 1.5 m.",  
 m: "External solid fence walls located on roof with the height not more than 1.1 m and thickness not more than 100 mm.",  
 n: "External mesh fences/railings located on roof including any feature at its top with the height not more than 1.5 m and is not used as a protective barrier. If lower part is a solid fence wall, the height and thickness are not more than 300 mm and 100 mm respectively.",  
 o: "Unauthorised installation of glass panels on external walls on upper floors having area of each glass panel not exceeding 6 m and the least dimension of such glass panel not exceeding 1.8 m and not overloading to cantilevered slab type balconies. The remaining external wall can provide adequate protection for openings to prevent the spread of fire between floors concerned.",  
 p: "Supporting frames for suspending an air-conditioning/mechanical ventilation plant of more than 150 kg inside a building certified by AP/RSE as being structurally safe with supporting calculations.",  
 q: "Unauthorised openings/slabs over existing floors for food hoists and pipe ducts having been certified by AP/RSE as being structurally safe with supporting calculations and should not result in additional gross floor area under the Building (Planning) Regulations.",  
 r: "Unauthorised alteration/removal of approved access ramp for persons with a disability/accessible toilet for persons with a disability/accessible urinal subject to re-provision of access ramp for persons with a disability/accessible toilet for persons with a disability/accessible urinal in compliance with Design Manual: Barrier Free Access.",  
 s: "Hollow raised platforms with a height not greater than 600 mm within premises and not hindering the access for persons with a disability.",  
 t: "Hollow raised platforms with a height between 600 mm to 2,000 mm within premises if such platforms having been certified by AP/RSE as being structurally safe with supporting calculations and not hindering the access for persons with a disability.",  
 u: "Removal of internal staircase having been certified by AP/RSE as being structurally safe with supporting calculations and the staircase void should be provided by protective barriers in accordance with Building (Construction) Regulation.",  
 v: "Kitchens and toilets within approved premises and drainage works certified as being in proper function and properly connected.",  
 w: "Small storage chamber for LPG cylinders with an aggregate capacity not exceeding 130 L and complying with 'LPG Installation for Catering Purposes in Commercial Premises' issued by the Electrical and Mechanical Services Department.",  
 },  
 frc\_q2\_q2\_cat2\_3b: {  
 a: "The enclosure walls separating the licensed premises from adjoining units which is non-licence area, as highlighted on the attached plan, should be constructed up to structural ceiling and have an FRR of not less than \_\_\_\_\_ minutes.",  
 },  
 frc\_q3\_cat2\_3b: {  
 a: "The altered walls, as highlighted on the attached plan, should be constructed up to structural ceiling and have an FRR of not less than \_\_\_\_\_ minutes.",  
 b: "The entrance door of the licensed premises should have an FRR of not less than \_\_\_\_\_ minutes. Such door should be self-closing and provided with smoke seals.",  
 c: "Fire stop for sealing up openings for passage of building services through the the enclosure wall of the licensed premises facing the common internal corridor having an FRR of not less than \_\_\_\_\_ minutes.",  
 d: "Fire dampers for protecting openings for passage of ventilation ducts through the the enclosure wall of the licensed premises facing the common internal corridor having an FRR of not less than \_\_\_\_\_ minutes.",  
 },  
 frc\_q4\_cat2\_3b: {  
 a: "The new walls and doors for the fireman's lift lobby/ protected lobby of exit staircase should have an FRR of not less than \_\_\_ minutes. The walls should be constructed up to the structural ceiling and the doors should be self-closing and provided with vision panels and smoke seals.",  
 b: "The new doors of Male/ Female LAV. within the protected lobby of the exit staircase, as highlighted on the attached plan, should have an FRR of not less than \_\_\_ minutes. Such door should be self-closing and provided with smoke seals.",  
 c: "The solid return having a minimum width of 450mm up to structural floor should be provided along the shopfront of the premises adjacent to the exit staircase of the building and should have an FRR of not less than 60 minutes.",  
 },  
 ubw\_q1\_q3\_cat2\_4: {  
 a: "Unauthorised rooftop/flat roof/lane/yard/open area structures should be removed.",  
 b: "Unauthorised structures on/suspended from balcony/verandah/canopy should be removed.",  
 c: "Unauthorised canopies/retractable awnings/projections projecting from the external wall of building should be removed.",  
 d: "Unauthorised signboards should be removed.",  
 e: "Unauthorised supporting structures/frames for air-conditioning units/light fittings/mechanical plants projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof should be removed.",  
 f: "Unauthorised external metal ventilation ducts and associated supporting frames projecting from the external wall of building or on/hung underneath the soffit of balcony/verandah/canopy or located on-grade/on roof should be removed.",  
 g: "Any obstruction to smoke vents is removed and smoke vents reinstated in accordance with the approved plans. The AP/RSE is required to certify the fire resistance rating (FRR) for the material used for reinstatement of the smoke vents and submit duly completed Appendix A to Practice Note for Authorized Persons and Registered Structural Engineers and Registered Geotechnical Engineers (PNAP) APP-13 with supporting test/assessment reports to BD (Such reports are not required for submission if PNAP ADM-20 is applicable).",  
 h: "Unauthorised alteration/removal of compartment walls/fire resisting walls and doors should be reinstated in accordance with approved plan.",  
 i: "Unauthorised openings/slab over existing floors for pipe ducts should be reinstated in accordance with approved plan.",  
 j: "Unauthorised installation of glass panels on external walls on upper floors should be removed and the configuration of external walls should be reinstated in accordance with approved plan.",  
 k: "Protective barrier/external wall should be reinstated in accordance with approved plan.",  
 l: "The access ramp for persons with a disability/ The accessible toilet for persons with a disability / The accessible urinal should be reinstated in accordance with approved plan.",  
 m: "Unauthorised hollow raised platforms within premises hindering the access for persons with a disability should be removed.",  
 n: "Internal staircase should be reinstated in accordance with approved plan.",  
 o: "Unauthorised supporting frames for suspending an air-conditioning/mechanical ventilation plants of weight more than 150 kg inside the premises should be removed.",  
 p: "Unauthorised external solid fence walls/mesh fences/railings/poles on roof should be removed.",  
 q: "Unauthorised slabs filling up approved cockloft/staircase voids should be removed and reinstated in accordance with approved plan.",  
 r: "Unauthorised cocklofts/intermediate floors/floor extensions should be removed.",  
 s: "Unauthorised staircases should be removed.",  
 t: "Unauthorised openings through structural slabs/walls should be rectified and reinstated in accordance with approved plan.",  
 u: "Unauthorised removal/partial removal/major alteration of structural members should be reinstated in accordance with approved plan.",  
 v: "UBWs in common areas resulting in obstruction of means of escape from the licensed premises/the building should be removed.",  
 },  
 ubw\_q3\_cat2\_4: {  
 a: "The accessible ramp/ accessible toilet for persons with a disability should be reinstated in accordance with the approved plan or constructed in accordance with “Design Manual Barrier Free Access 2008”.",  
 },  
 structural\_schnlhkinds\_q3\_cat2\_2a\_q: "The AP/RSE is required to check and confirm the structural adequacy of the existing floor in respect of the additional loads due to the following :",  
 structural\_schnlhkinds\_q3\_cat2\_2a: {  
 a: "Existing raised screed",  
 b: "Additional solid partition",  
 },  
 structural\_schnlhkinds\_q1\_cat3: {  
 a: "A structural justification should be submitted by AP/RSE to demonstrate that the the concerned premises on xx/F are structurally capable of withstanding the imposed load of 3 kPa for School use.",  
 },  
 frc\_q1\_q2a\_cat3: {  
 a: "Common internal corridor serving rooms or flats should be separated from rooms or flats by fire barriers having an fire resistance rating (FRR) of not less than that of \_\_\_\_\_ minutes.",  
 },  
 ubw\_q4\_cat3: {  
 a: "Structure within dedicated area / public passage which were excluded from GFA calculation in the original building design should be removed.",  
 },  
};  
export default CAT\_DESCRIPTIONS;

## File: bd-scs-backend-web/src/routes/advance-search/index.jsx

import { useTranslation } from 'react-i18next'  
import { Controller, useForm } from 'react-hook-form'  
import React from 'react'  
import { useNavigate, useLocation } from 'react-router-dom'  
import Button from '@mui/material/Button'  
import Stack from '@mui/material/Stack'  
import Divider from '@mui/material/Divider'  
import Grid from '@mui/material/Grid'  
import Box from '@mui/material/Box'  
import Checkbox from 'components/basic/Checkbox'  
import RadioGroup from '@mui/material/RadioGroup'  
import Radio from '@mui/material/Radio'  
import DatePicker from 'src/components/basic/DatePicker'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import OpenAddressDialogButton from 'components/OpenAddressDialogButton'  
import FormHelperText from '@mui/material/FormHelperText'  
import Autocomplete from 'components/basic/Autocomplete'  
import GradientButton from 'components/basic/Button/GradientButton'  
import SquareButton from 'components/basic/Button/SquareIconButton'  
import Typography from 'src/components/basic/Typography'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import usePersistData from 'src/hooks/usePersistData'  
import colors from 'src/constants/colors'  
import { Applications } from 'src/constants/options'  
import { Category, Nature } from 'components/ApplicationForm/constants'  
import CustomDatePicker from 'src/components/basic/DatePicker'  
import { use } from 'i18next'  
function AdvanceSearch() {  
 const navigate = useNavigate()  
 const location = useLocation()  
 const { isClickedBack } = location?.state || {}  
 const { t, i18n } = useTranslation(['advanceSearch'])  
 const [values, setValues] = usePersistData({ keyName: 'advanceSearch' })  
 React.useEffect(() => {  
 if (isClickedBack) setValues({})  
 }, [isClickedBack])  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 values: isClickedBack ? values : {},  
 })  
 const onSubmit = (data) => {  
 let searchQuery = ''  
 for (const key in data) {  
 if (data[key]) {  
 searchQuery +=  
 key === 'Category' || key === 'Nature'  
 ? `${key}=${data[key].id}&`  
 : `${key}=${data[key]}&`  
 }  
 }  
 setValues(data)  
 navigate(`result?${searchQuery}`)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy">  
 {t('Licensing Case Search')}  
 </Typography>  
 <ResponsiveForm>  
 {(isDesktop) => {  
 return (  
 <Grid  
 container  
 direction="column"  
 flexWrap="nowrap"  
 my={2}  
 >  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="LicensingCaseNo"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('licensingCaseNo')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="LiceningCaseId"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('licensingCaseId')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs container>  
 {}  
 <Grid item xs={10}>  
 <Controller  
 name="FileReference"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('fileReference')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {}  
 <Grid item xs={1}>  
 <Button  
 variant="contained"  
 sx={{  
 marginLeft: 1,  
 marginTop: 4,  
 backgroundColor: '#ffffff',  
 color: '#000000',  
 }}  
 onClick={() => handleBrowseClick()}  
 >  
 ...  
 </Button>  
 </Grid>  
 </Grid>  
 <Grid item xs>  
 <FormControl  
 title={t('LAFileReference')}  
 labelPlacement="top"  
 fullWidth  
 >  
 <Stack  
 direction="row"  
 gap={2}  
 alignItems="center"  
 >  
 <Controller  
 name="lAFileReference1"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 placeholder="Prefix"  
 sx={{ width: 140 }}  
 {...field}  
 />  
 )}  
 />  
 <Typography>{t('in')}</Typography>  
 <Controller  
 name="lAFileReference2"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <TextField  
 fullWidth  
 placeholder="Ref."  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </FormControl>  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="NameOfSchoolCN"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('schoolNameChi')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="NameOfSchoolEN"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('schoolNameEng')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs container>  
 {}  
 <Grid item xs={10}>  
 <Controller  
 name="SeniorCaseOfficerReceive"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'seniorCaseOfficerReceive'  
 )}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {}  
 <Grid item xs={1}>  
 <Button  
 variant="contained"  
 sx={{  
 marginLeft: 1,  
 marginTop: 4,  
 backgroundColor: '#ffffff',  
 color: '#000000',  
 }}  
 onClick={() => handleBrowseClick()}  
 >  
 ...  
 </Button>  
 </Grid>  
 </Grid>  
 <Grid item xs container>  
 {}  
 <Grid item xs={10}>  
 <Controller  
 name="SeniorCaseOfficerReply"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t(  
 'seniorCaseOfficerReply'  
 )}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {}  
 <Grid item xs={1}>  
 <Button  
 variant="contained"  
 sx={{  
 marginLeft: 1,  
 marginTop: 4,  
 backgroundColor: '#ffffff',  
 color: '#000000',  
 }}  
 onClick={() => handleBrowseClick()}  
 >  
 ...  
 </Button>  
 </Grid>  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="Category"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title="Category"  
 labelPlacement="top"  
 fullWidth={true}  
 error={invalid}  
 helperText={error?.message}  
 >  
 <Autocomplete  
 options={Category}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 placeholder="Please Select"  
 disableClearable  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="Nature"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title="Nature"  
 labelPlacement="top"  
 fullWidth={true}  
 error={invalid}  
 helperText={error?.message}  
 >  
 <Autocomplete  
 options={Nature}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 placeholder="Please Select"  
 disableClearable  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="Region"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('region')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Autocomplete  
 options={Object.keys(  
 Applications  
 ).map((key) => ({  
 id: Applications[key],  
 label: key,  
 }))}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disableClearable  
 placeholder={t(  
 'placeholder.pleaseSelect'  
 )}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 {}  
 <Grid container gap={2} alignItems="center">  
 {}  
 <Grid item xs={5}>  
 <Controller  
 name="ReceivedDateStart"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 title={t(  
 'receivedDate'  
 )}  
 labelPlacement="top"  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <CustomDatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {}  
 <Grid item xs="auto" marginTop={4}>  
 <Typography>{t('To')}</Typography>  
 </Grid>  
 {}  
 <Grid item xs={5} marginTop={4}>  
 <Controller  
 name="ReceivedDateEnd"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <CustomDatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 {  
}  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="district"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('district')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Autocomplete  
 disabled  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect'  
 ),  
 id: '',  
 },  
 {  
 label: 'Option 1',  
 id: '1',  
 },  
 {  
 label: 'Option 2',  
 id: '2',  
 },  
 ]}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disableClearable  
 placeholder={t(  
 'placeholder.pleaseSelect'  
 )}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="area"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('area')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Autocomplete  
 disabled  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect'  
 ),  
 id: '',  
 },  
 {  
 label: 'Option 1',  
 id: '1',  
 },  
 {  
 label: 'Option 2',  
 id: '2',  
 },  
 ]}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disableClearable  
 placeholder={t(  
 'placeholder.pleaseSelect'  
 )}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 {}  
 <Grid container gap={2} alignItems="center">  
 {}  
 <Grid item xs={5}>  
 <Controller  
 name="ActualReplyDateStart"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 title={t(  
 'actualReplyDate'  
 )}  
 labelPlacement="top"  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <CustomDatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {}  
 <Grid item xs="auto" marginTop={4}>  
 <Typography>{t('To')}</Typography>  
 </Grid>  
 {}  
 <Grid item xs={5} marginTop={4}>  
 <Controller  
 name="ActualReplyDateEnd"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <CustomDatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 </Grid>  
 <Grid item xs>  
 {}  
 <Grid container gap={2} alignItems="center">  
 {}  
 <Grid item xs={5}>  
 <Controller  
 name="TargetReplyDateStart"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 title={t(  
 'targetReplyDate'  
 )}  
 labelPlacement="top"  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <CustomDatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {}  
 <Grid item xs="auto" marginTop={4}>  
 <Typography>{t('To')}</Typography>  
 </Grid>  
 {}  
 <Grid item xs={5} marginTop={4}>  
 <Controller  
 name="TargetReplyDateEnd"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <CustomDatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 {}  
 <Grid container gap={2} alignItems="center">  
 {}  
 <Grid item xs={5}>  
 <Controller  
 name="DateOfAVPStart"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 title={t('dateOfAVP')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <CustomDatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {}  
 <Grid item xs="auto" marginTop={4}>  
 <Typography>{t('To')}</Typography>  
 </Grid>  
 {}  
 <Grid item xs={5} marginTop={4}>  
 <Controller  
 name="DateOfAVPEnd"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <CustomDatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 </Grid>  
 <Grid item xs>  
 {}  
 <Grid container gap={2} alignItems="center">  
 {}  
 <Grid item xs={5}>  
 <Controller  
 name="ExpiryDateLicenceStart"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 title={t(  
 'expiryDateLicence'  
 )}  
 labelPlacement="top"  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <CustomDatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {}  
 <Grid item xs="auto" marginTop={4}>  
 <Typography>{t('To')}</Typography>  
 </Grid>  
 {}  
 <Grid item xs={5} marginTop={4}>  
 <Controller  
 name="ExpiryDateLicenceEnd"  
 control={control}  
 render={({  
 field,  
 fieldState: {  
 invalid,  
 error,  
 },  
 }) => (  
 <FormControl  
 error={invalid}  
 helperText={  
 error?.message  
 }  
 fullWidth  
 >  
 <CustomDatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs container>  
 <Grid item xs={8}>  
 <Controller  
 name="Remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('remarks')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {}  
 <Grid  
 item  
 marginLeft={2}  
 marginTop={4}  
 xs={3}  
 display="flex"  
 alignItems="center"  
 >  
 <FormControlLabel  
 control={  
 <Controller  
 name="EApplicationCheckbox"  
 control={control}  
 render={({ field }) => (  
 <Checkbox {...field} />  
 )}  
 />  
 }  
 label={t('eApplication')}  
 />  
 </Grid>  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="AuditResult"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('auditResult')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <Autocomplete  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect'  
 ),  
 id: '',  
 },  
 {  
 label: 'Option 1',  
 id: '1',  
 },  
 {  
 label: 'Option 2',  
 id: '2',  
 },  
 ]}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disableClearable  
 placeholder={t(  
 'placeholder.pleaseSelect'  
 )}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="EKRSCode"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('EKRSCode')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs></Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 >  
 <Grid item xs>  
 <Controller  
 name="searchOptions"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('searchOptions')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <RadioGroup row>  
 <FormControlLabel  
 value="all"  
 control={<Radio />}  
 label={t(  
 'permissionOptions.all'  
 )}  
 />  
 <FormControlLabel  
 value="any"  
 control={<Radio />}  
 label={t(  
 'permissionOptions.any'  
 )}  
 />  
 <FormControlLabel  
 value="includeUnitSign"  
 control={<Radio />}  
 label={t(  
 'permissionOptions.includeUnitSign'  
 )}  
 />  
 <FormControlLabel  
 value="includeHistory"  
 control={<Radio />}  
 label={t(  
 'permissionOptions.includeHistory'  
 )}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {isDesktop && <Grid item xs />}  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={2}  
 >  
 <Controller  
 name="outstandingCasesOnly"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControlLabel  
 label={t('outstandingCasesOnly')}  
 control={<Checkbox />}  
 />  
 )}  
 />  
 <Controller  
 name="withReminderMoreThan"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <Stack  
 direction="row"  
 alignItems="center"  
 >  
 <Checkbox />  
 <Typography mr={1}>  
 {t('withReminderMoreThan')}  
 </Typography>  
 <Autocomplete  
 options={[  
 {  
 label: t(  
 'placeholder.pleaseSelect'  
 ),  
 id: '',  
 },  
 {  
 label: 'Option 1',  
 id: '1',  
 },  
 {  
 label: 'Option 2',  
 id: '2',  
 },  
 ]}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 value={  
 !field?.value?.id  
 ? null  
 : field?.value  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disableClearable  
 sx={{ minWidth: 120 }}  
 />  
 <Typography ml={1}>  
 {t('no')}  
 </Typography>  
 </Stack>  
 )}  
 />  
 </ResponsiveGrid>  
 <Divider sx={{ my: 3 }} />  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.search')}  
 </GradientButton>  
 </Grid>  
 )  
 }}  
 </ResponsiveForm>  
 </Box>  
 )  
}  
export default AdvanceSearch

## File: bd-scs-backend-web/src/routes/advance-search/result.jsx

import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import { useNavigate } from 'react-router-dom'  
import Box from '@mui/material/Box'  
import Button from '@mui/material/Button'  
import DataTable from 'components/basic/DataGrid'  
import Typography from 'src/components/basic/Typography'  
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined'  
import colors from 'src/constants/colors'  
function AdvanceSearch() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['advanceSearch'])  
 const query = new URLSearchParams(window.location.search)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 })  
 const onClickBack = () => {  
 navigate('/advance-search', { state: { isClickedBack: true } })  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy">  
 {t('Licensing Case Search')}  
 </Typography>  
 <Button  
 sx={{  
 color: 'rgb(59 130 246)',  
 cursor: 'pointer',  
 width: 'fit-content',  
 textTransform: 'none',  
 my: 3,  
 }}  
 onClick={onClickBack}  
 startIcon={<ArrowBackOutlinedIcon />}  
 >  
 {t('backToSearch')}  
 </Button>  
 <DataTable  
 api={`/cases?${query.toString()}`}  
 columns={[  
 {  
 field: '\_id',  
 headerName: 'Case No.',  
 width: 260,  
 },  
 {  
 field: 'Category',  
 headerName: 'Category',  
 width: 200,  
 },  
 {  
 field: 'Nature',  
 headerName: 'Nature',  
 width: 200,  
 },  
 {  
 field: 'EApplication',  
 headerName: 'EApp',  
 width: 150,  
 valueGetter: (value, row) => {  
 return row.SubmissionType == 'EAPPLICATION' ? 'Yes' : 'No'  
 },  
 },  
 {  
 field: 'TargetReplyDate',  
 headerName: 'Target Reply Date',  
 width: 300,  
 },  
 {  
 field: 'createdAt',  
 headerName: 'Submission Date',  
 width: 300,  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 color: '#fff',  
 },  
 '& .MuiDataGrid-columnHeaderCheckbox': {  
 color: '#fff',  
 },  
 '& .MuiButtonBase-root': {  
 color: '#000',  
 },  
 }}  
 onRowClick={(row) => {  
 navigate(`/applications/${row.row.application}/cases/${row.id}`)  
 }}  
 />  
 </Box>  
 )  
}  
export default AdvanceSearch

## File: bd-scs-backend-web/src/routes/application/add.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { useNavigate } from 'react-router-dom'  
import { Controller, useForm } from 'react-hook-form'  
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Box from '@mui/material/Box'  
import Radio from '@mui/material/Radio'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import RadioGroup from '@mui/material/RadioGroup'  
import Typography from 'components/basic/Typography'  
import Autocomplete from 'components/basic/Autocomplete'  
import FormControl from 'src/components/basic/FormControl'  
import GradientButton from 'src/components/basic/Button/GradientButton'  
import TextField from 'src/components/basic/TextField'  
import Button from 'src/components/basic/Button'  
import ApplicationForm from 'components/ApplicationForm/index'  
import FileReferenceDialog from 'components/Dialog/FileReferenceDialog'  
import colors from 'src/constants/colors'  
import { createApplication } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { Stack } from '@mui/material'  
const schema = yup.object({  
 ApplicationType: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 ApplicationNo: yup.string().nullable(),  
 FileReference: yup.string().required('Required'),  
 Region: yup.string().nullable(),  
 District: yup.string().nullable(),  
 Area: yup.string().nullable(),  
 AddressOfPremiseEN: yup.string().required('Required'),  
 AddressOfPremiseENFloor: yup.string().nullable(),  
 AddressOfPremiseENUnit: yup.string().nullable(),  
 AddressOfPremiseCN: yup.string().required('Required'),  
 AddressOfPremiseCNFloor: yup.string().nullable(),  
 AddressOfPremiseCNUnit: yup.string().nullable(),  
 NameOfSchoolEN: yup.string().required('Required'),  
 NameOfSchoolCN: yup.string().nullable(),  
 AgeOfStudent: yup.object().nullable(),  
 EstimatedNoOfStudent: yup.string().nullable(),  
 RelatedPremises: yup.array().of(  
 yup.object({  
 ID: yup.string(),  
 Type: yup.string(),  
 AdditionalInfo: yup.string(),  
 OPYear: yup.string(),  
 Address: yup.string(),  
 })  
 ),  
 RelatedPremise: yup.string().nullable(),  
 RelatedPremiseType: yup.string().nullable(),  
 RelatedPremiseAdditionalInfo: yup.string().nullable(),  
 RelatedPremiseOPYear: yup.string().nullable(),  
 RelatedPremiseAddress: yup.string().nullable(),  
 ApplicantNameEN: yup.string().required('Required'),  
 ApplicantNameCN: yup.string().nullable(),  
 ApplicantAddress: yup.string().required('Required'),  
 ApplicantTel: yup.string().nullable(),  
 ApplicantFax: yup.string().nullable(),  
 ContactPersonEN: yup.string().nullable(),  
 ContactPersonCN: yup.string().nullable(),  
 ApplicantEmail: yup.string().email().nullable(),  
 ContactPersonEmail: yup.string().email().nullable(),  
 ApplicantMobile: yup.string().nullable(),  
 ContactPersonTel: yup.string().nullable(),  
 APP13: yup.array().of(  
 yup.object({  
 NameEN: yup.string().nullable(),  
 NameCN: yup.string().nullable(),  
 RegistrationNumberPrefix: yup.string().nullable(),  
 RegistrationNumber: yup.string().nullable(),  
 Email: yup.string().email().nullable(),  
 Mobile: yup.string().nullable(),  
 })  
 ),  
 SelfCertification: yup.object({  
 NameEN: yup.string().nullable(),  
 NameCN: yup.string().nullable(),  
 RegistrationNumberPrefix: yup.string().nullable(),  
 RegistrationNumber: yup.string().nullable(),  
 Email: yup.string().email().nullable(),  
 Mobile: yup.string().nullable(),  
 }),  
})  
function CreateApplication() {  
 const navigate = useNavigate()  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const [fileReferenceDiaglogOpen, setFileReferenceDiaglogOpen] =  
 React.useState(false)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 defaultValues: {  
 RelatedPremises: [  
 {  
 ID: '',  
 Type: '',  
 AdditionalInfo: '',  
 OPYear: '',  
 Address: '',  
 },  
 ],  
 APP13: [  
 {  
 NameEN: '',  
 NameCN: '',  
 RegistrationNumberPrefix: '',  
 RegistrationNumber: '',  
 Email: '',  
 Mobile: '',  
 },  
 ],  
 },  
 })  
 console.log('errors', errors)  
 const mutation = useMutation({  
 mutationFn: (payload) => createApplication(payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 navigate(`/applications/${data?.\_id}/cases/add`)  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log('formData', formData)  
 const isSCH = formData?.ApplicationType?.id === 'SCH'  
 let payload = {  
 ...formData,  
 ApplicationType: formData?.ApplicationType?.id,  
 AgeOfStudent: isSCH ? formData?.AgeOfStudent?.id : undefined,  
 DescriptionOfSchool: isSCH  
 ? formData?.DescriptionOfSchool?.id  
 : undefined,  
 EstimatedNoOfStudent: isSCH ? formData?.EstimatedNoOfStudent : undefined,  
 }  
 mutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <ApplicationForm control={control} setValue={setValue} type="create" />  
 {  
}  
 <Button  
 sx={{ width: 'fit-content', alignSelf: 'flex-end', marginTop: 5 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 Save & Next  
 </Button>  
 </Box>  
 )  
}  
export default CreateApplication

## File: bd-scs-backend-web/src/routes/application/detail.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { useNavigate, useParams } from 'react-router-dom'  
import { Controller, useForm } from 'react-hook-form'  
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import useMediaQuery from '@mui/material/useMediaQuery'  
import VisibilityIcon from '@mui/icons-material/Visibility'  
import PrintIcon from '@mui/icons-material/Print'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Divider from '@mui/material/Divider'  
import Typography from 'components/basic/Typography'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Button from 'src/components/basic/Button'  
import DataTable from 'components/basic/DataGrid'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import ApplicationForm from 'components/ApplicationForm'  
import DownloadIcon from '@mui/icons-material/Download'  
import colors from 'src/constants/colors'  
import {  
 SchoolDeacriptionList,  
 AgeOfStudentList,  
} from 'components/ApplicationForm/constants'  
import { updateApplication, getApplication } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import AttachmentTable from 'components/AttachmentTable'  
import FileReferenceDialog from 'components/Dialog/FileReferenceDialog'  
import constants from 'src/constants'  
import UserPositionPicker from 'components/basic/UserPositionPicker'  
function ApplicationDetails() {  
 const { t } = useTranslation()  
 const navigate = useNavigate()  
 const { applicationId } = useParams()  
 const attachRef = React.useRef()  
 const queryClient = useQueryClient()  
 const [fileReferenceDiaglogOpen, setFileReferenceDiaglogOpen] =  
 React.useState(false)  
 const isDesktop = useMediaQuery('(min-width:800px)')  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { data, isLoading } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 setValue,  
 getValues,  
 } = useForm({  
 values: {  
 ...data,  
 AgeOfStudent: AgeOfStudentList.find(  
 (item) => item.id === data?.AgeOfStudent  
 ),  
 DescriptionOfSchool: SchoolDeacriptionList.find(  
 (item) => item.id === data?.DescriptionOfSchool  
 ),  
 },  
 })  
 const handleOnClickCreateCase = () => {  
 navigate(`/applications/${applicationId}/cases/add`)  
 }  
 const mutation = useMutation({  
 mutationFn: (payload) => updateApplication(applicationId, payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 if (getValues('attachments')?.[0]?.file) {  
 attachRef.current.submit()  
 } else {  
 queryClient.invalidateQueries(['getApplication', { applicationId }])  
 }  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const payload = {  
 ...formData,  
 AgeOfStudent: formData?.AgeOfStudent?.id,  
 DescriptionOfSchool: formData?.DescriptionOfSchool?.id,  
 }  
 mutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="text.secondary">  
 {t('Application Details')}  
 </Typography>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ApplicationNo"  
 control={control}  
 render={({ field }) => (  
 <FormControl  
 title={t('Application No.')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicationType"  
 control={control}  
 render={({ field }) => (  
 <FormControl  
 title={t('Application Type')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="FileReferenceDisplay"  
 control={control}  
 render={({ field }) => (  
 <FormControl  
 title={t('File Reference')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <Stack direction="row" gap={2}>  
 <TextField fullWidth {...field} disabled />  
 <Button  
 onClick={() => setFileReferenceDiaglogOpen(true)}  
 disabled={field.value?.length > 0}  
 >  
 {t('Search')}  
 </Button>  
 </Stack>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="AssignedGR"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('Assigned GR')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth  
 error={invalid}  
 helperText={error?.message}  
 required  
 >  
 <UserPositionPicker field={field} role="GR" />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="assignedSBS"  
 control={control}  
 render={({ field }) => (  
 <FormControl  
 title={t('Senior Case Officer (Receive)')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="assignedBS"  
 control={control}  
 render={({ field }) => (  
 <FormControl  
 title={t('Case Officer (Receive)')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="Region"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('Region')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField disabled {...field} fullWidth />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="District"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('District')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField disabled {...field} fullWidth />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid isDesktop={isDesktop} spacing={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="Area"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('Area')}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField disabled {...field} fullWidth />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </>  
 )}  
 </ResponsiveForm>  
 <ApplicationForm  
 control={control}  
 setValue={setValue}  
 getValues={getValues}  
 type="edit"  
 />  
 <Box  
 sx={{  
 display: 'flex',  
 justifyContent: 'flex-end',  
 gap: 2,  
 mt: 2,  
 mb: 4,  
 }}  
 >  
 <Button onClick={() => handleSubmit(onSubmit)()}>{t('Save')}</Button>  
 </Box>  
 <Divider sx={{ my: 4 }} />  
 <Typography variant="title2" color="text.secondary" sx={{ mt: 4 }}>  
 {t('Case Submission')}  
 </Typography>  
 <Button  
 sx={{ alignSelf: 'flex-end', mb: 2 }}  
 onClick={handleOnClickCreateCase}  
 >  
 {t('Create Child Case')}  
 </Button>  
 <Box height={500} minHeight={500} maxHeight={500}>  
 <DataTable  
 api={`/applications/${applicationId}/cases`}  
 disableRowSelectionOnClick  
 columns={[  
 {  
 field: 'efolio',  
 headerName: t('E-Folio'),  
 width: 300,  
 },  
 {  
 field: 'filePartNo',  
 headerName: t('File Part No.'),  
 width: 300,  
 },  
 {  
 field: 'licensingCaseId',  
 headerName: t('Licensing Case ID'),  
 width: 300,  
 },  
 {  
 field: 'fileReference',  
 headerName: t('File Reference'),  
 width: 300,  
 },  
 {  
 field: 'Category',  
 headerName: t('Category'),  
 width: 300,  
 },  
 {  
 field: 'Nature',  
 headerName: t('Nature'),  
 width: 300,  
 },  
 {  
 field: 'attachmentDescription',  
 headerName: t('Attachment Description'),  
 width: 300,  
 },  
 {  
 field: 'caseDescription.DescriptionOfCase',  
 headerName: t('Description of Case'),  
 width: 300,  
 },  
 {  
 field: 'Referrer.email',  
 headerName: t('Referrer'),  
 width: 300,  
 valueGetter: (params, row) => row?.Referrer?.Email,  
 },  
 {  
 field: 'ReceivedDate',  
 headerName: t('Received Date'),  
 width: 300,  
 },  
 {  
 field: 'TargetReplyDate',  
 headerName: t('Target Reply Date'),  
 width: 300,  
 },  
 {  
 field: 'ActualReplyDate',  
 headerName: t('Actual Reply Date'),  
 width: 300,  
 },  
 {  
 field: 'Remarks',  
 headerName: t('Remark'),  
 width: 300,  
 },  
 {  
 field: '\_id',  
 headerName: t('Case No.'),  
 width: 300,  
 },  
 {  
 field: 'createdAt',  
 headerName: t('Submission Date'),  
 width: 300,  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 }}  
 onRowClick={(row) => navigate(`cases/${row.id}`)}  
 />  
 </Box>  
 <Typography variant="title2" color="text.secondary" sx={{ mt: 4 }}>  
 {t('Application History')}  
 </Typography>  
 <AttachmentTable  
 data={data?.attachments?.filter((item) =>  
 ['APPLICATION'].includes(item.type)  
 )}  
 disabled  
 ref={attachRef}  
 type="APPLICATION"  
 applicationId={applicationId}  
 onSuccess={() => {  
 queryClient.invalidateQueries(['getApplication', { applicationId }])  
 }}  
 />  
 <Box height={500} minHeight={500} maxHeight={500}>  
 <DataTable  
 api={`/applications/${applicationId}/history`}  
 disableRowSelectionOnClick  
 columns={[  
 {  
 field: 'efolio',  
 headerName: t('E-Folio'),  
 width: 300,  
 },  
 {  
 field: 'filePartNo',  
 headerName: t('File Part No.'),  
 width: 300,  
 },  
 {  
 field: 'fileReference',  
 headerName: t('File Reference'),  
 width: 300,  
 },  
 {  
 field: 'receivedDate',  
 headerName: t('Received Date'),  
 width: 300,  
 },  
 {  
 field: 'type',  
 headerName: t('TYPE'),  
 width: 300,  
 },  
 {  
 field: '\_id',  
 headerName: t('Actions'),  
 width: 400,  
 renderCell: (params) => {  
 if (!params.row.file) return null  
 const fileUrl = `${constants.apiEndPoint}attachments/${params.row.\_id}`  
 const fileName = params.row.originalName || 'document'  
 const fileType = fileName.split('.').pop().toLowerCase()  
 const handlePreview = () => {  
 const previewableTypes = ['pdf', 'jpg', 'jpeg', 'png', 'gif']  
 if (previewableTypes.includes(fileType)) {  
 window.open(fileUrl, '\_blank')  
 } else {  
 setSnackbar({  
 type: 'info',  
 message: 'Preview not available for this file type',  
 })  
 }  
 }  
 const handlePrint = () => {  
 if (['pdf', 'jpg', 'jpeg', 'png', 'gif'].includes(fileType)) {  
 const printWindow = window.open(fileUrl, '\_blank')  
 printWindow?.addEventListener('load', () => {  
 setTimeout(() => printWindow.print(), 1000)  
 })  
 } else {  
 setSnackbar({  
 type: 'warning',  
 message: 'Printing not supported for this file type',  
 })  
 }  
 }  
 const handleDownload = () => {  
 const link = document.createElement('a')  
 link.href = fileUrl  
 link.download = fileName  
 document.body.appendChild(link)  
 link.click()  
 document.body.removeChild(link)  
 }  
 return (  
 <Box  
 sx={{  
 display: 'flex',  
 justifyContent: 'center',  
 alignItems: 'center',  
 width: '100%',  
 height: '100%',  
 gap: 2,  
 }}  
 >  
 {}  
 <Button  
 variant="contained"  
 size="small"  
 startIcon={<VisibilityIcon fontSize="small" />}  
 onClick={(e) => {  
 e.stopPropagation()  
 handlePreview()  
 }}  
 sx={{  
 minWidth: 90,  
 mx: 1,  
 }}  
 >  
 {t('Preview')}  
 </Button>  
 {}  
 <Button  
 variant="contained"  
 size="small"  
 startIcon={<PrintIcon fontSize="small" />}  
 onClick={(e) => {  
 e.stopPropagation()  
 handlePrint()  
 }}  
 sx={{  
 minWidth: 90,  
 mx: 1,  
 }}  
 >  
 {t('Print')}  
 </Button>  
 {}  
 <Button  
 variant="contained"  
 size="small"  
 color="primary"  
 startIcon={<DownloadIcon fontSize="small" />}  
 onClick={(e) => {  
 e.stopPropagation()  
 handleDownload()  
 }}  
 sx={{  
 minWidth: 90,  
 mx: 1,  
 }}  
 >  
 {t('Download')}  
 </Button>  
 </Box>  
 )  
 },  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 }}  
 />  
 </Box>  
 <FileReferenceDialog  
 open={fileReferenceDiaglogOpen}  
 onClose={() => setFileReferenceDiaglogOpen(false)}  
 onChange={(newFileRef) => {  
 setValue('FileReference', newFileRef.sysFileRefId)  
 setValue('FileReferenceDisplay', newFileRef.display)  
 }}  
 />  
 </Box>  
 )  
}  
export default ApplicationDetails

## File: bd-scs-backend-web/src/routes/application/index.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { useNavigate } from 'react-router-dom'  
import Box from '@mui/material/Box'  
import Radio from '@mui/material/Radio'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import RadioGroup from '@mui/material/RadioGroup'  
import Typography from 'components/basic/Typography'  
import Autocomplete from 'components/basic/Autocomplete'  
import FormControl from 'src/components/basic/FormControl'  
import GradientButton from 'src/components/basic/Button/GradientButton'  
import TextField from 'src/components/basic/TextField'  
import Button from 'src/components/basic/Button'  
import colors from 'src/constants/colors'  
import { request } from 'apis/index'  
function CreatePage() {  
 const navigate = useNavigate()  
 const [selected, setSelected] = React.useState()  
 const [applicationNo, setApplicationNo] = React.useState('') //66eadbf16009fea50b1fedc0  
 const handleOnClickNext = () => {  
 if (selected === 'new') {  
 navigate('/applications/add')  
 } else {  
 request({  
 method: 'GET',  
 url: `/applications?ApplicationNo=${applicationNo}`,  
 }).then((res) => {  
 navigate(`/applications/${res[0].\_id}`)  
 })  
 }  
 }  
 const isNextDisabled = () => {  
 if (selected === 'new') {  
 return false  
 } else {  
 return applicationNo === ''  
 }  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy">  
 Create Application / Case  
 </Typography>  
 <RadioGroup  
 value={selected}  
 onChange={(e) => {  
 setSelected(e.target.value)  
 // if (e.target.value === "new") {  
 // setApplicationNo("");  
 // }  
 }}  
 >  
 <FormControlLabel  
 value={'new'}  
 control={<Radio />}  
 label="New Application"  
 />  
 <>  
 <FormControlLabel  
 value={'detail'}  
 control={<Radio />}  
 label="Application Number"  
 />  
 <div style={{ flexDirection: 'row', display: 'flex' }}>  
 <TextField  
 sx={{ width: 300, ml: 3.5 }}  
 value={applicationNo}  
 onChange={(e) => setApplicationNo(e.target.value)}  
 disabled={!selected || selected === 'new'}  
 />  
 <Button sx={{ ml: 5, width: 250 }} onClick={() => {}}>  
 Search by file ref. number  
 </Button>  
 </div>  
 </>  
 <FormControlLabel  
 value={'others'}  
 control={<Radio />}  
 label="Others"  
 disabled  
 />  
 <Button  
 sx={{ mt: 3, width: 140 }}  
 disabled={isNextDisabled()}  
 onClick={handleOnClickNext}  
 >  
 Next  
 </Button>  
 </RadioGroup>  
 </Box>  
 )  
}  
export default CreatePage

## File: bd-scs-backend-web/src/routes/applicationDetails/index.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { useNavigate, useParams, useSearchParams } from 'react-router-dom'  
import { Controller, get, useForm } from 'react-hook-form'  
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Radio from '@mui/material/Radio'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import RadioGroup from '@mui/material/RadioGroup'  
import Divider from '@mui/material/Divider'  
import Typography from 'components/basic/Typography'  
import Autocomplete from 'components/basic/Autocomplete'  
import FormControl from 'src/components/basic/FormControl'  
import GradientButton from 'src/components/basic/Button/GradientButton'  
import TextField from 'src/components/basic/TextField'  
import Button from 'src/components/basic/Button'  
import DataTable from 'components/basic/DataGrid'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import ApplicationForm from 'components/ApplicationForm'  
import colors from 'src/constants/colors'  
import {  
 SchoolDeacriptionList,  
 AgeOfStudentList,  
} from 'components/ApplicationForm/constants'  
import { updateApplication, getApplication } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
function ApplicationDetails() {  
 const navigate = useNavigate()  
 const { applicationId } = useParams()  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 getValues,  
 } = useForm({  
 values: {  
 ...data,  
 AgeOfStudent: AgeOfStudentList.find(  
 (item) => item.id === data?.AgeOfStudent  
 ),  
 DescriptionOfSchool: SchoolDeacriptionList.find(  
 (item) => item.id === data?.DescriptionOfSchool  
 ),  
 },  
 })  
 const handleOnClickCreateCase = () => {  
 navigate(`/applications/${applicationId}/cases/add`)  
 }  
 const mutation = useMutation({  
 mutationFn: (payload) => updateApplication(applicationId, payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 console.log('error', error)  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log('Saving', formData)  
 const payload = {  
 ...formData,  
 AgeOfStudent: formData?.AgeOfStudent?.id,  
 DescriptionOfSchool: formData?.DescriptionOfSchool?.id,  
 }  
 mutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="text.secondary">  
 Application Details  
 </Typography>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="ApplicationNo"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Application No."  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="ApplicationType"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Application Type"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="SubmissionType"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Sumission Type"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="FileReference"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="File Reference"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="SeniorCaseOfficerReceive"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Senior Case Officer (Receive)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="caseOfficerReceive"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Case Officer (Receive)"  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </>  
 )}  
 </ResponsiveForm>  
 <ApplicationForm control={control} />  
 <Button  
 sx={{ alignSelf: 'flex-end' }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 Save  
 </Button>  
 <Divider sx={{ my: 4 }} />  
 <Typography variant="title2" color="text.secondary" sx={{ mt: 4 }}>  
 Case Submission  
 </Typography>  
 <Button  
 sx={{ alignSelf: 'flex-end', mb: 2 }}  
 onClick={handleOnClickCreateCase}  
 >  
 Create Child Case  
 </Button>  
 <Box height={500} minHeight={500} maxHeight={500}>  
 <DataTable  
 api={`/applications/${applicationId}/cases`}  
 disableRowSelectionOnClick  
 columns={[  
 {  
 field: 'eFolio',  
 headerName: 'E-Folio',  
 width: 300,  
 },  
 {  
 field: 'filePartNo',  
 headerName: 'File Part No.',  
 width: 300,  
 },  
 {  
 field: 'Category',  
 headerName: 'Category',  
 width: 300,  
 },  
 {  
 field: 'Nature',  
 headerName: 'Nature',  
 width: 300,  
 },  
 {  
 field: '\_id',  
 headerName: 'Case No.',  
 width: 300,  
 },  
 {  
 field: 'createdAt',  
 headerName: 'Submission Date',  
 width: 300,  
 },  
 {  
 field: 'description',  
 headerName: 'Description',  
 width: 300,  
 },  
 {  
 field: 'licensingCaseId',  
 headerName: 'Licensing Case ID',  
 width: 300,  
 },  
 {  
 field: 'fileReference',  
 headerName: 'File Reference',  
 width: 300,  
 },  
 {  
 field: 'ReceivedDate',  
 headerName: 'Received Date',  
 width: 300,  
 },  
 {  
 field: 'remark',  
 headerName: 'Remark',  
 width: 300,  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 }}  
 onRowClick={(row) => {  
 }}  
 />  
 </Box>  
 <Typography variant="title2" color="text.secondary" sx={{ mt: 4 }}>  
 Application History  
 </Typography>  
 <Button sx={{ alignSelf: 'flex-end', mb: 2 }}>Upload Document</Button>  
 <Box height={500} minHeight={500} maxHeight={500}>  
 <DataTable  
 api={`/applications/${applicationId}/submissions`}  
 disableRowSelectionOnClick  
 columns={[  
 {  
 field: 'eFolio',  
 headerName: 'E-Folio',  
 width: 300,  
 },  
 {  
 field: 'filePartNo',  
 headerName: 'File Part No.',  
 width: 300,  
 },  
 {  
 field: 'submissionDate',  
 headerName: 'Submission Date',  
 width: 300,  
 },  
 {  
 field: 'description',  
 headerName: 'Description',  
 width: 300,  
 },  
 {  
 field: 'licensingCaseId',  
 headerName: 'Licensing Case ID',  
 width: 300,  
 },  
 {  
 field: 'fileReference',  
 headerName: 'File Reference',  
 width: 300,  
 },  
 {  
 field: 'receivedDate',  
 headerName: 'Received Date',  
 width: 300,  
 },  
 {  
 field: 'remark',  
 headerName: 'Remark',  
 width: 300,  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 }}  
 onRowClick={(row) => {  
 }}  
 />  
 </Box>  
 </Box>  
 )  
}  
export default ApplicationDetails

## File: bd-scs-backend-web/src/routes/applicationDetails/old\_index.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import A2Form from 'components/FormDetail/a2'  
import B2Form from 'components/FormDetail/b2'  
import C2Form from 'components/FormDetail/c2'  
import E2Form from 'components/FormDetail/e2'  
import F2Form from 'components/FormDetail/f2'  
import G2Form from 'components/FormDetail/g2'  
import Cccaic2Form from 'components/FormDetail/cccaic2'  
import Cccaimh3Form from 'components/FormDetail/cccaimh3'  
import Jokc02Form from 'components/FormDetail/jokc02'  
import { TaskTitle } from 'src/constants/tasks/'  
import {  
 updateApplication,  
 getApplication,  
 getSubmissions,  
} from 'src/apis/application'  
import { getMyTasks } from 'src/apis/user'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { Box, Grid } from '@mui/material'  
import DataTable from 'components/basic/DataGrid'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import TaskBar from 'components/TaskBar'  
function ApplicationDetails() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const queryClient = useQueryClient()  
 const { applicationId } = useParams()  
 const {  
 data: applicationData,  
 error: applicationError,  
 isError: applicationIsError,  
 isLoading: applicationIsLoading,  
 } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 data: taskData,  
 error: taskError,  
 isError: taskIsError,  
 isLoading: taskIsLoading,  
 } = useQuery({  
 queryKey: ['myTask', { applicationId }],  
 queryFn: async () => await getMyTasks({ application: applicationId }),  
 })  
 console.log(taskData)  
 const {  
 data: submissionsData,  
 error: submissionsError,  
 isError: submissionsIsError,  
 isLoading: submissionsIsLoading,  
 } = useQuery({  
 queryKey: ['getSubmissions', { applicationId }],  
 queryFn: async () => await getSubmissions(applicationId),  
 })  
 const updateMutation = useMutation({  
 mutationFn: (payload) => updateApplication(applicationId, payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 queryClient.invalidateQueries({  
 queryKey: ['getApplication', { applicationId }],  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const renderFormContent = () => {  
 if (!applicationData) return null  
 switch (applicationData.ApplicationType) {  
 case 'NEWSCH': {  
 return (  
 <A2Form  
 updateMutation={updateMutation}  
 data={applicationData}  
 mode="view"  
 />  
 )  
 }  
 case 'EXTSCH': {  
 return (  
 <E2Form  
 updateMutation={updateMutation}  
 data={applicationData}  
 mode="view"  
 />  
 )  
 }  
 case 'NEWJOKC': {  
 return (  
 <Jokc02Form  
 updateMutation={updateMutation}  
 data={applicationData}  
 mode="view"  
 />  
 )  
 }  
 case 'NEWCCC': {  
 return (  
 <Cccaic2Form  
 updateMutation={updateMutation}  
 data={applicationData}  
 mode="view"  
 />  
 )  
 }  
 case 'NEWMUT': {  
 return (  
 <Cccaimh3Form  
 updateMutation={updateMutation}  
 data={applicationData}  
 mode="view"  
 />  
 )  
 }  
 }  
 }  
 return (  
 <Box pb={2} px={5} overflow="auto">  
 <TaskBar type="APPLICATION" />  
 <Grid container spacing={5} marginBottom={5}>  
 <Grid item xs={12} sm={6}>  
 <FormControl  
 title={'Application No.'}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <TextField  
 disabled  
 value={applicationData?.ApplicationNo}  
 />  
 </FormControl>  
 </Grid>  
 <Grid item xs={12} sm={6}>  
 <FormControl  
 title={'Application Type'}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <TextField  
 disabled  
 value={applicationData?.ApplicationType}  
 />  
 </FormControl>  
 </Grid>  
 <Grid item xs={12} sm={6}>  
 <FormControl  
 title={'Submission Type'}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <TextField  
 disabled  
 value={  
 applicationData?.EApplication  
 ? 'E-Application'  
 : 'Paper Application'  
 }  
 />  
 </FormControl>  
 </Grid>  
 <Grid item xs={12} sm={6}>  
 <FormControl  
 title={'Assigned GR'}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <TextField disabled />  
 </FormControl>  
 </Grid>  
 <Grid item xs={12} sm={6}>  
 <FormControl  
 title={'Assigned BS'}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <TextField disabled />  
 </FormControl>  
 </Grid>  
 </Grid>  
 <Grid  
 id="application-possible-submissions"  
 container  
 spacing={2}  
 marginBottom={5}  
 >  
 {applicationData?.ApplicationType == 'NEWSCH' && (  
 <Grid item xs={12} sm={6} md={3}>  
 <GradientButton  
 bgColor="navy"  
 sx={{  
 width: '100%',  
 mt: 2,  
 p: 5,  
 fontSize: 18,  
 borderRadius: 2,  
 }}  
 onClick={() =>  
 navigate(  
 `/applications/${applicationData?.\_id}/b2`  
 )  
 }  
 >  
 B2 Form  
 </GradientButton>  
 </Grid>  
 )}  
 {applicationData?.ApplicationType == 'NEWSCH' && (  
 <Grid item xs={12} sm={6} md={3}>  
 <GradientButton  
 bgColor="navy"  
 sx={{  
 width: '100%',  
 mt: 2,  
 p: 5,  
 fontSize: 18,  
 borderRadius: 2,  
 }}  
 onClick={() =>  
 navigate(  
 `/applications/${applicationData?.\_id}/c2`  
 )  
 }  
 >  
 C2 Form  
 </GradientButton>  
 </Grid>  
 )}  
 {applicationData?.ApplicationType == 'EXTSCH' && (  
 <Grid item xs={12} sm={6} md={3}>  
 <GradientButton  
 bgColor="navy"  
 sx={{  
 width: '100%',  
 mt: 2,  
 p: 5,  
 fontSize: 18,  
 borderRadius: 2,  
 }}  
 onClick={() =>  
 navigate(  
 `/applications/${applicationData?.\_id}/f2`  
 )  
 }  
 >  
 F2 Form  
 </GradientButton>  
 </Grid>  
 )}  
 {applicationData?.ApplicationType == 'EXTSCH' && (  
 <Grid item xs={12} sm={6} md={3}>  
 <GradientButton  
 bgColor="navy"  
 sx={{  
 width: '100%',  
 mt: 2,  
 p: 5,  
 fontSize: 18,  
 borderRadius: 2,  
 }}  
 onClick={() =>  
 navigate(  
 `/applications/${applicationData?.\_id}/g2`  
 )  
 }  
 >  
 G2 Form  
 </GradientButton>  
 </Grid>  
 )}  
 {['NEWSCH', 'EXTSCH', 'NEWJOKC', 'NEWCCC', 'NEWMUT'].includes(  
 applicationData?.ApplicationType  
 ) && (  
 <Grid item xs={12} sm={6} md={3}>  
 <GradientButton  
 bgColor="navy"  
 sx={{  
 width: '100%',  
 mt: 2,  
 p: 5,  
 fontSize: 18,  
 borderRadius: 2,  
 }}  
 onClick={() =>  
 navigate(  
 `/applications/${applicationData?.\_id}/submit-doc`  
 )  
 }  
 >  
 Submit Supporting Document  
 </GradientButton>  
 </Grid>  
 )}  
 {['NEWSCH', 'EXTSCH', 'NEWJOKC', 'NEWCCC', 'NEWMUT'].includes(  
 applicationData?.ApplicationType  
 ) && (  
 <Grid item xs={12} sm={6} md={3}>  
 <GradientButton  
 bgColor="navy"  
 sx={{  
 width: '100%',  
 mt: 2,  
 p: 5,  
 fontSize: 18,  
 borderRadius: 2,  
 }}  
 onClick={() =>  
 navigate(  
 `/applications/${applicationData?.\_id}/update-info`  
 )  
 }  
 >  
 Update Application and School Information  
 </GradientButton>  
 </Grid>  
 )}  
 {['NEWSCH', 'EXTSCH', 'NEWJOKC', 'NEWCCC', 'NEWMUT'].includes(  
 applicationData?.ApplicationType  
 ) && (  
 <Grid item xs={12} sm={6} md={3}>  
 <GradientButton  
 bgColor="navy"  
 sx={{  
 width: '100%',  
 mt: 2,  
 p: 5,  
 fontSize: 18,  
 borderRadius: 2,  
 }}  
 onClick={() =>  
 navigate(  
 `/applications/${applicationData?.\_id}/appoint-ap`  
 )  
 }  
 >  
 Appointment of Authorized Person (AP) and Structural  
 Engineer (RSE)  
 </GradientButton>  
 </Grid>  
 )}  
 {['NEWSCH', 'EXTSCH', 'NEWJOKC', 'NEWCCC', 'NEWMUT'].includes(  
 applicationData?.ApplicationType  
 ) && (  
 <Grid item xs={12} sm={6} md={3}>  
 <GradientButton  
 bgColor="navy"  
 sx={{  
 width: '100%',  
 mt: 2,  
 p: 5,  
 fontSize: 18,  
 borderRadius: 2,  
 }}  
 onClick={() =>  
 navigate(  
 `/applications/${applicationData?.\_id}/inspect-req`  
 )  
 }  
 >  
 Request for Inspection  
 </GradientButton>  
 </Grid>  
 )}  
 {['NEWSCH', 'EXTSCH', 'NEWJOKC', 'NEWCCC', 'NEWMUT'].includes(  
 applicationData?.ApplicationType  
 ) && (  
 <Grid item xs={12} sm={6} md={3}>  
 <GradientButton  
 bgColor="red"  
 sx={{  
 width: '100%',  
 mt: 2,  
 p: 5,  
 fontSize: 18,  
 borderRadius: 2,  
 }}  
 onClick={() =>  
 navigate(  
 `/applications/${applicationData?.\_id}/withdraw-app`  
 )  
 }  
 >  
 Withdraw Application  
 </GradientButton>  
 </Grid>  
 )}  
 </Grid>  
 <Box id="application-details" mb={5}>  
 <Typography variant="title2" color="text.secondary">  
 Application Details  
 </Typography>  
 <Box  
 mt={2}  
 display="flex"  
 flexDirection="column"  
 maxHeight="50vh"  
 >  
 {renderFormContent()}  
 </Box>  
 </Box>  
 <Box id="application-submissions">  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Typography variant="title2" color="text.secondary">  
 Submissions  
 </Typography>  
 </Stack>  
 <Box height={500} minHeight={500} maxHeight={500}>  
 <DataTable  
 api={`/applications/${applicationId}/submissions`}  
 disableRowSelectionOnClick  
 columns={[  
 {  
 field: 'SubmissionType',  
 headerName: 'Submission Type',  
 width: 300,  
 },  
 {  
 field: 'createdAt',  
 headerName: 'Submitted At',  
 width: 300,  
 valueGetter: (value, row) => {  
 return (  
 value &&  
 dayjs(value).format('DD-MM-YYYY HH:mm:ss')  
 )  
 },  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 mt: 2,  
 }}  
 onRowClick={(row) => {  
 navigate(`/submissions/${row.id}`)  
 }}  
 />  
 </Box>  
 </Box>  
 </Box>  
 )  
}  
export default ApplicationDetails

## File: bd-scs-backend-web/src/routes/building-search/index.jsx

import React from 'react'  
import { Controller, useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import { useLocation, useNavigate } from 'react-router-dom'  
import Box from '@mui/material/Box'  
import Button from '@mui/material/Button'  
import Divider from '@mui/material/Divider'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import Grid from '@mui/material/Grid'  
import Radio from '@mui/material/Radio'  
import RadioGroup from '@mui/material/RadioGroup'  
import Autocomplete from 'components/basic/Autocomplete'  
import GradientButton from 'components/basic/Button/GradientButton'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Typography from 'src/components/basic/Typography'  
import usePersistData from 'src/hooks/usePersistData'  
import DataTable from 'components/basic/DataGrid'  
import BuildingInformationDialog from 'components/Case/BuildingInformation/index'  
import DVTableTaskPage from 'src/routes/case/DvTable'  
import { BuildingType, PremiseType } from 'components/ApplicationForm/constants'  
import colors from 'src/constants/colors'  
function BuildingInformationSearch() {  
 const navigate = useNavigate()  
 const location = useLocation()  
 const { isClickedBack } = location?.state || {}  
 const { t, i18n } = useTranslation(['advanceSearch', 'common'])  
 const [values, setValues] = usePersistData({ keyName: 'buildingSearch' })  
 const [dialog, setDialog] = React.useState({ open: false })  
 const [dvDialog, setDvDialog] = React.useState({ open: false })  
 React.useEffect(() => {  
 if (isClickedBack) setValues({})  
 }, [isClickedBack])  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 values: isClickedBack ? values : {},  
 })  
 const onSubmit = (data) => {  
 let searchQuery = ''  
 for (const key in data) {  
 if (data[key]) {  
 searchQuery +=  
 key === 'Category' || key === 'Nature'  
 ? `${key}=${data[key].id}&`  
 : `${key}=${data[key]}&`  
 }  
 }  
 setValues(data)  
 navigate(`result?${searchQuery}`)  
 }  
 return (  
 <>  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy">  
 {t('Building Information Search')}  
 </Typography>  
 <ResponsiveForm>  
 {(isDesktop) => {  
 return (  
 <Grid container direction="column" flexWrap="nowrap" my={2}>  
 {}  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs container>  
 {}  
 <Grid item xs container>  
 <Grid item xs={10}>  
 <Controller  
 name="FileReference"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('fileReference')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs={2}>  
 <Button  
 variant="contained"  
 sx={{  
 marginLeft: 1,  
 marginTop: 4,  
 backgroundColor: '#ffffff',  
 color: '#000000',  
 }}  
 onClick={() => handleBrowseClick()}  
 >  
 ...  
 </Button>  
 </Grid>  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="Occupation Permit"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('Occupation Permit')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {}  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="Address"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('Address')}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {  
}  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="fireEngineeringApproach"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Designed by Fire Engineering Approach"  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <RadioGroup row>  
 <FormControlLabel  
 value="yes"  
 control={<Radio />}  
 label={t('yes')}  
 />  
 <FormControlLabel  
 value="no"  
 control={<Radio />}  
 label={t('no')}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="cinemaExists"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Cinema exists in the building"  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <RadioGroup row>  
 <FormControlLabel  
 value="yes"  
 control={<Radio />}  
 label={t('yes')}  
 />  
 <FormControlLabel  
 value="no"  
 control={<Radio />}  
 label={t('no')}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {}  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} mb={4}>  
 <Grid item xs>  
 <Controller  
 name="Type of Building"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Type of Building"  
 labelPlacement="top"  
 fullWidth={true}  
 error={invalid}  
 helperText={error?.message}  
 >  
 <Autocomplete  
 options={BuildingType}  
 getOptionLabel={(option) => option.label}  
 {...field}  
 value={!field?.value?.id ? null : field?.value}  
 onChange={(e, value) => field.onChange(value)}  
 placeholder="Please Select"  
 disableClearable  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="Type of Premise"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Type of Premise"  
 labelPlacement="top"  
 fullWidth={true}  
 error={invalid}  
 helperText={error?.message}  
 >  
 <Autocomplete  
 options={PremiseType}  
 getOptionLabel={(option) => option.label}  
 {...field}  
 value={!field?.value?.id ? null : field?.value}  
 onChange={(e, value) => field.onChange(value)}  
 placeholder="Please Select"  
 disableClearable  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 {  
}  
 <Divider sx={{ my: 3 }} />  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.search')}  
 </GradientButton>  
 </Grid>  
 )  
 }}  
 </ResponsiveForm>  
 <Box>  
 <DataTable  
 api={`${import.meta.env.VITE\_API\_ROOT}/building`}  
 staticData={[  
 {  
 fileReference: '12/4161/80',  
 address: '124-126 Castle Peak Road',  
 buildingType: 'Commercial',  
 premiseType: {},  
 occupationPermit: 'NK57/82',  
 fireEngineeringApproach: false,  
 cinemaExists: false,  
 buildingInfo: {},  
 },  
 ]}  
 disableRowSelectionOnClick={true}  
 columns={[  
 {  
 field: 'fileReference',  
 headerName: 'File Reference',  
 headerAlign: 'center',  
 flex: 1,  
 align: 'center',  
 },  
 {  
 field: 'address',  
 headerName: 'Address',  
 headerAlign: 'center',  
 flex: 2,  
 align: 'center',  
 },  
 {  
 field: 'buildingType',  
 headerName: 'Type of Building',  
 headerAlign: 'center',  
 flex: 1,  
 align: 'center',  
 },  
 {  
 field: 'premiseType',  
 headerName: 'Type of Premises',  
 headerAlign: 'center',  
 width: 200,  
 display: 'flex',  
 align: 'center',  
 renderCell: (params) => {  
 return (  
 <GradientButton onClick={() => setDialog({ open: true })}>  
 Building Information  
 </GradientButton>  
 )  
 },  
 },  
 {  
 field: 'occupationPermit',  
 headerName: 'Occupation Permit',  
 headerAlign: 'center',  
 flex: 1,  
 align: 'center',  
 },  
 {  
 field: 'fireEngineeringApproach',  
 headerName: 'Fire Engineering Approach',  
 headerAlign: 'center',  
 flex: 1,  
 align: 'center',  
 valueGetter: (field) => {  
 return field ? 'Y' : 'N'  
 },  
 },  
 {  
 field: 'cinemaExists',  
 headerName: 'Cinema Exists',  
 headerAlign: 'center',  
 flex: 1,  
 align: 'center',  
 valueGetter: (field) => {  
 return field ? 'Y' : 'N'  
 },  
 },  
 {  
 field: 'buildingInfo',  
 headerName: 'Building Information',  
 headerAlign: 'center',  
 width: 200,  
 display: 'flex',  
 align: 'center',  
 renderCell: (params) => {  
 return (  
 <GradientButton onClick={() => setDialog({ open: true })}>  
 Building Information  
 </GradientButton>  
 )  
 },  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 color: '#fff',  
 },  
 '& .MuiDataGrid-columnHeaderCheckbox': {  
 color: '#fff',  
 },  
 '& .MuiButtonBase-root': {  
 color: '#000',  
 },  
 }}  
 />  
 </Box>  
 </Box>  
 <BuildingInformationDialog  
 open={dialog.open}  
 onClose={() => setDialog({ open: false })}  
 />  
 <DVTableTaskPage  
 open={dvDialog.open}  
 onClose={() => setDvDialog({ open: false })}  
 />  
 </>  
 )  
}  
export default BuildingInformationSearch

## File: bd-scs-backend-web/src/routes/case/BsRecommendation/index.jsx

import { useTranslation } from 'react-i18next'  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Select from '@mui/material/Select'  
import MenuItem from '@mui/material/MenuItem'  
import Table from '@mui/material/Table';  
import TableBody from '@mui/material/TableBody';  
import TableCell from '@mui/material/TableCell';  
import TableContainer from '@mui/material/TableContainer';  
import TableHead from '@mui/material/TableHead';  
import TableRow from '@mui/material/TableRow';  
import Paper from '@mui/material/Paper';  
import Radio from '@mui/material/Radio'  
import RadioGroup from '@mui/material/RadioGroup'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import DatePicker from 'components/basic/DatePicker'  
import GradientButton from 'components/basic/Button/GradientButton'  
import DataTable from 'components/basic/DataGrid'  
import FormControl from 'components/basic/FormControl'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import Button from 'components/basic/Button'  
import TextField from 'components/basic/TextField'  
import Typography from 'components/basic/Typography'  
import TableA from 'components/Case/SCH/TableA'  
import TableB from 'components/Case/SCH/TableB'  
import TableC from 'components/Case/SCH/TableC'  
import InfoInput from 'components/Case/SCH/Info'  
import { InputAdornment } from '@mui/material'  
import colors from 'src/constants/colors'  
import { userContext } from 'src/context/UserProvider'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { updateCase } from 'src/apis/case'  
const schema = yup.object().shape({  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 remarks: yup.string().nullable(),  
 }),  
 q2: yup.object({  
 answer: yup.boolean().nullable(),  
 remarks: yup.string().nullable(),  
 })  
})  
function BsRecommendationTaskPage({ data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { t, i18n } = useTranslation(['P3Task'])  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 q1: {  
 answer: data?.q1?.answer?.toString(),  
 remarks: data?.q1?.remarks,  
 },  
 q2: {  
 answer: data?.q2?.answer?.toString(),  
 remarks: data?.q2?.remarks,  
 }  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const payload = {  
 bs\_recommendation: {  
 ...data,  
 ...formData,  
 },  
 }  
 mutation.mutate(payload)  
 console.log('formData', formData)  
 console.log('payload', payload)  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%' }}  
 bgcolor={colors.bgColor}  
 >  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Typography variant="title2" mx={3}>  
 BS Recommendation  
 </Typography>  
 <Box px={4} py={3}>  
 <Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`1. The content of document checklist is verified ?`}</Typography>  
 </Grid>  
 <Grid item>  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Controller  
 name="q1.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 key="q1.answer"  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value)  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`No`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="q1.remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 type="text"  
 placeholder="Remarks"  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </Grid>  
 </Grid>  
 <Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 wrap="nowrap"  
 >  
 <Grid item>  
 <Typography>{`2. TIssuing of Certificates/ Acceptance of compliance with building safety requirements in LoR is recommended?`}</Typography>  
 </Grid>  
 <Grid item>  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Controller  
 name="q2.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 key="q2.answer"  
 >  
 <RadioGroup  
 row  
 sx={{ flexWrap: 'nowrap' }}  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value)  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`No`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="q2.remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 type="text"  
 placeholder="Remarks"  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </Grid>  
 </Grid>  
 </Box>  
 <Button  
 sx={{ alignSelf: 'flex-end', mr: 2 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 Save  
 </Button>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default BsRecommendationTaskPage

## File: bd-scs-backend-web/src/routes/case/CCC/index.jsx

import { useTranslation } from 'react-i18next'  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Select from '@mui/material/Select'  
import MenuItem from '@mui/material/MenuItem'  
import Table from '@mui/material/Table';  
import TableBody from '@mui/material/TableBody';  
import TableCell from '@mui/material/TableCell';  
import TableContainer from '@mui/material/TableContainer';  
import TableHead from '@mui/material/TableHead';  
import TableRow from '@mui/material/TableRow';  
import Paper from '@mui/material/Paper';  
import GradientButton from 'components/basic/Button/GradientButton'  
import DataTable from 'components/basic/DataGrid'  
import FormControl from 'components/basic/FormControl'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import Button from 'components/basic/Button'  
import TextField from 'components/basic/TextField'  
import Typography from 'components/basic/Typography'  
import TableA from 'components/Case/SCH/TableA'  
import TableB from 'components/Case/SCH/TableB'  
import TableC from 'components/Case/SCH/TableC'  
import InfoInput from 'components/Case/SCH/Info'  
import colors from 'src/constants/colors'  
import { userContext } from 'src/context/UserProvider'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { updateCase } from 'src/apis/case'  
function CCCTaskPage({ data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId, caseId } = useParams()  
 const queryClient = useQueryClient()  
 console.log("data", data)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 reset,  
 setValue,  
 } = useForm({  
 values: {  
 ...data  
 }  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log("onSubmit", formData)  
 mutation.mutate({  
 documentChecklist: formData  
 })  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%', py: 4 }}  
 bgcolor={colors.bgColor}  
 >  
 <Box m={2}>  
 <Typography variant="h5" textAlign="center">{`Documents Checklist on Application for Self-certification (Child Care Centre)`}</Typography>  
 <Typography variant="h5" textAlign="center">{`for \*Child Care Centre Registration/Alteration of Registered Child Care Centre at f( )`}</Typography>  
 </Box>  
 {  
}  
 <Stack mx={2} mt={2} gap={2}>  
 <Typography variant="h6" sx={{ 'text-decoration': 'underline' }}>Part A: Desk Study (To be completed by SO/Lic)</Typography>  
 <TableA control={control} />  
 </Stack>  
 <Stack mx={2} mt={2} gap={2}>  
 <Typography variant="h6" sx={{ 'text-decoration': 'underline' }}>Part B: Document Check (To be completed by SO/Lic)</Typography>  
 <TableB control={control} />  
 </Stack>  
 <Stack mx={2} mt={2} gap={2}>  
 <Typography variant="h6" sx={{ 'text-decoration': 'underline' }}>Part C: Recommendation (To be completed by BS/Lic)</Typography>  
 <TableC control={control} />  
 </Stack>  
 {}  
 <Button  
 sx={{ mx: 2, mt: 4, alignSelf: 'flex-end' }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 Save  
 </Button>  
 </Stack>  
 )  
}  
export default SCHTaskPage

## File: bd-scs-backend-web/src/routes/case/DeskStudy/detail.jsx

import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useMutation, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import Stack from '@mui/material/Stack'  
import Form from 'components/Case/P2'  
import BuildingInformationDialog from 'components/Case/BuildingInformation/index'  
import colors from 'src/constants/colors'  
import { updateCase } from 'src/apis/case'  
import DVTableTaskDialog from '../DvTable'  
const schema = yup.object().shape({  
 q1: yup.object({  
 value: yup.array().of(  
 yup.object({  
 submission: yup.object().nullable(),  
 luReply: yup.string().nullable(),  
 })  
 ),  
 }),  
 q2: yup.object({  
 answer: yup.boolean().nullable(),  
 }),  
 q3: yup.object({  
 value: yup.array().of(  
 yup.object({  
 fileRef: yup.string().nullable(),  
 approvedDate: yup.date().nullable(),  
 acknowledgementDate: yup.date().nullable(),  
 })  
 ),  
 }),  
 q4: yup.object({  
 answer: yup.boolean().nullable(),  
 options\_cat2\_3a: yup.array().of(yup.string()).nullable(),  
 value\_cat2\_3a: yup.string().nullable(),  
 }),  
 q5: yup.object({  
 answer: yup.boolean().nullable(),  
 answer\_cat1: yup.boolean().nullable(),  
 }),  
 q6: yup.object({  
 answer: yup.boolean().nullable(),  
 }),  
 q7: yup.object({  
 answer: yup.boolean().nullable(),  
 }),  
 q8: yup.object({  
 answer: yup.string().nullable(),  
 }),  
})  
function P2TaskPage({ data, buildingData, dvData }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const [dialog, setDialog] = React.useState({ open: false })  
 const [dvDialog, setDvDialog] = React.useState({ open: false })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 getValues,  
 } = useForm({  
 resolver: yupResolver(schema),  
 defaultValues: {  
 q1: {  
 value: [  
 {  
 submission: null,  
 luReply: '',  
 },  
 ],  
 },  
 q2: {  
 answer: null,  
 },  
 q3: {  
 value: [  
 {  
 fileRef: '',  
 approvedDate: null,  
 acknowledgementDate: null,  
 },  
 ],  
 },  
 q4: {  
 answer: null,  
 options\_cat2\_3a: '',  
 value\_cat2\_3a: '',  
 },  
 q5: {  
 answer: null,  
 answer\_cat1: null,  
 },  
 q6: {  
 answer: null,  
 },  
 q7: {  
 answer: null,  
 },  
 q8: {  
 answer: null,  
 },  
 },  
 values: {  
 q1: {  
 value: data?.q1?.map((item) => {  
 return {  
 submission: item.submission,  
 luReply: item.luReply,  
 // nature: { value: item.nature },  
 // inspectionReport: { value: item.inspectionReport },  
 // licensePlan: { value: item.licensePlan },  
 }  
 }),  
 },  
 q2: {  
 answer: data?.q2?.answer,  
 },  
 q3: {  
 value: data?.q3?.map((item) => {  
 return {  
 fileRef: item.fileRef,  
 approvedDate: item.approvedDate  
 ? dayjs(item.approvedDate)  
 : null,  
 acknowledgementDate: item.acknowledgementDate  
 ? dayjs(item.approvedDate)  
 : null,  
 }  
 }),  
 },  
 q4: {  
 ...data?.q4,  
 },  
 q5: {  
 ...data?.q5,  
 },  
 q6: {  
 ...data?.q6,  
 },  
 q7: {  
 ...data?.q7,  
 },  
 q8: {  
 ...data?.q8,  
 },  
 },  
 })  
 console.log('errors', errors)  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log('formData', formData)  
 const payload = {  
 deck\_study: {  
 q1: formData.q1.value  
 .map((item) => {  
 if (!item.submission && !item.luReply) {  
 return null  
 }  
 return {  
 submission: item?.submission?.\_id,  
 luReply: item?.luReply,  
 }  
 })  
 .filter((item) => {  
 return item !== null  
 }),  
 q2: {  
 answer: formData.q2.answer,  
 },  
 q3: formData.q3.value  
 .map((item) => {  
 if (  
 !item.fileRef &&  
 !item.approvedDate &&  
 !item.acknowledgementDate  
 ) {  
 return null  
 }  
 return {  
 fileRef: item.fileRef,  
 approvedDate: item.approvedDate,  
 acknowledgementDate: item.acknowledgementDate,  
 }  
 })  
 .filter((item) => {  
 return item !== null  
 }),  
 q4: {  
 answer: formData.q4.answer,  
 options\_cat2\_3a: formData.q4.options\_cat2\_3a,  
 value\_cat2\_3a: formData.q4.options\_cat2\_3a?.includes('a')  
 ? formData.q4.value\_cat2\_3a  
 : null,  
 },  
 q5: {  
 answer: formData.q5.answer,  
 answer\_cat1: formData.q5.answer\_cat1,  
 },  
 q6: {  
 answer: formData.q6.answer,  
 },  
 q7: {  
 answer: formData.q7.answer,  
 },  
 q8: {  
 answer: formData.q8.answer,  
 },  
 },  
 }  
 mutation.mutate(payload)  
 console.log('formData', formData)  
 console.log('payload', payload)  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto' }}  
 bgcolor={colors.bgColor}  
 >  
 <Form  
 control={control}  
 onSubmit={() => handleSubmit(onSubmit)()}  
 setValue={setValue}  
 getValues={getValues}  
 openBuildingInformationDialog={() => setDialog({ open: true })}  
 openDvTableDialog={() => setDvDialog({ open: true })}  
 />  
 <BuildingInformationDialog  
 open={dialog.open}  
 onClose={() => setDialog({ open: false })}  
 data={buildingData}  
 />  
 <DVTableTaskDialog  
 open={dvDialog.open}  
 onClose={() => setDvDialog({ open: false })}  
 data={dvData}  
 />  
 </Stack>  
 )  
}  
export default P2TaskPage

## File: bd-scs-backend-web/src/routes/case/DvTable/index.jsx

import { useTranslation } from 'react-i18next'  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Select from '@mui/material/Select'  
import MenuItem from '@mui/material/MenuItem'  
import Table from '@mui/material/Table'  
import TableBody from '@mui/material/TableBody'  
import TableCell from '@mui/material/TableCell'  
import TableContainer from '@mui/material/TableContainer'  
import TableHead from '@mui/material/TableHead'  
import TableRow from '@mui/material/TableRow'  
import Paper from '@mui/material/Paper'  
import Button from 'components/basic/Button'  
import GradientButton from 'components/basic/Button/GradientButton'  
import DataTable from 'components/basic/DataGrid'  
import FormControl from 'components/basic/FormControl'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import TextField from 'components/basic/TextField'  
import Typography from 'components/basic/Typography'  
import TableA from 'components/Case/SCH/TableA'  
import TableB from 'components/Case/SCH/TableB'  
import TableC from 'components/Case/SCH/TableC'  
import InfoInput from 'components/Case/SCH/Info'  
import OccupantCapacityOfRoomsTable from 'components/Case/DV/OccupantCapacityOfRoomsTable'  
import AdequacyOfExitsFromStoreysTable from 'components/Case/DV/AdequacyOfExitsFromStoreysTable'  
import AdequacyOfStaircasesTable from 'components/Case/DV/AdequacyOfStaircasesTable'  
import colors from 'src/constants/colors'  
import { userContext } from 'src/context/UserProvider'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { updateCase } from 'src/apis/case'  
import Dialog from '@mui/material/Dialog'  
import DialogActions from '@mui/material/DialogActions'  
import DialogContent from '@mui/material/DialogContent'  
import DialogContentText from '@mui/material/DialogContentText'  
import DialogTitle from '@mui/material/DialogTitle'  
export const defOCRValue = {  
 floor: '',  
 compartment: '',  
 roomName: '',  
 roomUse: '',  
 areaId: '',  
 ufa: '',  
 FSFactor: '',  
 approvedCapacity: '',  
 approvedCapacity\_previousApprovedCapacity: '',  
 newOccupantCapacity: '',  
 revisedOccupantCapacity: '',  
 changeInPopulation: '',  
 noOfExit: '',  
 exitCapacity1: '',  
 exitCapacity2: '',  
 exitCapacity3: '',  
 exitCapacity4: '',  
 exitCapacity5: '',  
 exitCapacity6: '',  
 exitCapacity7: '',  
 exitCapacity8: '',  
 exitCapacity9: '',  
 exitCapacity10: '',  
 remark: '',  
}  
export const defAESValue = {  
 floor: '',  
 compartment: '',  
 capacity: '',  
 minimumNoOfExitDoorRouteRequired: '',  
 minimumNoOfExitDoorRouteProvided: '',  
 minimumWidthOfExitDoorTotalRequired: '',  
 minimumWidthOfExitDoorTotalProvided: '',  
 minimumWidthOfExiRouteTotalRequired: '',  
 minimumWidthOfExiRouteTotalProvided: '',  
 minimumWidthOfExitDoorEachRequired: '',  
 minimumWidthOfExitDoorEachProvided: '',  
 minimumWidthOfExiRouteEachRequired: '',  
 minimumWidthOfExiRouteEachProvided: '',  
 checkingResult: '',  
}  
export const defAOSValue = {  
 floor: '',  
 approvedPopulation: '',  
 totalAcceptedPopulation: '',  
 ST1: '',  
 ST2: '',  
}  
export default function DVTableTaskDialog({ open, onClose, data }) {  
 return (  
 <Dialog  
 open={open}  
 onClose={onClose}  
 aria-labelledby="alert-dialog-title"  
 aria-describedby="alert-dialog-description"  
 fullWidth  
 maxWidth="xl"  
 >  
 <DVTableTaskPage open={open} onClose={onClose} data={data} />  
 </Dialog>  
 )  
}  
export function DVTableTaskPage({ data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId, caseId } = useParams()  
 const queryClient = useQueryClient()  
 const mapValues = React.useMemo(() => {  
 if (data) {  
 const dynamicKeyCount =  
 data?.adequacyOfStaircases?.[0]?.staircasePopulation?.length || 0  
 const totalOccupantStaircase = {}  
 ;[  
 ...data?.adequacyOfStaircasesResult?.totalOccupantCapacity  
 ?.staircasePopulation,  
 ]  
 .slice(0, dynamicKeyCount)  
 .forEach(  
 (key, index) =>  
 (totalOccupantStaircase[`ST${index + 1}`] =  
 data?.adequacyOfStaircasesResult?.totalOccupantCapacity?.staircasePopulation[  
 index  
 ])  
 )  
 const totalDischargeStaircase = {}  
 ;[  
 ...data?.adequacyOfStaircasesResult?.totalDischargeValue  
 ?.staircasePopulation,  
 ]  
 .slice(0, dynamicKeyCount)  
 .forEach(  
 (key, index) =>  
 (totalDischargeStaircase[`ST${index + 1}`] =  
 data?.adequacyOfStaircasesResult?.totalDischargeValue?.staircasePopulation[  
 index  
 ])  
 )  
 const checkingResult = {}  
 ;[...data?.adequacyOfStaircasesResult?.checkingResult]  
 .slice(0, dynamicKeyCount)  
 .forEach(  
 (key, index) =>  
 (checkingResult[`ST${index + 1}`] =  
 data?.adequacyOfStaircasesResult?.checkingResult?.[index])  
 )  
 return {  
 ...data,  
 adequacyOfStaircases: data?.adequacyOfStaircases.map((item) => {  
 const target = {}  
 ;[...item.staircasePopulation].forEach(  
 (key, index) =>  
 (target[`ST${index + 1}`] = item.staircasePopulation[index])  
 )  
 return {  
 floor: item.floor,  
 approvedPopulation: item.approvedPopulation,  
 totalAcceptedPopulation: item.totalAcceptedPopulation,  
 ...target,  
 }  
 }),  
 adequacyOfStaircasesResult: {  
 totalOccupantCapacity: {  
 approvedPopulation:  
 data?.adequacyOfStaircasesResult?.totalOccupantCapacity  
 ?.approvedPopulation || '',  
 totalAcceptedPopulation:  
 data?.adequacyOfStaircasesResult?.totalOccupantCapacity  
 ?.totalAcceptedPopulation || '',  
 staircasePopulation: { ...totalOccupantStaircase },  
 },  
 totalDischargeValue: {  
 totalAcceptedPopulation:  
 data?.adequacyOfStaircasesResult?.totalDischargeValue  
 ?.totalAcceptedPopulation || '',  
 staircasePopulation: {  
 ...totalDischargeStaircase,  
 },  
 },  
 checkingResult: {  
 ...checkingResult,  
 },  
 },  
 }  
 }  
 }, [data])  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 reset,  
 setValue,  
 getValues,  
 } = useForm({  
 defaultValues: {  
 occupantCapacityOfRooms: [  
 {  
 ...defOCRValue,  
 },  
 ],  
 adequacyOfExitsFromStoreys: [  
 {  
 ...defAESValue,  
 },  
 ],  
 adequacyOfStaircases: [  
 {  
 ...defAOSValue,  
 },  
 ],  
 },  
 values: mapValues,  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log('onSubmit', formData)  
 const dynamicKeys = Object.keys(formData.adequacyOfStaircases?.[0]).filter(  
 (key) => key.includes('ST')  
 )  
 const payload = {  
 occupantCapacityOfRooms: formData.occupantCapacityOfRooms,  
 adequacyOfExitsFromStoreys: formData.adequacyOfExitsFromStoreys,  
 adequacyOfStaircases: formData.adequacyOfStaircases.map((item) => {  
 const staircasePopulation = dynamicKeys.map((key) => item[key] || '')  
 return {  
 floor: item.floor,  
 approvedPopulation: item.approvedPopulation,  
 totalAcceptedPopulation: item.totalAcceptedPopulation,  
 staircasePopulation: staircasePopulation,  
 }  
 }),  
 adequacyOfStaircasesResult: {  
 totalOccupantCapacity: {  
 approvedPopulation:  
 formData.adequacyOfStaircasesResult?.totalOccupantCapacity  
 ?.approvedPopulation || '',  
 totalAcceptedPopulation:  
 formData.adequacyOfStaircasesResult?.totalOccupantCapacity  
 ?.totalAcceptedPopulation || '',  
 staircasePopulation: Object.keys(  
 formData.adequacyOfStaircasesResult.totalOccupantCapacity  
 .staircasePopulation  
 ).map(  
 (key) =>  
 formData.adequacyOfStaircasesResult.totalOccupantCapacity  
 .staircasePopulation[key] || ''  
 ),  
 },  
 totalDischargeValue: {  
 totalAcceptedPopulation:  
 formData.adequacyOfStaircasesResult?.totalDischargeValue  
 ?.totalAcceptedPopulation || '',  
 staircasePopulation: Object.keys(  
 formData.adequacyOfStaircasesResult.totalDischargeValue  
 .staircasePopulation  
 ).map(  
 (key) =>  
 formData.adequacyOfStaircasesResult.totalDischargeValue  
 .staircasePopulation[key] || ''  
 ),  
 },  
 checkingResult: Object.keys(  
 formData.adequacyOfStaircasesResult?.checkingResult  
 ).map(  
 (key) => formData.adequacyOfStaircasesResult.checkingResult[key] || ''  
 ),  
 },  
 }  
 mutation.mutate({  
 dv: payload,  
 })  
 console.log('payload', payload)  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%', p: 4 }}  
 bgcolor={colors.bgColor}  
 >  
 <Box direction="column" my={4}>  
 <Typography variant="title2">DV Table</Typography>  
 </Box>  
 <Box my={2}>  
 <Typography variant="h6" sx={{ 'text-decoration': 'underline', mb: 2 }}>  
 Occupant Capacity of Rooms  
 </Typography>  
 <OccupantCapacityOfRoomsTable  
 control={control}  
 name="occupantCapacityOfRooms"  
 setValue={setValue}  
 />  
 </Box>  
 <Box my={2}>  
 <Typography variant="h6" sx={{ 'text-decoration': 'underline', mb: 2 }}>  
 Adequacy of exits from storeys  
 </Typography>  
 <AdequacyOfExitsFromStoreysTable  
 control={control}  
 name="adequacyOfExitsFromStoreys"  
 setValue={setValue}  
 />  
 </Box>  
 <Box my={2}>  
 <Typography variant="h6" sx={{ 'text-decoration': 'underline', mb: 2 }}>  
 Adequacy Of Staircases  
 </Typography>  
 <AdequacyOfStaircasesTable  
 control={control}  
 name="adequacyOfStaircases"  
 setValue={setValue}  
 data={getValues('adequacyOfStaircases')}  
 />  
 </Box>  
 <Button  
 sx={{ mx: 2, mt: 4, alignSelf: 'flex-end' }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 Save  
 </Button>  
 </Stack>  
 )  
}

## File: bd-scs-backend-web/src/routes/case/FRC/detail.jsx

import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useMutation, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs';  
import { useParams } from 'react-router-dom';  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import Stack from '@mui/material/Stack';  
import Form from 'components/Case/P5'  
import colors from 'src/constants/colors'  
import { updateCase } from 'src/apis/case'  
const schema = yup.object().shape({  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 }),  
 q2a: yup.object({  
 cat3: yup.array().of(yup.string()).nullable(),  
 cat3\_other: yup.string().nullable(),  
 }),  
 q2b: yup.object({  
 answer: yup.boolean().nullable(),  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_other: yup.string().nullable(),  
 }),  
 }),  
 q2: yup.object({  
 answer: yup.boolean().nullable(),  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_other: yup.string().nullable(),  
 }),  
 q2: yup.object({  
 answer: yup.boolean().nullable(),  
 cat2\_3b: yup.array().of(yup.string()).nullable(),  
 cat2\_3b\_other: yup.string().nullable(),  
 })  
 }),  
 q3: yup.object({  
 answer: yup.boolean().nullable(),  
 cat2\_3b: yup.array().of(yup.string()).nullable(),  
 cat2\_3b\_other: yup.string().nullable(),  
 }),  
 q4: yup.object({  
 answer: yup.boolean().nullable(),  
 cat2\_3b: yup.array().of(yup.string()).nullable(),  
 cat2\_3b\_other: yup.string().nullable(),  
 }),  
 q5: yup.object({  
 answer: yup.boolean().nullable(),  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_other: yup.string().nullable(),  
 }),  
 q6: yup.object({  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_other: yup.string().nullable(),  
 }),  
})  
function P5TaskPage({ data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const { handleSubmit, control, formState: { errors }, watch, setValue } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 q1: {  
 answer: data?.q1?.answer?.toString() || null,  
 q1: {  
 answer: data?.q1?.q1?.answer?.toString() || null,  
 },  
 q2a: {  
 cat3: data?.q1?.q2a?.cat3 || [],  
 cat3\_other: data?.q1?.q2a?.cat3\_other || "",  
 },  
 q2b: {  
 answer: data?.q1?.q2b?.answer?.toString() || null,  
 cat1: data?.q1?.q2b?.cat1 || [],  
 cat1\_other: data?.q1?.q2b?.cat1\_other || "",  
 },  
 },  
 q2: {  
 answer: data?.q2?.answer?.toString() || null,  
 q1: {  
 answer: data?.q2?.q1?.answer?.toString() || null,  
 cat1: data?.q2?.q1?.cat1 || [],  
 cat1\_other: data?.q2?.q1?.cat1\_other || "",  
 },  
 q2: {  
 answer: data?.q2?.q2?.answer?.toString() || null,  
 cat2\_3b: data?.q2?.q2?.cat2\_3b || [],  
 cat2\_3b\_other: data?.q2?.q2?.cat2\_3b\_other || "",  
 }  
 },  
 q3: {  
 answer: data?.q3?.answer?.toString() || null,  
 cat2\_3b: data?.q3?.cat2\_3b || [],  
 cat2\_3b\_other: data?.q3?.cat2\_3b\_other || "",  
 },  
 q4: {  
 answer: data?.q4?.answer?.toString() || null,  
 cat2\_3b: data?.q4?.cat2\_3b || [],  
 cat2\_3b\_other: data?.q4?.cat2\_3b\_other || "",  
 },  
 q5: {  
 answer: data?.q5?.answer?.toString() || null,  
 cat1: data?.q5?.cat1 || [],  
 cat1\_other: data?.q5?.cat1\_other || "",  
 },  
 q6: {  
 cat1: data?.q6?.cat1 || [],  
 cat1\_other: data?.q6?.cat1\_other || "",  
 },  
 }  
 })  
 console.log("errors", errors)  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const payload = {  
 frc: {  
 ...formData,  
 }  
 }  
 mutation.mutate(payload)  
 console.log("payload", payload)  
 }  
 return (  
 <Stack direction="column" sx={{ overflow: 'auto', height: '100%' }} bgcolor={colors.bgColor}>  
 <Form  
 control={control}  
 onSubmit={() => handleSubmit(onSubmit)()}  
 setValue={setValue}  
 />  
 </Stack>  
 )  
}  
export default P5TaskPage;

## File: bd-scs-backend-web/src/routes/case/MISC/detail.jsx

import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useMutation, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs';  
import { useParams } from 'react-router-dom';  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import Stack from '@mui/material/Stack';  
import Form from 'components/Case/P7'  
import colors from 'src/constants/colors'  
import { updateCase } from 'src/apis/case'  
const schema = yup.object().shape({  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_1: yup.array().of(yup.string()).nullable(),  
 cat1\_1\_other: yup.string().nullable(),  
 cat1\_other: yup.string().nullable(),  
 cat2\_1a: yup.array().of(yup.string()).nullable(),  
 cat2\_1a\_other: yup.string().nullable(),  
 cat2\_1b: yup.array().of(yup.string()).nullable(),  
 cat2\_1b\_other: yup.string().nullable(),  
 cat2\_2a: yup.array().of(yup.string()).nullable(),  
 cat2\_2a\_other: yup.string().nullable(),  
 cat2\_3b: yup.array().of(yup.string()).nullable(),  
 cat2\_3b\_other: yup.string().nullable(),  
 cat2\_4: yup.array().of(yup.string()).nullable(),  
 cat2\_4\_other: yup.string().nullable(),  
 cat3: yup.array().of(yup.string()).nullable(),  
 cat3\_other: yup.string().nullable(),  
 }),  
})  
function P7TaskPage({ data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const { handleSubmit, control, formState: { errors }, watch, setValue } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 q1: {  
 answer: data?.q1?.answer?.toString() || null,  
 cat1: data?.q1?.cat1 || [],  
 cat1\_1: data?.q1?.cat1\_1 || [],  
 cat1\_1\_other: data?.q1?.cat1\_1\_other || '',  
 cat1\_other: data?.q1?.cat1\_other || '',  
 cat2\_1a: data?.q1?.cat2\_1a || [],  
 cat2\_1a\_other: data?.q1?.cat2\_1a\_other || '',  
 cat2\_1b: data?.q1?.cat2\_1b || [],  
 cat2\_1b\_other: data?.q1?.cat2\_1b\_other || '',  
 cat2\_2a: data?.q1?.cat2\_2a || [],  
 cat2\_2a\_other: data?.q1?.cat2\_2a\_other || '',  
 cat2\_3b: data?.q1?.cat2\_3b || [],  
 cat2\_3b\_other: data?.q1?.cat2\_3b\_other || '',  
 cat2\_4: data?.q1?.cat2\_4 || [],  
 cat2\_4\_other: data?.q1?.cat2\_4\_other || '',  
 cat3: data?.q1?.cat3 || [],  
 cat3\_other: data?.q1?.cat3\_other || '',  
 }  
 }  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const payload = {  
 misc: formData  
 }  
 console.log("payload", payload)  
 mutation.mutate(payload)  
 }  
 return (  
 <Stack direction="column" sx={{ overflow: 'auto', height: '100%' }} bgcolor={colors.bgColor}>  
 <Form  
 control={control}  
 setValue={setValue}  
 onSubmit={() => handleSubmit(onSubmit)()}  
 />  
 </Stack>  
 )  
}  
export default P7TaskPage;

## File: bd-scs-backend-web/src/routes/case/MOE/detail.jsx

import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useMutation, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs';  
import { useParams } from 'react-router-dom';  
import React from 'react'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import Stack from '@mui/material/Stack';  
import Form from 'components/Case/P4'  
import colors from 'src/constants/colors'  
import { updateCase } from 'src/apis/case'  
import { yupResolver } from '@hookform/resolvers/yup';  
import \* as yup from 'yup';  
const schema = yup.object({  
 q1: yup.object({  
 q3: yup.object({  
 answer: yup.boolean().nullable(),  
 cat3: yup.array().of(yup.string()).nullable(),  
 cat3\_other: yup.string().nullable(),  
 }),  
 q4: yup.object({  
 numberOfKindergartenFloor: yup.number().typeError('A number is required').nullable(),  
 numberOfCommercialFloor: yup.number().typeError('A number is required').nullable(),  
 answer: yup.boolean().nullable(),  
 cat3: yup.array().of(yup.string()).nullable(),  
 cat3\_other: yup.string().nullable(),  
 }),  
 q5: yup.object({  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 }),  
 q2: yup.object({  
 answer: yup.boolean().nullable(),  
 cat3: yup.array().of(yup.string()).nullable(),  
 cat3\_other: yup.string().nullable(),  
 }),  
 }),  
 q9: yup.object({  
 cat3: yup.array().of(yup.string()).nullable(),  
 cat3\_other: yup.string().nullable(),  
 cat2\_4: yup.array().of(yup.string()).nullable(),  
 cat2\_4\_other: yup.string().nullable(),  
 }),  
 }),  
 q2: yup.object({  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 cat2\_4: yup.array().of(yup.string()).nullable(),  
 cat2\_4\_other: yup.string().nullable(),  
 }),  
 q2: yup.object({  
 answer: yup.boolean().nullable(),  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_other: yup.string().nullable(),  
 }),  
 q3: yup.object({  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_other: yup.string().nullable(),  
 cat2\_4: yup.array().of(yup.string()).nullable(),  
 cat2\_4\_other: yup.string().nullable(),  
 }),  
 }),  
 q3: yup.object({  
 answer: yup.boolean().nullable(),  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_other: yup.string().nullable(),  
 }),  
 q4: yup.object({  
 answer: yup.boolean().nullable(),  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_other: yup.string().nullable(),  
 }),  
})  
function P4TaskPage({ data, applicationType, deskStudy, buildingInformation, dv }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const { handleSubmit, control, formState: { errors }, watch, setValue } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 q1: {  
 q3: {  
 answer: data?.q1?.q3?.answer?.toString() || null,  
 cat3: data?.q1?.q3?.cat3 || [],  
 cat3\_other: data?.q1?.q3?.cat3\_other || "",  
 },  
 q4: {  
 numberOfKindergartenFloor: data?.q1?.q4?.numberOfKindergartenFloor,  
 numberOfCommercialFloor: data?.q1?.q4?.numberOfCommercialFloor,  
 answer: data?.q1?.q4?.answer?.toString() || null,  
 cat3: data?.q1?.q4?.cat3 || [],  
 cat3\_other: data?.q1?.q4?.cat3\_other || "",  
 },  
 q5: {  
 q1: {  
 answer: data?.q1?.q5?.q1?.answer?.toString() || null,  
 },  
 q2: {  
 answer: data?.q1?.q5?.q2?.answer?.toString() || null,  
 cat3: data?.q1?.q5?.q2?.cat3 || [],  
 cat3\_other: data?.q1?.q5?.q2?.cat3\_other || "",  
 },  
 },  
 q9: {  
 cat3: data?.q1?.q9?.cat3 || [],  
 cat3\_other: data?.q1?.q9?.cat3\_other || "",  
 cat2\_4: data?.q1?.q9?.cat2\_4 || [],  
 cat2\_4\_other: data?.q1?.q9?.cat2\_4\_other || "",  
 },  
 },  
 q2: {  
 q1: {  
 answer: data?.q2?.q1?.answer?.toString() || null,  
 cat2\_4: data?.q2?.q1?.cat2\_4 || [],  
 cat2\_4\_other: data?.q2?.q1?.cat2\_4\_other || "",  
 },  
 q2: {  
 answer: data?.q2?.q2?.answer?.toString() || null,  
 cat1: data?.q2?.q2?.cat1 || [],  
 cat1\_other: data?.q2?.q2?.cat1\_other || "",  
 },  
 q3: {  
 cat1: data?.q2?.q3?.cat1 || [],  
 cat1\_other: data?.q2?.q3?.cat1\_other || "",  
 cat2\_4: data?.q2?.q3?.cat2\_4 || [],  
 cat2\_4\_other: data?.q2?.q3?.cat2\_4\_other || "",  
 },  
 },  
 q3: {  
 answer: data?.q3?.answer?.toString() || null,  
 cat1: data?.q3?.cat1 || [],  
 cat1\_other: data?.q3?.cat1\_other || "",  
 },  
 q4: {  
 answer: data?.q4?.answer?.toString() || null,  
 cat1: data?.q4?.cat1 || [],  
 cat1\_other: data?.q4?.cat1\_other || "",  
 }  
 }  
 })  
 console.log("errors", errors)  
 React.useEffect(() => {  
 if (Object.keys(errors).length != 0) {  
 setSnackbar({  
 type: 'error',  
 message: "The information you entered is incorrect. Please check your information and try again."  
 })  
 }  
 }, [errors])  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const payload = {  
 moe: formData  
 }  
 mutation.mutate(payload)  
 console.log("payload", payload)  
 }  
 return (  
 <Stack direction="column" sx={{ overflow: 'auto', height: '100%' }} bgcolor={colors.bgColor} >  
 <Form  
 control={control}  
 setValue={setValue}  
 onSubmit={() => handleSubmit(onSubmit)()}  
 applicationType={applicationType}  
 dv={dv}  
 deskStudy={deskStudy}  
 buildingInformation={buildingInformation}  
 />  
 </Stack >  
 )  
}  
export default P4TaskPage;

## File: bd-scs-backend-web/src/routes/case/P1/detail.jsx

import { useTranslation } from 'react-i18next'  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Stack from '@mui/material/Stack'  
import Form from 'components/Case/P1'  
import colors from 'src/constants/colors'  
import { userContext } from 'src/context/UserProvider'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { createSiteInSpection } from 'src/apis/case'  
const schema = yup.object({  
 to: yup.string().required('Required'),  
 content: yup.string().required('Required'),  
})  
function P1TaskPage({ data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const [user] = React.useContext(userContext)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 reset,  
 setValue,  
 } = useForm({  
 defaultValues: {  
 docs: [  
 {  
 doc: null,  
 },  
 ],  
 },  
 values: {  
 docs: [  
 {  
 doc: {  
 ...data,  
 name: data?.originalname,  
 },  
 },  
 ],  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => createSiteInSpection(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 formData?.docs.forEach(async (item, index) => {  
 const formData = new FormData()  
 formData.append('file', item.doc)  
 if (!item?.doc) {  
 return setSnackbar({  
 type: 'error',  
 message: 'Please upload a document',  
 })  
 }  
 mutation.mutate(formData)  
 })  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%' }}  
 bgcolor={colors.bgColor}  
 >  
 <Form  
 control={control}  
 setValue={setValue}  
 onSubmit={() => handleSubmit(onSubmit)()}  
 />  
 </Stack>  
 )  
}  
export default P1TaskPage

## File: bd-scs-backend-web/src/routes/case/P8/detail.jsx

import { useTranslation } from 'react-i18next'  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Stack from '@mui/material/Stack'  
import Form from 'components/Case/P8'  
import Button from 'components/basic/Button'  
import colors from 'src/constants/colors'  
import { userContext } from 'src/context/UserProvider'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { createSiteInSpection, resetLetter } from 'src/apis/case'  
const schema = yup.object({  
 to: yup.string().required('Required'),  
 content: yup.string().required('Required'),  
})  
function P8TaskPage({ data, caseDetail }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId, caseId } = useParams()  
 const queryClient = useQueryClient()  
 const [user] = React.useContext(userContext)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 reset,  
 setValue,  
 } = useForm({  
 defaultValues: {  
 docs: [  
 {  
 doc: null,  
 },  
 ],  
 },  
 values: {  
 docs: [  
 {  
 doc: {  
 ...data,  
 name: data?.originalname,  
 },  
 },  
 ],  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => createSiteInSpection(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 formData?.docs.forEach(async (item, index) => {  
 const formData = new FormData()  
 formData.append('file', item.doc)  
 if (!item?.doc) {  
 return setSnackbar({  
 type: 'error',  
 message: 'Please upload a document',  
 })  
 }  
 mutation.mutate(formData)  
 })  
 }  
 console.log('Parent P8 Attachment', data)  
 const [loading, setLoading] = React.useState(false)  
 const handleResetLetter = async () => {  
 setLoading(true)  
 await resetLetter(caseId)  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setLoading(false)  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%' }}  
 bgcolor={colors.bgColor}  
 >  
 <Button disabled={loading} onClick={() => handleResetLetter()}>  
 {loading  
 ? 'Loading...'  
 : 'Reset custom text'}  
 </Button>  
 <Form  
 control={control}  
 setValue={setValue}  
 onSubmit={() => handleSubmit(onSubmit)()}  
 caseId={caseId}  
 applicationId={applicationId}  
 isScs={caseDetail?.ViaSCS ?? false}  
 attachments={data?.filter((item) => item.type === 'PREPARE\_LETTER')}  
 />  
 </Stack>  
 )  
}  
export default P8TaskPage

## File: bd-scs-backend-web/src/routes/case/P9/detail.jsx

import { useTranslation } from 'react-i18next'  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Stack from '@mui/material/Stack'  
import Form from 'components/Case/P8'  
import Button from 'components/basic/Button'  
import colors from 'src/constants/colors'  
import { userContext } from 'src/context/UserProvider'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { createSiteInSpection, issueLetter } from 'src/apis/case'  
import AttachmentTable from 'components/AttachmentTable'  
import { set } from 'lodash'  
import { type } from '@testing-library/user-event/dist/type'  
import { LETTERS, SELFCERT\_COMPLETE\_LETTERS } from 'src/constants/letters'  
const schema = yup.object({  
 to: yup.string().required('Required'),  
 content: yup.string().required('Required'),  
})  
function P9TaskPage({ data, caseDetail }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId, caseId } = useParams()  
 const queryClient = useQueryClient()  
 const [user] = React.useContext(userContext)  
 const [open, setOpen] = React.useState({ open: false, index: -1 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 reset,  
 setValue,  
 } = useForm({  
 defaultValues: {  
 docs: [  
 {  
 doc: null,  
 },  
 ],  
 },  
 values: {  
 docs: [  
 {  
 doc: {  
 ...data,  
 name: data?.originalname,  
 },  
 },  
 ],  
 },  
 })  
 const [loading, setLoading] = React.useState(false)  
 const handleIssueLetter = async () => {  
 let letter = LETTERS;  
 if(caseDetail?.ViaSCS){  
 letter = SELFCERT\_COMPLETE\_LETTERS;  
 }  
 console.log(  
 data?.filter((item) => item.type == 'PREPARE\_LETTER').length  
 )  
 if (data?.filter((item) => item.type == 'PREPARE\_LETTER').length != Object.keys(letter).length) {  
 setSnackbar({  
 type: 'error',  
 message: 'Please prepare all letters before issuing',  
 })  
 return  
 }  
 setLoading(true)  
 await issueLetter(caseId)  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setLoading(false)  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%' }}  
 bgcolor={colors.bgColor}  
 >  
 <Button disabled={loading} onClick={() => handleIssueLetter()}>  
 {loading  
 ? 'Loading...'  
 : data?.filter((item) => item.subType == 'ISSUE\_LETTER')  
 .length >= 1  
 ? 'Re-issue Letter'  
 : 'Issue Letter'}  
 </Button>  
 <div style={{ padding: '32px' }}>  
 <AttachmentTable  
 data={data.filter((item) => item.subType == 'ISSUE\_LETTER')}  
 applicationId={applicationId}  
 submissionCaseId={caseId}  
 onSuccess={(caseId) => {  
 queryClient.invalidateQueries([  
 'fetchCaseDetail',  
 { caseId },  
 ])  
 }}  
 onDialogClose={() => {  
 queryClient.invalidateQueries([  
 'fetchCaseDetail',  
 { caseId },  
 ])  
 }}  
 disabled  
 type="ISSUE\_LETTER"  
 subType={'ISSUE\_LETTER'}  
 max={2}  
 min={2}  
 hide={{  
 filePartNo: true,  
 }}  
 />  
 </div>  
 </Stack>  
 )  
}  
export default P9TaskPage

## File: bd-scs-backend-web/src/routes/case/SCH/index.jsx

import { useTranslation } from 'react-i18next'  
import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useQuery, useMutation, useQueryClient } from '@tanstack/react-query'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Select from '@mui/material/Select'  
import MenuItem from '@mui/material/MenuItem'  
import Table from '@mui/material/Table';  
import TableBody from '@mui/material/TableBody';  
import TableCell from '@mui/material/TableCell';  
import TableContainer from '@mui/material/TableContainer';  
import TableHead from '@mui/material/TableHead';  
import TableRow from '@mui/material/TableRow';  
import Paper from '@mui/material/Paper';  
import Radio from '@mui/material/Radio'  
import RadioGroup from '@mui/material/RadioGroup'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import DatePicker from 'components/basic/DatePicker'  
import GradientButton from 'components/basic/Button/GradientButton'  
import DataTable from 'components/basic/DataGrid'  
import FormControl from 'components/basic/FormControl'  
import HtmlEditor from 'components/basic/HtmlEditor'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import Button from 'components/basic/Button'  
import TextField from 'components/basic/TextField'  
import Typography from 'components/basic/Typography'  
import TableA from 'components/Case/SCH/TableA'  
import TableB from 'components/Case/SCH/TableB'  
import TableC from 'components/Case/SCH/TableC'  
import InfoInput from 'components/Case/SCH/Info'  
import { InputAdornment } from '@mui/material'  
import colors from 'src/constants/colors'  
import { userContext } from 'src/context/UserProvider'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { updateCase } from 'src/apis/case'  
const schema = yup.object().shape({  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 }),  
 q2: yup.object({  
 answer: yup.string().nullable(),  
 remarks: yup.string().nullable(),  
 }),  
})  
function SCHTaskPage({ data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { t, i18n } = useTranslation(['P3Task'])  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 q1: {  
 date: data?.q1?.date?.toString(),  
 },  
 q2: {  
 answer: data?.q2?.answer?.toString(),  
 remarks: data?.q2?.remarks,  
 },  
 q3: {  
 answer: data?.q3?.answer?.toString(),  
 remarks: data?.q3?.remarks,  
 },  
 q4: {  
 answer: data?.q4?.answer?.toString(),  
 remarks: data?.q4?.remarks,  
 },  
 q5: {  
 answer: data?.q5?.answer?.toString(),  
 remarks: data?.q5?.remarks,  
 },  
 q6: {  
 answer: data?.q6?.answer?.toString(),  
 remarks: data?.q6?.remarks,  
 },  
 q7: {  
 answer: data?.q7?.answer?.toString(),  
 remarks: data?.q7?.remarks,  
 },  
 q8: {  
 answer: data?.q8?.answer?.toString(),  
 remarks: data?.q8?.remarks,  
 },  
 q9: {  
 answer: data?.q9?.answer?.toString(),  
 remarks: data?.q9?.remarks,  
 remarks\_options: data?.q9?.remarks\_options,  
 }  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const payload = {  
 document\_checklist: {  
 ...data,  
 ...formData,  
 },  
 }  
 mutation.mutate(payload)  
 console.log('formData', formData)  
 console.log('payload', payload)  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%' }}  
 bgcolor={colors.bgColor}  
 >  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Typography variant="title2" mx={3}>  
 Document Checklist  
 </Typography>  
 <Box px={4} py={3}>  
 <Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`1. Date of \*Form SC(S)-1 / Form SC(S)-2`}</Typography>  
 </Grid>  
 <Grid item>  
 <Controller  
 name="q1.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 key={`${field.id}-${field.value}`}  
 >  
 <DatePicker  
 {...field}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 <Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`2. \*Form SC(S)-1 / Form SC(S)-2 is duly completed ?`}</Typography>  
 </Grid>  
 <Grid item>  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Controller  
 name="q2.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 key="q2.answer"  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(  
 e.target.value  
 )  
 }}  
 >  
 <FormControlLabel  
 value={`Yes`}  
 control={<Radio />}  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={`No`}  
 control={<Radio />}  
 label={`No`}  
 />  
 <FormControlLabel  
 value={`N/A`}  
 control={<Radio />}  
 label={`N/A`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="q2.remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 type="text"  
 placeholder="Remarks"  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </Grid>  
 </Grid>  
 <Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`3. Minor works submission records are found in order ? `}</Typography>  
 </Grid>  
 <Grid item>  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Controller  
 name="q3.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value)  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`No`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`N/A`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="q3.remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 type="text"  
 placeholder="Remarks"  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </Grid>  
 </Grid>  
 <Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`4. The plans and photos showing the unauthorised buildings works (UBWs) is submitted ?`}</Typography>  
 </Grid>  
 <Grid item>  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Controller  
 name="q4.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value)  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`No`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`N/A`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="q4.remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 type="text"  
 placeholder="Remarks"  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </Grid>  
 </Grid><Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`5. Structural assessment justifying the integrity of the UBWs is submitted ?`}</Typography>  
 </Grid>  
 <Grid item>  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Controller  
 name="q5.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value)  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`No`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`N/A`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="q5.remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 type="text"  
 placeholder="Remarks"  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </Grid>  
 </Grid><Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`6. Photos showing the completion of works is submitted ? `}</Typography>  
 </Grid>  
 <Grid item>  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Controller  
 name="q6.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value)  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`No`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`N/A`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="q6.remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 type="text"  
 placeholder="Remarks"  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </Grid>  
 </Grid><Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`7. Appendix A to PNAP APP-13 is submitted ?`}</Typography>  
 </Grid>  
 <Grid item>  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Controller  
 name="q7.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value)  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`No`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`N/A`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="q7.remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 type="text"  
 placeholder="Remarks"  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </Grid>  
 </Grid><Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`8. Revised plan incorporating this Department’s comments is submitted ?`}</Typography>  
 </Grid>  
 <Grid item>  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Controller  
 name="q8.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value)  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`No`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`N/A`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="q8.remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 type="text"  
 placeholder="Remarks"  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </Grid>  
 </Grid><Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`9. Other irregularities noted ?`}</Typography>  
 </Grid>  
 <Grid item>  
 <Stack direction="row" spacing={2} alignItems="center">  
 <Controller  
 name="q9.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value)  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`Yes`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`No`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`N/A`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="q9.remarks"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 type="text"  
 placeholder="Remarks"  
 {...field}  
 />  
 )}  
 />  
 </Stack>  
 </Grid>  
 </Grid>  
 </Box>  
 <Button  
 sx={{ alignSelf: 'flex-end', mr: 2 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 Save  
 </Button>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default SCHTaskPage

## File: bd-scs-backend-web/src/routes/case/Structural/bs/cccdetail.jsx

import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useMutation, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import Stack from '@mui/material/Stack'  
import Form from 'components/Case/P3ForCCC'  
import colors from 'src/constants/colors'  
import { updateCase } from 'src/apis/case'  
function P3TaskPage({ data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 defaultValues: {  
 q1: {  
 options\_cat1: [],  
 options\_cat2ia: [],  
 options\_cat2ib: [],  
 options\_cat2iia: [],  
 options\_cat2iv: [],  
 options\_cat3: [],  
 },  
 q3: {  
 options\_cat3: [],  
 },  
 q4: {  
 options\_cat2\_2a: [],  
 },  
 q5: {  
 options\_cat1\_1: [],  
 options\_cat1\_2: [],  
 options\_cat1\_3: [],  
 options\_cat1\_4: [],  
 },  
 q6: {  
 q1\_options\_cat1: [],  
 q1\_options\_cat2: [],  
 q1\_options\_cat3: [],  
 q2\_options\_cat1: [],  
 q2\_options\_cat2: [],  
 q2\_options\_cat3: [],  
 q3\_options\_cat1: [],  
 q3\_options\_cat2: [],  
 q3\_options\_cat3: [],  
 q4\_options\_cat1: [],  
 q4\_options\_cat2: [],  
 q4\_options\_cat3: [],  
 q5\_options\_cat1: [],  
 q5\_options\_cat2: [],  
 q5\_options\_cat3: [],  
 q6a\_options\_cat1: [],  
 q6a\_options\_cat2: [],  
 q6a\_options\_cat3: [],  
 q6b\_options\_cat1: [],  
 q6b\_options\_cat2: [],  
 q6b\_options\_cat3: [],  
 },  
 },  
 values: {  
 ...data,  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const payload = {  
 structural: {  
 q1: {  
 answer: formData.q1.answer  
 ? JSON.parse(formData.q1.answer)  
 : null,  
 options\_cat3: formData.q1.options\_cat3,  
 },  
 q2: {  
 answer: formData.q2.answer  
 ? JSON.parse(formData.q2.answer)  
 : null,  
 issue\_cert2: formData.q2.issue\_cert2,  
 not\_issue\_cert2: formData.q2.not\_issue\_cert2,  
 },  
 q3: {  
 answer: formData.q3.answer  
 ? JSON.parse(formData.q3.answer)  
 : null,  
 options\_cat2\_2a: formData.q3.options\_cat2\_2a,  
 other\_cat2\_2a: formData.q3.options\_cat2\_2a?.includes(  
 'other'  
 )  
 ? formData.q3.other\_cat2\_2a  
 : null,  
 },  
 q4: {  
 answer: formData.q4.answer  
 ? JSON.parse(formData.q4.answer)  
 : null,  
 },  
 q5: {  
 options\_cat1\_1: formData.q5.options\_cat1\_1,  
 options\_cat1\_2: formData.q5.options\_cat1\_2,  
 options\_cat1\_3: formData.q5.options\_cat1\_3,  
 options\_cat1\_4: formData.q5.options\_cat1\_4,  
 other\_cat1\_4: formData.q5.options\_cat1\_4?.includes('other')  
 ? formData.q5.other\_cat1\_4  
 : null,  
 },  
 q6: {  
 options: formData.q6.options,  
 other: formData.q6.options?.includes('other')  
 ? formData.q6.other  
 : null,  
 },  
 },  
 }  
 mutation.mutate(payload)  
 console.log('formData', formData)  
 console.log('payload', payload)  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%' }}  
 bgcolor={colors.bgColor}  
 >  
 <Form  
 control={control}  
 onSubmit={() => handleSubmit(onSubmit)()}  
 setValue={setValue}  
 />  
 </Stack>  
 )  
}  
export default P3TaskPage

## File: bd-scs-backend-web/src/routes/case/Structural/bs/detail.jsx

import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useMutation, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import Stack from '@mui/material/Stack'  
import Form from 'components/Case/P3'  
import colors from 'src/constants/colors'  
import { updateCase } from 'src/apis/case'  
const schema = yup.object().shape({  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 options\_cat3: yup.array().of(yup.string()).nullable(),  
 other\_cat3: yup.string().nullable(),  
 }),  
 q2: yup.object({  
 answer: yup.boolean().nullable(),  
 }),  
 q3: yup.object({  
 answer: yup.boolean().nullable(),  
 options\_cat2\_2a: yup.array().of(yup.string()).nullable(),  
 other\_cat2\_2a: yup.string().nullable(),  
 }),  
 q4: yup.object({  
 answer: yup.boolean().nullable(),  
 options\_cat1\_1: yup.array().of(yup.string()).nullable(),  
 options\_cat1\_2: yup.array().of(yup.string()).nullable(),  
 options\_cat1\_3: yup.array().of(yup.string()).nullable(),  
 options\_cat1\_4: yup.array().of(yup.string()).nullable(),  
 other\_cat1\_4: yup.string().nullable(),  
 }),  
 q5: yup.object({  
 options\_cat1: yup.array().of(yup.string()).nullable(),  
 other\_cat1: yup.string().nullable(),  
 }),  
 q6: yup.object({  
 options\_cat1: yup.array().of(yup.string()).nullable(),  
 other\_cat1: yup.string().nullable(),  
 }),  
})  
function P3TaskPage({ data, nature }) {  
 console.log('P3TaskPage', data)  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 getValues,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 ...data,  
 q1: {  
 ...data.q1,  
 answer: data?.q1?.answer?.toString(),  
 },  
 q2: {  
 ...data.q2,  
 answer: data?.q2?.answer?.toString(),  
 },  
 q3: {  
 ...data.q3,  
 answer: data?.q3?.answer,  
 },  
 q4: {  
 ...data.q4,  
 answer: data?.q4?.answer,  
 },  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log('formData', formData)  
 const payload = {  
 structural\_schnlhkinds: formData,  
 }  
 mutation.mutate(payload)  
 console.log('formData', formData)  
 console.log('payload', payload)  
 }  
 if(Object.keys(errors).length) {  
 console.log('errors', errors)  
 alert("errors");  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%' }}  
 bgcolor={colors.bgColor}  
 >  
 <Form  
 control={control}  
 onSubmit={() => handleSubmit(onSubmit)()}  
 setValue={setValue}  
 getValues={getValues}  
 nature={nature}  
 />  
 </Stack>  
 )  
}  
export default P3TaskPage

## File: bd-scs-backend-web/src/routes/case/Structural/se/cccdetail.jsx

import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useMutation, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import Stack from '@mui/material/Stack'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Radio from '@mui/material/Radio';  
import RadioGroup from '@mui/material/RadioGroup';  
import FormControlLabel from '@mui/material/FormControlLabel';  
import Form from 'components/Case/P3ForCCC'  
import Typography from 'components/basic/Typography'  
import FormControl from 'components/basic/FormControl';  
import Button from 'components/basic/Button'  
import TextField from 'components/basic/TextField'  
import QuestionHeader from 'components/Case/QuestionHeader'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import colors from 'src/constants/colors'  
import { updateCase } from 'src/apis/case'  
function P3TaskPage({ data }) {  
 const { t, i18n } = useTranslation(['P3Task'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 values: {  
 ...data,  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const payload = {  
 structural\_ccc\_se: {  
 q1\_a\_playground: JSON.parse(formData.q1\_a\_playground),  
 q1\_a\_childcarecentres: JSON.parse(formData.q1\_a\_childcarecentres),  
 q1\_b\_playground: JSON.parse(formData.q1\_b\_playground),  
 q1\_b\_childcarecentres: JSON.parse(formData.q1\_b\_childcarecentres),  
 q1\_c: formData.q1\_c,  
 },  
 }  
 mutation.mutate(payload)  
 console.log('formData', formData)  
 console.log('payload', payload)  
 }  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%' }}  
 bgcolor={colors.bgColor}  
 >  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Typography variant="title2" mx={3}>  
 Structural  
 </Typography>  
 <Box px={4} py={3}>  
 <Typography>{t('seccc.content')}</Typography>  
 <Grid container direction="row" mt={4} gap={4}>  
 <Grid item xs />  
 <Grid item sx={{ width: 300 }}>  
 <Typography>{t('seccc.title')}</Typography>  
 </Grid>  
 <Grid item sx={{ width: 300 }}>  
 <Typography>{t('seccc.title2')}</Typography>  
 </Grid>  
 </Grid>  
 <Grid container direction="row" gap={4} alignItems='center'>  
 <Grid item xs>  
 <Typography>{`(a) ${t('seccc.q1a.question')}`}</Typography>  
 </Grid>  
 <Grid item sx={{ width: 300 }}>  
 <Controller  
 name="q1\_a\_playground"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value);  
 onChange && onChange(e.target.value);  
 }}  
 >  
 <FormControlLabel value={true} control={<Radio />} label={t('yes', { ns: "common" })} />  
 <FormControlLabel value={false} control={<Radio />} label={t('no', { ns: "common" })} />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item sx={{ width: 300 }}>  
 <Controller  
 name="q1\_a\_childcarecentres"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value);  
 onChange && onChange(e.target.value);  
 }}  
 >  
 <FormControlLabel value={true} control={<Radio />} label={t('yes', { ns: "common" })} />  
 <FormControlLabel value={false} control={<Radio />} label={t('no', { ns: "common" })} />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 <Grid container direction="row" gap={4} mt={2} alignItems='center'>  
 <Grid item xs>  
 <Typography>{`(b) ${t('seccc.q1b.question')}`}</Typography>  
 </Grid>  
 <Grid item sx={{ width: 300 }}>  
 <Controller  
 name="q1\_b\_playground"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value);  
 onChange && onChange(e.target.value);  
 }}  
 >  
 <FormControlLabel value={true} control={<Radio />} label={t('yes', { ns: "common" })} />  
 <FormControlLabel value={false} control={<Radio />} label={t('no', { ns: "common" })} />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item sx={{ width: 300 }}>  
 <Controller  
 name="q1\_b\_childcarecentres"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(e.target.value);  
 onChange && onChange(e.target.value);  
 }}  
 >  
 <FormControlLabel value={true} control={<Radio />} label={t('yes', { ns: "common" })} />  
 <FormControlLabel value={false} control={<Radio />} label={t('no', { ns: "common" })} />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 <Grid container direction="row" gap={4} mt={2} alignItems='center'>  
 <Grid item xs>  
 <Typography>{`(c) ${t('seccc.q1c.question')}`}</Typography>  
 </Grid>  
 <Grid item sx={{ width: 300 }}>  
 <Controller  
 name="q1\_c"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item sx={{ width: 300 }} />  
 </Grid>  
 </Box>  
 <Button sx={{ alignSelf: 'flex-end', mr: 2 }} onClick={() => handleSubmit(onSubmit)()}>Save</Button>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default P3TaskPage

## File: bd-scs-backend-web/src/routes/case/Structural/se/detail.jsx

import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useMutation, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs'  
import { useParams } from 'react-router-dom'  
import React from 'react'  
import { useTranslation } from 'react-i18next'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import Stack from '@mui/material/Stack'  
import Grid from '@mui/material/Grid'  
import Form from 'components/Case/P3'  
import Button from 'components/basic/Button'  
import Box from '@mui/material/Box'  
import FormControl from 'components/basic/FormControl'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import Radio from '@mui/material/Radio'  
import RadioGroup from '@mui/material/RadioGroup'  
import Typography from 'components/basic/Typography'  
import TextField from 'components/basic/TextField'  
import InputAdornment from '@mui/material/InputAdornment'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import QuestionHeader from 'components/Case/QuestionHeader'  
import Q1QuestionBody from 'components/Case/P3/Q1'  
import colors from 'src/constants/colors'  
import { updateCase } from 'src/apis/case'  
const schema = yup.object().shape({  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 }),  
})  
function P3TaskPage({ data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { t, i18n } = useTranslation(['P3Task'])  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 q1: {  
 answer: data?.q1?.answer?.toString(),  
 kpa: data?.q1?.kpa,  
 },  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const payload = {  
 structural\_schnlhkinds: {  
 ...data,  
 ...formData,  
 },  
 }  
 mutation.mutate(payload)  
 console.log('formData', formData)  
 console.log('payload', payload)  
 }  
 const watchQ1Ans = watch('q1.answer')  
 return (  
 <Stack  
 direction="column"  
 sx={{ overflow: 'auto', height: '100%' }}  
 bgcolor={colors.bgColor}  
 >  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <Typography variant="title2" mx={3}>  
 Structural  
 </Typography>  
 <Box px={4} py={3}>  
 <Grid  
 container  
 direction="row"  
 alignItems="center"  
 justifyContent="space-between"  
 mt={2}  
 >  
 <Grid item>  
 <Typography>{`1. Premises on grade / suspended slab`}</Typography>  
 </Grid>  
 <Grid item>  
 <Controller  
 name="q1.answer"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 key={`${field.id}-${field.value}`}  
 >  
 <RadioGroup  
 row  
 {...field}  
 onChange={(e) => {  
 field.onChange(  
 e.target.value  
 )  
 }}  
 >  
 <FormControlLabel  
 value={true}  
 control={<Radio />}  
 label={`On grade`}  
 />  
 <FormControlLabel  
 value={false}  
 control={<Radio />}  
 label={`On suspended slab`}  
 />  
 </RadioGroup>  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </Grid>  
 {watchQ1Ans === 'false' && (  
 <Box sx={{ backgroundColor: '#fff', p: 2 }}>  
 <Controller  
 name="q1.kpa"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <TextField  
 InputProps={{  
 endAdornment: (  
 <InputAdornment position="end">  
 kpa  
 </InputAdornment>  
 ),  
 }}  
 type="number"  
 {...field}  
 />  
 )}  
 />  
 </Box>  
 )}  
 </Box>  
 <Button  
 sx={{ alignSelf: 'flex-end', mr: 2 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 Save  
 </Button>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 )  
}  
export default P3TaskPage

## File: bd-scs-backend-web/src/routes/case/UBW/detail.jsx

import { Controller, useForm, useFieldArray } from 'react-hook-form'  
import { useMutation, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs';  
import { useParams } from 'react-router-dom';  
import React from 'react'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import Stack from '@mui/material/Stack';  
import Form from 'components/Case/P6'  
import colors from 'src/constants/colors'  
import { updateCase } from 'src/apis/case'  
import { getValue } from '@testing-library/user-event/dist/utils';  
const schema = yup.object().shape({  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 options: yup.array().of(yup.string()).nullable(),  
 other: yup.string().nullable(),  
 q1: yup.object({  
 answer: yup.boolean().nullable(),  
 options: yup.array().of(yup.object({ value: yup.string(), type: yup.boolean() })).nullable(),  
 }),  
 q3: yup.object({  
 cat1\_1: yup.array().of(yup.string()).nullable(),  
 cat1\_1\_other: yup.string().nullable(),  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_other: yup.string().nullable(),  
 cat2\_4: yup.array().of(yup.string()).nullable(),  
 cat2\_4\_other: yup.string().nullable(),  
 }),  
 }),  
 q2: yup.object({  
 answer: yup.boolean().nullable(),  
 q1: yup.object({  
 option: yup.array().of(yup.string()).nullable(),  
 other: yup.string().nullable(),  
 }),  
 q2: yup.object({  
 answer: yup.boolean().nullable(),  
 cat2\_1a: yup.array().of(yup.string()).nullable(),  
 cat2\_1a\_other: yup.string().nullable(),  
 })  
 }),  
 q3: yup.object({  
 answer: yup.boolean().nullable(),  
 cat2\_4: yup.array().of(yup.string()).nullable(),  
 cat2\_4\_other: yup.string().nullable(),  
 }),  
 q4: yup.object({  
 answer: yup.boolean().nullable(),  
 cat1: yup.array().of(yup.string()).nullable(),  
 cat1\_other: yup.string().nullable(),  
 cat3: yup.array().of(yup.string()).nullable(),  
 cat3\_other: yup.string().nullable(),  
 }),  
 q5: yup.object({  
 answer: yup.boolean().nullable(),  
 }),  
})  
function P6TaskPage({ data }) {  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { caseId } = useParams()  
 const queryClient = useQueryClient()  
 const dataValues = React.useMemo(() => {  
 if (!data) return {}  
 return {  
 q1: {  
 answer: data?.q1?.answer?.toString() || null,  
 options: data?.q1?.options || [],  
 other: data?.q1?.other || '',  
 q1: {  
 answer: data?.q1?.q1?.answer?.toString() || null,  
 options: [  
 ...(data?.q1?.q1?.option\_yes || [])?.map((v) => ({ value: v, type: 'true' })),  
 ...(data?.q1?.q1?.option\_no || [])?.map((v) => ({ value: v, type: 'false' }))  
 ],  
 },  
 q3: {  
 cat1\_1: data?.q1?.q3?.cat1\_1 || [],  
 cat1\_1\_other: data?.q1?.q3?.cat1\_1\_other || '',  
 cat1: data?.q1?.q3?.cat1 || [],  
 cat1\_other: data?.q1?.q3?.cat1\_other || '',  
 cat2\_4: data?.q1?.q3?.cat2\_4 || [],  
 cat2\_4\_other: data?.q1?.q3?.cat2\_4\_other || '',  
 },  
 },  
 q2: {  
 answer: data?.q2?.answer?.toString() || null,  
 q1: {  
 option: data?.q2?.q1?.option || [],  
 other: data?.q2?.q1?.other || '',  
 },  
 q2: {  
 answer: data?.q2?.q2?.answer?.toString() || null,  
 cat2\_1a: data?.q2?.q2?.cat2\_1a || [],  
 cat2\_1a\_other: data?.q2?.q2?.cat2\_1a\_other || '',  
 }  
 },  
 q3: {  
 answer: data?.q3?.answer?.toString() || null,  
 cat2\_4: data?.q3?.cat2\_4 || [],  
 cat2\_4\_other: data?.q3?.cat2\_4\_other || '',  
 },  
 q4: {  
 answer: data?.q4?.answer?.toString() || null,  
 cat1: data?.q4?.cat1 || [],  
 cat1\_other: data?.q4?.cat1\_other || '',  
 cat3: data?.q4?.cat3 || [],  
 cat3\_other: data?.q4?.cat3\_other || '',  
 },  
 q5: {  
 answer: data?.q5?.answer?.toString() || null,  
 },  
 }  
 }, [data])  
 const { handleSubmit, control, formState: { errors }, watch, setValue, getValues } = useForm({  
 resolver: yupResolver(schema),  
 values: dataValues,  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 const q1\_1options = formData?.q1?.q1?.options || []  
 const payload = {  
 ubw: {  
 ...formData,  
 q1: {  
 ...formData.q1,  
 q1: {  
 answer: formData.q1.q1.answer,  
 option\_yes: q1\_1options.filter((o) => o.type === true).map((o) => o.value),  
 option\_no: q1\_1options.filter((o) => o.type === false).map((o) => o.value),  
 },  
 },  
 }  
 }  
 mutation.mutate(payload)  
 console.log("payload", payload)  
 }  
 return (  
 <Stack direction="column" sx={{ overflow: 'auto', height: '100%' }} bgcolor={colors.bgColor}>  
 <Form  
 control={control}  
 onSubmit={() => handleSubmit(onSubmit)()}  
 setValue={setValue}  
 />  
 </Stack>  
 )  
}  
export default P6TaskPage;

## File: bd-scs-backend-web/src/routes/case/add.jsx

import React, { useMemo, useEffect, useRef } from 'react'  
import { useTranslation } from 'react-i18next'  
import { useNavigate, useParams, useSearchParams } from 'react-router-dom'  
import { useForm } from 'react-hook-form'  
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import Radio from '@mui/material/Radio'  
import FormControlLabel from '@mui/material/FormControlLabel'  
import RadioGroup from '@mui/material/RadioGroup'  
import Typography from 'components/basic/Typography'  
import Autocomplete from 'components/basic/Autocomplete'  
import FormControl from 'src/components/basic/FormControl'  
import GradientButton from 'src/components/basic/Button/GradientButton'  
import TextField from 'src/components/basic/TextField'  
import Button from 'src/components/basic/Button'  
import AttachmentTable from 'components/AttachmentTable'  
import colors from 'src/constants/colors'  
import CaseForm from 'components/CaseForm'  
import { createCase, addAttachments } from 'src/apis/case'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { updateApplication, getApplication } from 'src/apis/application'  
import dayjs from 'dayjs'  
const schema = yup.object({  
 Category: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 Nature: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 ReceivedDate: yup.date().required('Required'),  
 caseOfficerReceive: yup.string().required('Required'),  
 attachments: yup.array(),  
 SubmissionType: yup.object({  
 id: yup.string().required('Required'),  
 }),  
})  
function CreateCase() {  
 const navigate = useNavigate()  
 const { applicationId } = useParams()  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const attachRef = useRef()  
 const {  
 data: applicationData,  
 error: applicationError,  
 isError: isApplicationError,  
 isLoading,  
 isApplicationLoading,  
 } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 getValues,  
 reset,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 FileReference: applicationData?.FileReferenceDisplay,  
 Region: applicationData?.Region,  
 District: applicationData?.District,  
 Area: applicationData?.Area,  
 caseOfficerReceive: applicationData?.assignedBS,  
 seniorCaseOfficerReceive: applicationData?.assignedSBS,  
 },  
 defaultValues: {  
 attachments: [  
 {  
 efolio: '',  
 filePartNo: '',  
 receivedDate: null,  
 file: null,  
 remarks: '',  
 },  
 ],  
 SubmissionType: {  
 id: 'PAPER',  
 label: 'Paper',  
 },  
 TargetReplyDate: dayjs(new Date()),  
 },  
 })  
 console.log('Errors', errors)  
 const onNavigateToApplicationDetail = () => {  
 navigate(`/applications/${applicationId}`)  
 }  
 const mutation = useMutation({  
 mutationFn: (payload) => createCase(applicationId, payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 if (attachRef.current?.getAttachments()?.[0]?.file) {  
 attachRef.current?.submit(data?.\_id)  
 } else {  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 navigate(`/applications/${applicationId}/cases/${data?.\_id}`)  
 }  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formValue) => {  
 const { attachments, caseNo, liceningCaseId, ...rest } = formValue  
 const payload = {  
 ...rest,  
 Category: rest?.Category?.id,  
 Nature: rest?.Nature?.id,  
 Region: rest?.Region?.id,  
 ThreeTierReqt: rest?.ThreeTierReqt?.id,  
 ObjectiontoLR: rest?.ObjectiontoLR?.id,  
 AuditResult: rest?.AuditResult?.id,  
 Referrer: {  
 ...rest?.Referrer,  
 GovDepartment: rest?.Referrer?.GovDepartment?.id,  
 },  
 SubmissionType: rest?.SubmissionType?.id,  
 }  
 mutation.mutate(payload)  
 }  
 useEffect(() => {  
 if (applicationData) {  
 setValue('applicationNumber', applicationData.ApplicationNo)  
 setValue('CaseOfficer', applicationData.assignedBS?.\_id)  
 setValue('CaseOfficerName', applicationData.assignedBS?.name)  
 }  
 }, [applicationData])  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <CaseForm control={control} type="add" />  
 <AttachmentTable  
 ref={attachRef}  
 type="CASE"  
 applicationId={applicationId}  
 onSuccess={(caseId) => {  
 if (!caseId) return  
 navigate(`/applications/${applicationId}/cases/${caseId}`)  
 }}  
 />  
 <Stack direction="row" justifyContent="space-between" sx={{ mt: 3 }}>  
 {applicationId && (  
 <Button  
 sx={{ width: 'fit-content' }}  
 onClick={onNavigateToApplicationDetail}  
 >  
 Back  
 </Button>  
 )}  
 <Button  
 sx={{ width: 'fit-content' }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 Save  
 </Button>  
 </Stack>  
 </Box>  
 )  
}  
export default CreateCase

## File: bd-scs-backend-web/src/routes/case/detail.jsx

import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs'  
import React from 'react'  
import { Controller, useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import { useParams, useNavigate } from 'react-router-dom'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import EMinutes from 'components/Case/Eminutes'  
import CaseForm from 'components/CaseForm'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import Button from 'components/basic/Button'  
import FormControl from 'components/basic/FormControl'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import TextField from 'components/basic/TextField'  
import Typography from 'components/basic/Typography'  
import GradientButton from 'components/basic/Button/GradientButton'  
import AttachmentTable from 'components/AttachmentTable'  
import { ExpandLessOutlined, ExpandMoreOutlined } from '@mui/icons-material'  
import { fetchCaseDetail, updateCase, addAttachments } from 'src/apis/case'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import {  
 AuditResult,  
 Category,  
 Nature,  
 ObjectiontoLR,  
 ReferrerGovDepartment,  
 Region,  
 ThreeTierReqt,  
} from 'components/ApplicationForm/constants'  
import { submissionTypes } from 'src/constants/options'  
function CaseDetailPage() {  
 const { t, i18n } = useTranslation(['p1Task'])  
 let navigate = useNavigate()  
 const attachRef = React.useRef()  
 const { applicationId, caseId } = useParams()  
 const queryClient = useQueryClient()  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['fetchCaseDetail', { caseId }],  
 queryFn: async () => await fetchCaseDetail(caseId),  
 })  
 console.log('Case data', data)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 getValues,  
 } = useForm({  
 values: {  
 applicationNumber: data?.application?.ApplicationNo,  
 caseNumber: data?.\_id,  
 ViaSCS: data?.ViaSCS == true,  
 SubmissionType: submissionTypes.find(  
 (item) => item.id === data?.SubmissionType  
 ),  
 applicationType: data?.application?.ApplicationType,  
 FileReference: data?.FileReference,  
 LAFileReference: data?.LAFileReference,  
 Category: Category.find((item) => item.id === data?.Category),  
 Nature: Nature.find((item) => item.id === data?.Nature),  
 Region: data?.Region,  
 District: data?.District,  
 Area: data?.Area,  
 ReceivedDate: data?.ReceivedDate ? dayjs(data?.ReceivedDate) : null,  
 caseOfficerReceive: data?.caseOfficerReceive,  
 seniorCaseOfficerReceive: data?.seniorCaseOfficerReceive,  
 caseOfficerReply: data?.caseOfficerReply,  
 seniorCaseOfficerReply: data?.seniorCaseOfficerReply,  
 SubstantialReplyDate: data?.SubstantialReplyDate  
 ? dayjs(data?.SubstantialReplyDate)  
 : null,  
 ActualReplyDate: data?.ActualReplyDate  
 ? dayjs(data?.ActualReplyDate)  
 : null,  
 CaseOfficer: data?.assignedBS?.\_id,  
 CaseOfficerName: data?.assignedBS?.name,  
 CaseOfficerPost: data?.assignedBS?.position,  
 TargetReplyDate: data?.TargetReplyDate  
 ? dayjs(data?.TargetReplyDate)  
 : null,  
 ThreeTierReqt: ThreeTierReqt.find(  
 (item) => item.id === data?.ThreeTierReqt  
 ),  
 ObjectiontoLR: ObjectiontoLR.find(  
 (item) => item.id === data?.ObjectiontoLR  
 ),  
 AuditResult: AuditResult.find((item) => item.id === data?.AuditResult),  
 Remarks: data?.Remarks,  
 Referrer: {  
 ...data?.Referrer,  
 GovDepartment: ReferrerGovDepartment.find(  
 (item) => item.id === data?.Referrer?.GovDepartment  
 ),  
 },  
 Reminders: data?.Reminders,  
 caseDescription: {  
 efolio: data?.caseDescription?.efolio,  
 filePartNo: data?.caseDescription?.filePartNo,  
 description: data?.caseDescription?.description,  
 remarks: data?.caseDescription?.remarks,  
 },  
 },  
 })  
 const [caseProgressExpanded, setCaseProgressExpanded] = React.useState(true)  
 const [caseDetailsExpanded, setCaseDetailsExpanded] = React.useState(true)  
 const mutation = useMutation({  
 mutationFn: (payload) => updateCase(caseId, payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 if (getValues('attachments')?.[0]?.file) {  
 attachRef.current.submit(data?.\_id)  
 } else {  
 queryClient.invalidateQueries(['fetchCaseDetail', { caseId }])  
 }  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onSubmit = (formValue) => {  
 const {  
 applicationNumber,  
 caseNumber,  
 submissionType,  
 applicationType,  
 liceningCaseId,  
 attachments,  
 ...rest  
 } = formValue  
 const payload = {  
 ...rest,  
 Category: rest?.Category?.id,  
 Nature: rest?.Nature?.id,  
 ThreeTierReqt: rest?.ThreeTierReqt?.id,  
 ObjectiontoLR: rest?.ObjectiontoLR?.id,  
 AuditResult: rest?.AuditResult?.id,  
 Referrer: {  
 ...rest?.Referrer,  
 GovDepartment: rest?.Referrer?.GovDepartment?.id,  
 },  
 SubmissionType: rest?.SubmissionType?.id,  
 }  
 mutation.mutate(payload)  
 }  
 const progressTypes = [  
 ...new Set(data?.tasks.map((item) => item.progressType)),  
 ]  
 return (  
 <Box p={4} overflow="auto">  
 {}  
 <Stack  
 direction="row"  
 alignItems="right"  
 justifyContent="right"  
 position="sticky"  
 top={0}  
 zIndex={1}  
 gap={2}  
 sx={{ my: 2 }}  
 >  
 <GradientButton  
 bgColor="navy"  
 onClick={() =>  
 window.open(  
 `  
 /applications/${applicationId}  
 `,  
 '\_blank',  
 'rel=noopener,left=100,top=100,width=680,height=680 noreferrer'  
 )  
 }  
 >  
 Application  
 </GradientButton>  
 <GradientButton  
 bgColor="greenBlue"  
 onClick={() =>  
 window.open(  
 `  
 /applications/${applicationId}/cases/${caseId}  
 `,  
 '\_blank',  
 'rel=noopener,left=100,top=100,width=680,height=680 noreferrer'  
 )  
 }  
 >  
 Case  
 </GradientButton>  
 {  
}  
 </Stack>  
 <Stack>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Grid container direction="column">  
 <ResponsiveGrid container direction="row" alignItems="center">  
 <ResponsiveGrid  
 item  
 container  
 isDesktop={isDesktop}  
 alignItems="center"  
 xs  
 sx={{ py: 2 }}  
 >  
 <Typography  
 variant="title2"  
 color="text.secondary"  
 onClick={() => setCaseProgressExpanded((prev) => !prev)}  
 sx={{ cursor: 'pointer' }}  
 >  
 {`${t('caseProgress', {  
 ns: 'commonTask',  
 })}`}{' '}  
 {caseProgressExpanded ? (  
 <ExpandMoreOutlined />  
 ) : (  
 <ExpandLessOutlined />  
 )}  
 </Typography>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 item  
 container  
 isDesktop={isDesktop}  
 alignItems="center"  
 justifyContent="flex-end"  
 xs  
 >  
 <Typography sx={{ color: '#4caf50', mr: 1 }}>{`${10} ${t(  
 'dayLeft'  
 )}`}</Typography>  
 <Typography sx={{ color: '#9e9e9e' }}>{`(${dayjs().format(  
 'DD/MM/YYYY'  
 )})`}</Typography>  
 </ResponsiveGrid>  
 </ResponsiveGrid>  
 {caseProgressExpanded && (  
 <Box maxHeight="38vh" overflow="auto">  
 {progressTypes.map((progressType) => {  
 return (  
 <Box>  
 <Typography  
 variant="title2"  
 color="text.secondary"  
 sx={{ cursor: 'pointer' }}  
 >  
 {`${progressType} Progress`}  
 </Typography>  
 <ResponsiveGrid container isDesktop={isDesktop}>  
 {data?.tasks  
 .filter(  
 (item) => item.progressType === progressType  
 )  
 .map((item) => {  
 return (  
 <Grid  
 item  
 xs={3}  
 sx={{  
 px: 1,  
 pb: 2,  
 }}  
 key={item?.\_id}  
 >  
 <CaseProgressItem  
 {...item}  
 onClick={(taskId) =>  
 navigate(`tasks/${taskId}`)  
 }  
 />  
 </Grid>  
 )  
 })}  
 </ResponsiveGrid>  
 </Box>  
 )  
 })}  
 </Box>  
 )}  
 </Grid>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 <Divider variant="fullWidth" sx={{ my: 4 }} />  
 <Box overflow="hidden">  
 <Typography  
 variant="title2"  
 color="text.secondary"  
 sx={{ width: 'fit-content', cursor: 'pointer' }}  
 onClick={() => setCaseDetailsExpanded((prev) => !prev)}  
 >  
 {t('caseDetails', { ns: 'commonTask' })}{' '}  
 {caseDetailsExpanded ? (  
 <ExpandMoreOutlined />  
 ) : (  
 <ExpandLessOutlined />  
 )}  
 </Typography>  
 {caseDetailsExpanded && (  
 <ResponsiveForm sx={{ mt: 3 }}>  
 {(isDesktop) => (  
 <>  
 {  
}  
 <ResponsiveGrid container isDesktop={isDesktop} gap={3} pb={4}>  
 <Grid item xs>  
 <Controller  
 name="applicationNumber"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicationNumber', {  
 ns: 'commonTask',  
 })}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="applicationType"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title={t('applicationType', {  
 ns: 'commonTask',  
 })}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} disabled />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <CaseForm control={control} type="edit" />  
 <AttachmentTable  
 data={data?.attachments?.filter((item) =>  
 ['CASE'].includes(item.type)  
 )}  
 disabled  
 ref={attachRef}  
 type="CASE"  
 applicationId={applicationId}  
 submissionCaseId={caseId}  
 onSuccess={(caseId) => {  
 queryClient.invalidateQueries([  
 'fetchCaseDetail',  
 { caseId },  
 ])  
 }}  
 />  
 <Button onClick={() => handleSubmit(onSubmit)()}>Save</Button>  
 </>  
 )}  
 </ResponsiveForm>  
 )}  
 </Box>  
 <Divider variant="fullWidth" sx={{ my: 4 }} />  
 <Stack maxHeight="50vh" overflow="auto">  
 <EMinutes caseDetails={data} />  
 </Stack>  
 <Divider variant="fullWidth" sx={{ my: 4 }} />  
 <Button onClick={() => navigate(`/applications/${applicationId}`)}>  
 Back  
 </Button>  
 </Box>  
 )  
}  
export default CaseDetailPage

## File: bd-scs-backend-web/src/routes/case/helper.jsx

export const getApplicationType = (type) => {  
 if (!type) return undefined  
 if (type === "NEWSCH" || type === "EXTSCH") return "SCH"  
 if (type === "NLH") return "NLH"  
 if (type === "NEWCCC" || type === "NEWJOKC" || type === "NEWMUT") return "CCC"  
}

## File: bd-scs-backend-web/src/routes/efolio-search/index.jsx

import { useTranslation } from 'react-i18next'  
import { Controller, useForm } from 'react-hook-form'  
import React from 'react'  
import { useNavigate, useLocation } from 'react-router-dom'  
import Divider from '@mui/material/Divider'  
import Grid from '@mui/material/Grid'  
import Box from '@mui/material/Box'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import TextField from 'src/components/basic/TextField'  
import colors from 'src/constants/colors'  
import { getApplicationHistory } from 'src/apis/application'  
import DataTable from 'components/basic/DataGrid'  
import constants from 'src/constants'  
import Button from 'src/components/basic/Button'  
import VisibilityIcon from '@mui/icons-material/Visibility'  
import PrintIcon from '@mui/icons-material/Print'  
import DownloadIcon from '@mui/icons-material/Download'  
import Stack from '@mui/material/Stack'  
function EfolioSearch() {  
 const navigate = useNavigate()  
 const location = useLocation()  
 const [applicationHistories, setApplicationHistories] = React.useState([])  
 const { t, i18n } = useTranslation(['advanceSearch'])  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm()  
 const onSearch = async (data) => {  
 try {  
 const \_applicationHistories = await getApplicationHistory({  
 frefPref: data.fileRefPref,  
 frefSeq: data.fileRefSeq,  
 frefYr: data.fileRefYr,  
 frefSuf: data.fileRefSuf,  
 })  
 setApplicationHistories(\_applicationHistories)  
 } catch (error) {  
 console.error('Failed to fetch application histories:', error)  
 }  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy">  
 {t('E-Folio Search')}  
 </Typography>  
 <Grid container direction="column" flexWrap="nowrap">  
 {}  
 <Grid container direction="row" gap={2} alignItems="center" mt={2}>  
 <Typography sx={{ minWidth: 150 }}>File Reference ID:</Typography>  
 <Grid item xs>  
 <Controller  
 name="fileRefPref"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} placeholder="Prefix" fullWidth />  
 )}  
 />  
 </Grid>  
 <Typography>/</Typography>  
 <Grid item xs>  
 <Controller  
 name="fileRefSeq"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} placeholder="Sequence" fullWidth />  
 )}  
 />  
 </Grid>  
 <Typography>/</Typography>  
 <Grid item xs>  
 <Controller  
 name="fileRefYr"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} placeholder="Year" fullWidth />  
 )}  
 />  
 </Grid>  
 <Typography>/</Typography>  
 <Grid item xs>  
 <Controller  
 name="fileRefSuf"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <TextField {...field} placeholder="Suffix" fullWidth />  
 )}  
 />  
 </Grid>  
 <Grid item>  
 <GradientButton  
 onClick={handleSubmit(onSearch)}  
 bgColor="navy"  
 sx={{ padding: `8px 22px`, minWidth: 80 }}  
 >  
 Search  
 </GradientButton>  
 </Grid>  
 </Grid>  
 </Grid>  
 <Box  
 style={{ marginTop: 16 }}  
 height={500}  
 minHeight={500}  
 maxHeight={500}  
 >  
 <DataTable  
 staticData={applicationHistories}  
 disableRowSelectionOnClick  
 columns={[  
 {  
 field: 'efolio',  
 headerName: 'E-Folio',  
 width: 300,  
 },  
 {  
 field: 'filePartNo',  
 headerName: 'File Part No.',  
 width: 300,  
 },  
 {  
 field: 'sysFileRefDisplay',  
 headerName: 'File Reference',  
 width: 300,  
 },  
 {  
 field: 'receivedDate',  
 headerName: 'Received Date',  
 width: 300,  
 },  
 {  
 field: 'type',  
 headerName: 'TYPE',  
 width: 300,  
 },  
 {  
 field: '\_id',  
 headerName: 'Actions',  
 width: 400,  
 renderCell: (params) => {  
 if (!params.row.file) return null  
 const fileUrl = `${constants.apiEndPoint}attachments/${params.row.\_id}`  
 const fileName = params.row.originalName || 'document'  
 const fileType = fileName.split('.').pop().toLowerCase()  
 const handlePreview = () => {  
 const previewableTypes = ['pdf', 'jpg', 'jpeg', 'png', 'gif']  
 if (previewableTypes.includes(fileType)) {  
 window.open(fileUrl, '\_blank')  
 } else {  
 console.log('Preview not available for this file type')  
 }  
 }  
 const handlePrint = () => {  
 if (['pdf', 'jpg', 'jpeg', 'png', 'gif'].includes(fileType)) {  
 const printWindow = window.open(fileUrl, '\_blank')  
 printWindow?.addEventListener('load', () => {  
 setTimeout(() => printWindow.print(), 1000)  
 })  
 } else {  
 console.log('Printing not supported for this file type')  
 }  
 }  
 const handleDownload = () => {  
 const link = document.createElement('a')  
 link.href = fileUrl  
 link.download = fileName  
 document.body.appendChild(link)  
 link.click()  
 document.body.removeChild(link)  
 }  
 return (  
 <Box  
 sx={{  
 display: 'flex',  
 justifyContent: 'center',  
 alignItems: 'center',  
 width: '100%',  
 height: '100%',  
 gap: 2,  
 }}  
 >  
 {}  
 <Button  
 variant="contained"  
 size="small"  
 startIcon={<VisibilityIcon fontSize="small" />}  
 onClick={(e) => {  
 e.stopPropagation()  
 handlePreview()  
 }}  
 sx={{ minWidth: 90, mx: 1 }}  
 >  
 Preview  
 </Button>  
 {}  
 <Button  
 variant="contained"  
 size="small"  
 startIcon={<PrintIcon fontSize="small" />}  
 onClick={(e) => {  
 e.stopPropagation()  
 handlePrint()  
 }}  
 sx={{ minWidth: 90, mx: 1 }}  
 >  
 Print  
 </Button>  
 {}  
 <Button  
 variant="contained"  
 size="small"  
 color="primary"  
 startIcon={<DownloadIcon fontSize="small" />}  
 onClick={(e) => {  
 e.stopPropagation()  
 handleDownload()  
 }}  
 sx={{ minWidth: 90, mx: 1 }}  
 >  
 Download  
 </Button>  
 </Box>  
 )  
 },  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 }}  
 onRowClick={(row) => {  
 }}  
 />  
 <Button sx={{ marginTop: 2, marginBottom: 5 }}>Upload Document</Button>  
 </Box>  
 </Box>  
 )  
}  
export default EfolioSearch

## File: bd-scs-backend-web/src/routes/forms/a2.jsx

import { get, useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation } from '@tanstack/react-query'  
import React from 'react'  
import { useNavigate } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/a2'  
import dayjs from 'dayjs'  
import { createApplication, addAttachments } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
export const schema = yup.object().shape({  
 ApplicantName: yup.string().required('Required'),  
 ApplicantEmail: yup.string().email('Invalid email').required('Required'),  
 ApplicantMobile: yup.string().required('Required'),  
 ApplicantAddress: yup.string().required('Required'),  
 AddressOfPremiseEN: yup.string().required('Required'),  
 AddressOfPremiseCN: yup.string().required('Required'),  
 NameOfSchoolEN: yup.string().required('Required'),  
 NameOfSchoolCN: yup.string().required('Required'),  
 DescriptionOfSchool: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 OtherSchoolDesc: yup.string().when('DescriptionOfSchool', {  
 is: (val) => val.id === 'other\_sch',  
 then: (schema) => schema.required('Required'),  
 otherwise: (schema) => schema.nullable().optional(),  
 }),  
 AgeOfStudent: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 EstimatedNoOfStudent: yup.string().required('Required'),  
 docs: yup.array().of(  
 yup.object({  
 docType: yup.object({  
 id: yup.string().required('Required'),  
 }),  
 doc: yup.mixed().required('Required'),  
 })  
 )  
})  
function A2FormPage() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors, isSubmitSuccessful },  
 watch,  
 setValue,  
 getValues,  
 } = useForm({  
 resolver: yupResolver(schema),  
 })  
 const onUpload = async (applicationId, submissionId) => {  
 const docs = getValues('docs');  
 if (!Array.isArray(docs) || docs.length <= 0) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate('/forms')  
 return  
 }  
 docs.forEach(async (doc, index) => {  
 const formData = new FormData();  
 formData.append('file', doc.doc);  
 formData.append('type', doc?.docType?.id);  
 const response = await addAttachments(  
 applicationId,  
 submissionId,  
 formData  
 );  
 if (index === (docs.length - 1)) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate('/forms')  
 }  
 })  
 }  
 const createMutation = useMutation({  
 mutationFn: (payload) => createApplication(payload),  
 onSuccess: (data) => {  
 console.log('data', data)  
 onUpload(  
 data?.application,  
 data?.\_id  
 )  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = ({ docs, ...formData }) => {  
 let payload = {  
 ApplicationType: 'NEWSCH',  
 ...formData,  
 }  
 payload.AgeOfStudent = payload?.AgeOfStudent?.id  
 ; (payload.ApplicationDate = payload.ApplicationDate  
 ? dayjs(payload.ApplicationDate).toISOString()  
 : undefined),  
 (payload.RegTypeOfAp1 = payload?.RegTypeOfAp1?.id)  
 payload.RegTypeOfAp2 = payload?.RegTypeOfAp2?.id  
 payload.RegTypeOfRse = payload?.RegTypeOfRse?.id  
 payload.DescriptionOfSchool = payload?.DescriptionOfSchool?.id  
 payload.SubmissionType = 'A2'  
 payload.EApplication = false  
 createMutation.mutate(payload)  
 }  
 return (  
 <Stack  
 direction="column"  
 p={5}  
 overflow="auto"  
 sx={{ backgroundColor: colors.bgColor }}  
 >  
 <Typography  
 variant="title2"  
 color="text.secondary"  
 sx={{ textAlign: 'center', mb: 2 }}  
 >  
 {t('a2FormHeader')}  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <GradientButton  
 sx={{ alignSelf: 'end', mt: 2, height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.submit')}  
 </GradientButton>  
 </Stack>  
 )  
}  
export default A2FormPage

## File: bd-scs-backend-web/src/routes/forms/appointAp.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useQuery } from '@tanstack/react-query'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/appointAp'  
import { addAttachments, createSubmission } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { getApplication } from 'src/apis/application'  
import { Button } from '@mui/material'  
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined'  
import {  
 StudentAge,  
 SchoolCategory,  
 RegTypes1,  
 RegTypes2,  
 RegTypes3,  
} from 'src/constants/options'  
const schema = yup.object().shape({})  
function AppointApForm() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId } = useParams()  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 ...data,  
 ApplicationDate: data?.ApplicationDate  
 ? dayjs(data.ApplicationDate)  
 : undefined,  
 RegTypeOfAp1: RegTypes1.find(  
 (item) => item.id === data?.RegTypeOfAp1  
 ),  
 RegTypeOfAp2: RegTypes2.find(  
 (item) => item.id === data?.RegTypeOfAp2  
 ),  
 RegTypeOfRse: RegTypes3.find(  
 (item) => item.id === data?.RegTypeOfRse  
 ),  
 },  
 })  
 const createMutation = useMutation({  
 mutationFn: (payload) => createSubmission(applicationId, payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 let payload = { SubmissionType: 'appoint-ap', ...formData }  
 payload.RegTypeOfAp1 = payload?.RegTypeOfAp1?.id  
 payload.RegTypeOfAp2 = payload?.RegTypeOfAp2?.id  
 payload.RegTypeOfRse = payload?.RegTypeOfRse?.id  
 delete payload.\_id  
 delete payload.ApplicationNo  
 delete payload.createdAt  
 delete payload.\_\_v  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Button  
 sx={{  
 color: 'rgb(59 130 246)',  
 cursor: 'pointer',  
 width: 'fit-content',  
 textTransform: 'none',  
 my: 3,  
 }}  
 onClick={() => navigate(`/applications/${applicationId}`)}  
 startIcon={<ArrowBackOutlinedIcon />}  
 >  
 {`Application No. ${data?.ApplicationNo}`}  
 </Button>  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 Appointment of Authorized Person (AP) and Structural Engineer  
 (RSE)  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.submit')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default AppointApForm

## File: bd-scs-backend-web/src/routes/forms/b2.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation, useQuery } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Button from '@mui/material/Button'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/b2'  
import {  
 addAttachments,  
 createSubmission,  
 getApplication,  
} from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined'  
const schema = yup.object().shape({  
 AmendmentOfLayoutPlan: yup.string().required('Required'),  
 BDLetterRefDate: yup.date().required('Required'),  
 docs: yup.array().of(  
 yup.object({  
 docType: yup.object({  
 id: yup.string().required('Required'),  
 }),  
 doc: yup.mixed().required('Required'),  
 })  
 ),  
})  
function B2Form() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId } = useParams()  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 getValues,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 AmendmentOfLayoutPlan: 'Buildings Department',  
 ApplicationDate: data?.ApplicationDate  
 ? dayjs(data.ApplicationDate)  
 : undefined,  
 },  
 })  
 const onUpload = async (applicationId, submissionId) => {  
 const docs = getValues('docs')  
 if (!Array.isArray(docs) || docs.length <= 0) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 return  
 }  
 docs.forEach(async (doc, index) => {  
 const formData = new FormData()  
 formData.append('file', doc.doc)  
 formData.append('type', doc?.docType?.id)  
 const response = await addAttachments(  
 applicationId,  
 submissionId,  
 formData  
 )  
 if (index === docs.length - 1) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 }  
 })  
 }  
 const createMutation = useMutation({  
 mutationFn: (payload) => createSubmission(applicationId, payload),  
 onSuccess: (data) => {  
 onUpload(data?.application, data?.\_id)  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log('Submitting')  
 let payload = { SubmissionType: 'B2', ...formData }  
 ;(payload.ApplicationDate = payload?.ApplicationDate  
 ? dayjs(payload.ApplicationDate).toISOString()  
 : undefined),  
 (payload.BDLetterRefDate = payload?.BDLetterRefDate  
 ? dayjs(payload.BDLetterRefDate).toISOString()  
 : undefined),  
 console.log('payload', payload)  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Button  
 sx={{  
 color: 'rgb(59 130 246)',  
 cursor: 'pointer',  
 width: 'fit-content',  
 textTransform: 'none',  
 my: 3,  
 }}  
 onClick={() => navigate(`/applications/${applicationId}`)}  
 startIcon={<ArrowBackOutlinedIcon />}  
 >  
 {`Application No. ${data?.ApplicationNo}`}  
 </Button>  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 {t('b2FormHeader')}  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.submit')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default B2Form

## File: bd-scs-backend-web/src/routes/forms/c2.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation, useQuery } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Button from '@mui/material/Button'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/c2'  
import {  
 addAttachments,  
 createSubmission,  
 getApplication,  
} from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined'  
const schema = yup.object().shape({  
 AmendmentOfLayoutPlan: yup.string().required('Required'),  
 BDLetterRefDate: yup.date().required('Required'),  
 docs: yup.array().of(  
 yup.object({  
 docType: yup.object({  
 id: yup.string().required('Required'),  
 }),  
 doc: yup.mixed().required('Required'),  
 })  
 ),  
})  
function C2Form() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId } = useParams()  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 getValues,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 AmendmentOfLayoutPlan: 'Buildings Department',  
 ApplicationDate: data?.ApplicationDate  
 ? dayjs(data.ApplicationDate)  
 : undefined,  
 },  
 })  
 const onUpload = async (applicationId, submissionId) => {  
 const docs = getValues('docs')  
 if (!Array.isArray(docs) || docs.length <= 0) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 return  
 }  
 docs.forEach(async (doc, index) => {  
 const formData = new FormData()  
 formData.append('file', doc.doc)  
 formData.append('type', doc?.docType?.id)  
 const response = await addAttachments(  
 applicationId,  
 submissionId,  
 formData  
 )  
 if (index === docs.length - 1) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 }  
 })  
 }  
 const createMutation = useMutation({  
 mutationFn: (payload) => createSubmission(applicationId, payload),  
 onSuccess: (data) => {  
 onUpload(data?.application, data?.\_id)  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log('Submitting')  
 let payload = { SubmissionType: 'C2', ...formData }  
 ;(payload.ApplicationDate = payload?.ApplicationDate  
 ? dayjs(payload.ApplicationDate).toISOString()  
 : undefined),  
 (payload.BDLetterRefDate = payload?.BDLetterRefDate  
 ? dayjs(payload.BDLetterRefDate).toISOString()  
 : undefined),  
 console.log('payload', payload)  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Button  
 sx={{  
 color: 'rgb(59 130 246)',  
 cursor: 'pointer',  
 width: 'fit-content',  
 textTransform: 'none',  
 my: 3,  
 }}  
 onClick={() => navigate(`/applications/${applicationId}`)}  
 startIcon={<ArrowBackOutlinedIcon />}  
 >  
 {`Application No. ${data?.ApplicationNo}`}  
 </Button>  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 {t('c2FormHeader')}  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.submit')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default C2Form

## File: bd-scs-backend-web/src/routes/forms/cccaic2.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation } from '@tanstack/react-query'  
import { useNavigate } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/cccaic2'  
import { createApplication } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
const schema = yup.object().shape({  
 ApplicantTitle: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 ApplicantNameEN: yup.string().required('Required'),  
 ApplicantName: yup.string().required('Required'),  
 ApplicantEmail: yup.string().email('Invalid email').required('Required'),  
 ApplicantMobile: yup.string().required('Required'),  
 ApplicantHKIC: yup.string().required('Required'),  
 ApplicationDate: yup.date().required('Required'),  
 ApplicantAddressEN: yup.string().required('Required'),  
 ApplicantAddress: yup.string().required('Required'),  
 ApplicantTel: yup.string().required('Required'),  
 AddressOfPremiseEN: yup.string().required('Required'),  
 AddressOfPremiseCN: yup.string().required('Required'),  
 NameOfSchoolEN: yup.string().required('Required'),  
 NameOfSchoolCN: yup.string().required('Required'),  
 ContactPTitle: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 ContactPName: yup.string().required('Required'),  
 ContactPFax: yup.string().required('Required'),  
 ContactPEmail: yup.string().email('Invalid email').required('Required'),  
 ContactPMobile: yup.string().required('Required'),  
 ContactPAddress: yup.string().required('Required'),  
})  
function Cccaic2Form() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 })  
 const createMutation = useMutation({  
 mutationFn: (payload) => createApplication(payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate('/forms')  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 let payload = { ApplicationType: 'NEWCCC', ...formData }  
 ;(payload.ApplicationDate = payload?.ApplicationDate  
 ? dayjs(payload.ApplicationDate).toISOString()  
 : undefined),  
 (payload.ApplicantTitle = payload?.ApplicantTitle?.id)  
 payload.ContactPTitle = payload?.ContactPTitle?.id  
 payload.RegTypeOfAp1 = payload?.RegTypeOfAp1?.id  
 payload.RegTypeOfAp2 = payload?.RegTypeOfAp2?.id  
 payload.RegTypeOfRse = payload?.RegTypeOfRse?.id  
 payload.SubmissionType = 'CCCAIC2'  
 payload.EApplication = false  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 {t('cccaic2formHeader')}  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.save')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default Cccaic2Form

## File: bd-scs-backend-web/src/routes/forms/cccaimh3.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation } from '@tanstack/react-query'  
import { useNavigate } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/cccaimh3'  
import { createApplication } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
const schema = yup.object().shape({  
 ApplicantTitle: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 ApplicantNameEN: yup.string().required('Required'),  
 ApplicantName: yup.string().required('Required'),  
 ApplicantEmail: yup.string().email('Invalid email').required('Required'),  
 ApplicantMobile: yup.string().required('Required'),  
 ApplicantHKIC: yup.string().required('Required'),  
 ApplicationDate: yup.date().required('Required'),  
 ApplicantRepOrgEN: yup.string().required('Required'),  
 ApplicantRepOrg: yup.string().required('Required'),  
 AddressOfPremiseEN: yup.string().required('Required'),  
 AddressOfPremiseCN: yup.string().required('Required'),  
 NameOfSchoolEN: yup.string().required('Required'),  
 NameOfSchoolCN: yup.string().required('Required'),  
 ContactPTitle: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 ContactPName: yup.string().required('Required'),  
 ContactPFax: yup.string().required('Required'),  
 ContactPEmail: yup.string().email('Invalid email').required('Required'),  
 ContactPTel: yup.string().required('Required'),  
 ContactPAddress: yup.string().required('Required'),  
})  
function Cccaimh3Form() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['cccaimh3', 'form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 })  
 const createMutation = useMutation({  
 mutationFn: (payload) => createApplication(payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate('/forms')  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 let payload = {  
 ApplicationType: 'NEWMUT',  
 ...formData,  
 }  
 ;(payload.ApplicationDate = payload?.ApplicationDate  
 ? dayjs(payload.ApplicationDate).toISOString()  
 : undefined),  
 (payload.ApplicantTitle = payload?.ApplicantTitle?.id)  
 payload.ContactPTitle = payload?.ContactPTitle?.id  
 payload.RegTypeOfAp1 = payload?.RegTypeOfAp1?.id  
 payload.RegTypeOfAp2 = payload?.RegTypeOfAp2?.id  
 payload.RegTypeOfRse = payload?.RegTypeOfRse?.id  
 payload.SubmissionType = 'CCCAIMH/3'  
 payload.EApplication = false  
 console.log('payload', payload)  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 {t('formHeader')}  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.save', { ns: 'form' })}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default Cccaimh3Form

## File: bd-scs-backend-web/src/routes/forms/e2.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation } from '@tanstack/react-query'  
import { useNavigate } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/e2'  
import { createApplication } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
const schema = yup.object().shape({  
 ApplicantName: yup.string().required('Required'),  
 ApplicantEmail: yup.string().email('Invalid email').required('Required'),  
 ApplicantMobile: yup.string().required('Required'),  
 ApplicationDate: yup.date().required('Required'),  
 ApplicantAddress: yup.string().required('Required'),  
 AddressOfPremiseEN: yup.string().required('Required'),  
 AddressOfPremiseCN: yup.string().required('Required'),  
 NameOfSchoolEN: yup.string().required('Required'),  
 NameOfSchoolCN: yup.string().required('Required'),  
 DescriptionOfSchool: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 OtherSchoolDesc: yup.string().when('DescriptionOfSchool', {  
 is: (val) => val.id === 'other\_sch',  
 then: (schema) => schema.required('Required'),  
 otherwise: (schema) => schema.nullable().optional(),  
 }),  
 AgeOfStudent: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 EstimatedNoOfStudent: yup.string().required('Required'),  
})  
function E2FormPage() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 })  
 const createMutation = useMutation({  
 mutationFn: (payload) => createApplication(payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate('/forms')  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 let payload = {  
 ApplicationType: 'EXTSCH',  
 ...formData,  
 }  
 payload.AgeOfStudent = payload?.AgeOfStudent?.id  
 ;(payload.ApplicationDate = payload.ApplicationDate  
 ? dayjs(payload.ApplicationDate).toISOString()  
 : undefined),  
 (payload.RegTypeOfAp1 = payload?.RegTypeOfAp1?.id)  
 payload.RegTypeOfAp2 = payload?.RegTypeOfAp2?.id  
 payload.RegTypeOfRse = payload?.RegTypeOfRse?.id  
 payload.DescriptionOfSchool = payload?.DescriptionOfSchool?.id  
 payload.SubmissionType = 'E2'  
 payload.EApplication = false  
 createMutation.mutate(payload)  
 }  
 return (  
 <Stack  
 direction="column"  
 p={5}  
 overflow="auto"  
 sx={{ backgroundColor: colors.bgColor }}  
 >  
 <Typography  
 variant="title2"  
 color="text.secondary"  
 sx={{ textAlign: 'center', mb: 2 }}  
 >  
 {t('e2FormHeader')}  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.save')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Stack>  
 )  
}  
export default E2FormPage

## File: bd-scs-backend-web/src/routes/forms/f2.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation, useQuery } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Button from '@mui/material/Button'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/f2'  
import {  
 addAttachments,  
 createSubmission,  
 getApplication,  
} from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined'  
const schema = yup.object().shape({  
 AmendmentOfLayoutPlan: yup.string().required('Required'),  
 BDLetterRefDate: yup.date().required('Required'),  
 docs: yup.array().of(  
 yup.object({  
 docType: yup.object({  
 id: yup.string().required('Required'),  
 }),  
 doc: yup.mixed().required('Required'),  
 })  
 ),  
})  
function F2Form() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId } = useParams()  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 getValues,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 AmendmentOfLayoutPlan: 'Buildings Department',  
 ApplicationDate: data?.ApplicationDate  
 ? dayjs(data.ApplicationDate)  
 : undefined,  
 },  
 })  
 const onUpload = async (applicationId, submissionId) => {  
 const docs = getValues('docs')  
 if (!Array.isArray(docs) || docs.length <= 0) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 return  
 }  
 docs.forEach(async (doc, index) => {  
 const formData = new FormData()  
 formData.append('file', doc.doc)  
 formData.append('type', doc?.docType?.id)  
 const response = await addAttachments(  
 applicationId,  
 submissionId,  
 formData  
 )  
 if (index === docs.length - 1) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 }  
 })  
 }  
 const createMutation = useMutation({  
 mutationFn: (payload) => createSubmission(applicationId, payload),  
 onSuccess: (data) => {  
 onUpload(data?.application, data?.\_id)  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log('Submitting')  
 let payload = { SubmissionType: 'F2', ...formData }  
 ;(payload.ApplicationDate = payload?.ApplicationDate  
 ? dayjs(payload.ApplicationDate).toISOString()  
 : undefined),  
 (payload.BDLetterRefDate = payload?.BDLetterRefDate  
 ? dayjs(payload.BDLetterRefDate).toISOString()  
 : undefined),  
 console.log('payload', payload)  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Button  
 sx={{  
 color: 'rgb(59 130 246)',  
 cursor: 'pointer',  
 width: 'fit-content',  
 textTransform: 'none',  
 my: 3,  
 }}  
 onClick={() => navigate(`/applications/${applicationId}`)}  
 startIcon={<ArrowBackOutlinedIcon />}  
 >  
 {`Application No. ${data?.ApplicationNo}`}  
 </Button>  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 {t('f2FormHeader')}  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.submit')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default F2Form

## File: bd-scs-backend-web/src/routes/forms/g2.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation, useQuery } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Button from '@mui/material/Button'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/g2'  
import {  
 addAttachments,  
 createSubmission,  
 getApplication,  
} from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined'  
const schema = yup.object().shape({  
 AmendmentOfLayoutPlan: yup.string().required('Required'),  
 BDLetterRefDate: yup.date().required('Required'),  
 docs: yup.array().of(  
 yup.object({  
 docType: yup.object({  
 id: yup.string().required('Required'),  
 }),  
 doc: yup.mixed().required('Required'),  
 })  
 ),  
})  
function G2Form() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId } = useParams()  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 getValues,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 AmendmentOfLayoutPlan: 'Buildings Department',  
 ApplicationDate: data?.ApplicationDate  
 ? dayjs(data.ApplicationDate)  
 : undefined,  
 },  
 })  
 const onUpload = async (applicationId, submissionId) => {  
 const docs = getValues('docs')  
 if (!Array.isArray(docs) || docs.length <= 0) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 return  
 }  
 docs.forEach(async (doc, index) => {  
 const formData = new FormData()  
 formData.append('file', doc.doc)  
 formData.append('type', doc?.docType?.id)  
 const response = await addAttachments(  
 applicationId,  
 submissionId,  
 formData  
 )  
 if (index === docs.length - 1) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 }  
 })  
 }  
 const createMutation = useMutation({  
 mutationFn: (payload) => createSubmission(applicationId, payload),  
 onSuccess: (data) => {  
 onUpload(data?.application, data?.\_id)  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log('Submitting')  
 let payload = { SubmissionType: 'G2', ...formData }  
 ;(payload.ApplicationDate = payload?.ApplicationDate  
 ? dayjs(payload.ApplicationDate).toISOString()  
 : undefined),  
 (payload.BDLetterRefDate = payload?.BDLetterRefDate  
 ? dayjs(payload.BDLetterRefDate).toISOString()  
 : undefined),  
 console.log('payload', payload)  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Button  
 sx={{  
 color: 'rgb(59 130 246)',  
 cursor: 'pointer',  
 width: 'fit-content',  
 textTransform: 'none',  
 my: 3,  
 }}  
 onClick={() => navigate(`/applications/${applicationId}`)}  
 startIcon={<ArrowBackOutlinedIcon />}  
 >  
 {`Application No. ${data?.ApplicationNo}`}  
 </Button>  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 {t('g2FormHeader')}  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.submit')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default G2Form

## File: bd-scs-backend-web/src/routes/forms/index.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { useNavigate } from 'react-router-dom'  
import Box from '@mui/material/Box'  
import Typography from 'components/basic/Typography'  
import Autocomplete from 'components/basic/Autocomplete'  
import FormControl from 'src/components/basic/FormControl'  
import GradientButton from 'src/components/basic/Button/GradientButton'  
import colors from 'src/constants/colors'  
import { Applications } from 'src/constants/options'  
function Forms() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['common'])  
 const [value, setValue] = React.useState(null)  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy">  
 Applications  
 </Typography>  
 <FormControl  
 title="Select Application"  
 labelPlacement="top"  
 fullWidth  
 sx={{ mt: 2 }}  
 >  
 <Autocomplete  
 options={Object.keys(Applications).map((key) => ({  
 id: Applications[key],  
 label: key,  
 }))}  
 getOptionLabel={(option) => option.label}  
 value={value}  
 onChange={(e, value) => setValue(value)}  
 disableClearable  
 placeholder={t('placeholder.pleaseSelect')}  
 sx={{ width: '50%' }}  
 />  
 </FormControl>  
 <GradientButton  
 sx={{ width: 'fit-content', mt: 2 }}  
 onClick={() => navigate(`/forms/${value.id}`)}  
 disabled={!value?.id}  
 >  
 Next  
 </GradientButton>  
 </Box>  
 )  
}  
export default Forms

## File: bd-scs-backend-web/src/routes/forms/jokc02.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation } from '@tanstack/react-query'  
import { useNavigate } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import Divider from '@mui/material/Divider'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/jokc02'  
import { createApplication } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
const schema = yup.object().shape({  
 ApplicantTitle: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 ApplicantNameEN: yup.string().required('Required'),  
 ApplicantName: yup.string().required('Required'),  
 ApplicantEmail: yup.string().email('Invalid email').required('Required'),  
 ApplicantMobile: yup.string().required('Required'),  
 ApplicantHKIC: yup.string().required('Required'),  
 ApplicationDate: yup.date().required('Required'),  
 AddressOfPremiseEN: yup.string().required('Required'),  
 AddressOfPremiseCN: yup.string().required('Required'),  
 NameOfSchoolEN: yup.string().required('Required'),  
 NameOfSchoolCN: yup.string().required('Required'),  
 ApplicantAddressEN: yup.string().required('Required'),  
 ApplicantAddress: yup.string().required('Required'),  
 ApplicantTel: yup.string().required('Required'),  
 ContactPTitle: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 ContactPName: yup.string().required('Required'),  
 ContactPFax: yup.string().required('Required'),  
 ContactPEmail: yup.string().email('Invalid email').required('Required'),  
 ContactPTel: yup.string().required('Required'),  
 ContactPAddress: yup.string().required('Required'),  
})  
function Jokc02Form() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 })  
 const createMutation = useMutation({  
 mutationFn: (payload) => createApplication(payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate('/forms')  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 let payload = {  
 ApplicationType: 'NEWJOKC',  
 ...formData,  
 }  
 ;(payload.ApplicationDate = payload?.ApplicationDate  
 ? dayjs(payload.ApplicationDate).toISOString()  
 : undefined),  
 (payload.ApplicantTitle = payload?.ApplicantTitle?.id)  
 payload.ContactPTitle = payload?.ContactPTitle?.id  
 payload.RegTypeOfAp1 = payload?.RegTypeOfAp1?.id  
 payload.RegTypeOfAp2 = payload?.RegTypeOfAp2?.id  
 payload.RegTypeOfRse = payload?.RegTypeOfRse?.id  
 payload.SubmissionType = 'JOKC02'  
 payload.EApplication = false  
 console.log('payload', payload)  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 {t('jokc02formHeader')}  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.save')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default Jokc02Form

## File: bd-scs-backend-web/src/routes/forms/submitDoc.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useQuery } from '@tanstack/react-query'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/submitDoc'  
import { addAttachments, createSubmission } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { getApplication } from 'src/apis/application'  
import { Button } from '@mui/material'  
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined'  
const schema = yup.object().shape({  
 docs: yup.array().of(  
 yup.object({  
 docType: yup.object({  
 id: yup.string().required('Required'),  
 }),  
 doc: yup.mixed().required('Required'),  
 })  
 ),  
})  
function SubmitDocForm() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId } = useParams()  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 getValues,  
 } = useForm({  
 resolver: yupResolver(schema),  
 })  
 const onUpload = async (applicationId, submissionId) => {  
 const docs = getValues('docs')  
 console.log('Upload docs', docs)  
 if (!Array.isArray(docs) || docs.length <= 0) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 return  
 }  
 docs.forEach(async (doc, index) => {  
 const formData = new FormData()  
 formData.append('file', doc.doc)  
 formData.append('type', doc?.docType?.id)  
 const response = await addAttachments(  
 applicationId,  
 submissionId,  
 formData  
 )  
 if (index === docs.length - 1) {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 }  
 })  
 }  
 const createMutation = useMutation({  
 mutationFn: (payload) => createSubmission(applicationId, payload),  
 onSuccess: (data) => {  
 onUpload(data?.application, data?.\_id)  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = () => {  
 let payload = { SubmissionType: 'submit-doc' }  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Button  
 sx={{  
 color: 'rgb(59 130 246)',  
 cursor: 'pointer',  
 width: 'fit-content',  
 textTransform: 'none',  
 my: 3,  
 }}  
 onClick={() => navigate(`/applications/${applicationId}`)}  
 startIcon={<ArrowBackOutlinedIcon />}  
 >  
 {`Application No. ${data?.ApplicationNo}`}  
 </Button>  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 Submit Supporting Document  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.submit')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default SubmitDocForm

## File: bd-scs-backend-web/src/routes/forms/updateInfo.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useQuery } from '@tanstack/react-query'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/updateInfo'  
import { addAttachments, createSubmission } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { getApplication } from 'src/apis/application'  
import { Button } from '@mui/material'  
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined'  
import { StudentAge, SchoolCategory } from 'src/constants/options'  
const schema = yup.object().shape({  
 ApplicantName: yup.string().required('Required'),  
 ApplicantEmail: yup.string().email('Invalid email').required('Required'),  
 ApplicantMobile: yup.string().required('Required'),  
 ApplicantAddress: yup.string().required('Required'),  
 AddressOfPremiseEN: yup.string().required('Required'),  
 AddressOfPremiseCN: yup.string().required('Required'),  
 NameOfSchoolEN: yup.string().required('Required'),  
 NameOfSchoolCN: yup.string().required('Required'),  
 DescriptionOfSchool: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 OtherSchoolDesc: yup.string().when('DescriptionOfSchool', {  
 is: (val) => val.id === 'other\_sch',  
 then: (schema) => schema.required('Required'),  
 otherwise: (schema) => schema.nullable().optional(),  
 }),  
 AgeOfStudent: yup  
 .object({  
 id: yup.string().required('Required'),  
 })  
 .required('Required'),  
 EstimatedNoOfStudent: yup.string().required('Required'),  
})  
function UpdateInfoForm() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId } = useParams()  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 ...data,  
 ApplicationDate: data?.ApplicationDate  
 ? dayjs(data.ApplicationDate)  
 : undefined,  
 AgeOfStudent: StudentAge.find(  
 (item) => item.id === data?.AgeOfStudent  
 ),  
 DescriptionOfSchool: SchoolCategory.find(  
 (item) => item.id === data?.DescriptionOfSchool  
 ),  
 },  
 })  
 const createMutation = useMutation({  
 mutationFn: (payload) => createSubmission(applicationId, payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 let payload = { SubmissionType: 'update-info', ...formData }  
 payload.DescriptionOfSchool = payload?.DescriptionOfSchool?.id  
 payload.AgeOfStudent = payload?.AgeOfStudent?.id  
 delete payload.\_id  
 delete payload.ApplicationNo  
 delete payload.createdAt  
 delete payload.\_\_v  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Button  
 sx={{  
 color: 'rgb(59 130 246)',  
 cursor: 'pointer',  
 width: 'fit-content',  
 textTransform: 'none',  
 my: 3,  
 }}  
 onClick={() => navigate(`/applications/${applicationId}`)}  
 startIcon={<ArrowBackOutlinedIcon />}  
 >  
 {`Application No. ${data?.ApplicationNo}`}  
 </Button>  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 Update Application and School Information  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.submit')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default UpdateInfoForm

## File: bd-scs-backend-web/src/routes/forms/withdrawApp.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useQuery } from '@tanstack/react-query'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import Form from 'components/Form/Type/withdrawApp'  
import { addAttachments, createSubmission } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { getApplication } from 'src/apis/application'  
import { Button } from '@mui/material'  
import ArrowBackOutlinedIcon from '@mui/icons-material/ArrowBackOutlined'  
const schema = yup.object().shape({  
 confirmWithdrawApp: yup.boolean().oneOf([true], 'Required'),  
})  
function WithdrawAppForm() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const { applicationId } = useParams()  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getApplication', { applicationId }],  
 queryFn: async () => await getApplication(applicationId),  
 })  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 })  
 const createMutation = useMutation({  
 mutationFn: (payload) => createSubmission(applicationId, payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate(`/applications/${applicationId}`)  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = () => {  
 let payload = { SubmissionType: 'withdraw-app' }  
 createMutation.mutate(payload)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Button  
 sx={{  
 color: 'rgb(59 130 246)',  
 cursor: 'pointer',  
 width: 'fit-content',  
 textTransform: 'none',  
 my: 3,  
 }}  
 onClick={() => navigate(`/applications/${applicationId}`)}  
 startIcon={<ArrowBackOutlinedIcon />}  
 >  
 {`Application No. ${data?.ApplicationNo}`}  
 </Button>  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 Withdraw Application  
 </Typography>  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Form control={control} setValue={setValue} />  
 <Divider sx={{ borderColor: 'text.secondary' }} />  
 <Stack direction="row" gap={2} mt={2} justifyContent="flex-end">  
 <GradientButton  
 sx={{ height: 40, width: 160 }}  
 onClick={() => handleSubmit(onSubmit)()}  
 >  
 {t('button.submit')}  
 </GradientButton>  
 {}  
 </Stack>  
 </Box>  
 )  
}  
export default WithdrawAppForm

## File: bd-scs-backend-web/src/routes/legacy\_formDetail/index.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Typography from 'src/components/basic/Typography'  
import A2Form from 'components/FormDetail/a2'  
import B2Form from 'components/FormDetail/b2'  
import C2Form from 'components/FormDetail/c2'  
import E2Form from 'components/FormDetail/e2'  
import F2Form from 'components/FormDetail/f2'  
import G2Form from 'components/FormDetail/g2'  
import Cccaic2Form from 'components/FormDetail/cccaic2'  
import Cccaimh3Form from 'components/FormDetail/cccaimh3'  
import Jokc02Form from 'components/FormDetail/jokc02'  
import { updateApplication, getApplication } from 'src/apis/application'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
function FormDetail() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const queryClient = useQueryClient()  
 const { formId } = useParams()  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getApplication', { formId }],  
 queryFn: async () => await getApplication(formId),  
 })  
 const updateMutation = useMutation({  
 mutationFn: (payload) => updateApplication(formId, payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 queryClient.invalidateQueries({  
 queryKey: ['getApplication', { formId }],  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const renderFormContent = () => {  
 if (!data) return null  
 switch (data.ApplicationType) {  
 case 'NEWSCH': {  
 return <A2Form updateMutation={updateMutation} data={data} />  
 }  
 case 'EXTSCH': {  
 return <B2Form updateMutation={updateMutation} data={data} />  
 }  
 case 'C2': {  
 return <C2Form updateMutation={updateMutation} data={data} />  
 }  
 case 'E2': {  
 return <E2Form updateMutation={updateMutation} data={data} />  
 }  
 case 'F2': {  
 return <F2Form updateMutation={updateMutation} data={data} />  
 }  
 case 'G2': {  
 return <G2Form updateMutation={updateMutation} data={data} />  
 }  
 case 'NEWCCC': {  
 return (  
 <Cccaic2Form updateMutation={updateMutation} data={data} />  
 )  
 }  
 case 'NEWMUT': {  
 return (  
 <Cccaimh3Form updateMutation={updateMutation} data={data} />  
 )  
 }  
 case 'NEWJOKC': {  
 return (  
 <Jokc02Form updateMutation={updateMutation} data={data} />  
 )  
 }  
 }  
 }  
 return <>{renderFormContent()}</>  
}  
export default FormDetail

## File: bd-scs-backend-web/src/routes/submissionDetails/index.jsx

import React from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import colors from 'src/constants/colors'  
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import { useNavigate, useParams } from 'react-router-dom'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import dayjs from 'dayjs'  
import Divider from '@mui/material/Divider'  
import Stack from '@mui/material/Stack'  
import GradientButton from 'components/basic/Button/GradientButton'  
import CustomButton from 'components/basic/Button'  
import Typography from 'src/components/basic/Typography'  
import A2Form from 'components/FormDetail/a2'  
import B2Form from 'components/FormDetail/b2'  
import C2Form from 'components/FormDetail/c2'  
import E2Form from 'components/FormDetail/e2'  
import F2Form from 'components/FormDetail/f2'  
import G2Form from 'components/FormDetail/g2'  
import Cccaic2Form from 'components/FormDetail/cccaic2'  
import Cccaimh3Form from 'components/FormDetail/cccaimh3'  
import Jokc02Form from 'components/FormDetail/jokc02'  
import constants from 'src/constants'  
import { TaskTitle } from 'src/constants/tasks'  
import {  
 updateApplication,  
 getApplication,  
 getSubmissions,  
 getSubmission,  
 downloadAttachment,  
} from 'src/apis/application'  
import { getMyTasks } from 'src/apis/user'  
import { createCase } from 'src/apis/case'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { Box, Button, Grid, Link } from '@mui/material'  
import DataTable from 'components/basic/DataGrid'  
import UserDialog from 'components/Dialog/UserDialog'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import UpdateInfoForm from 'components/FormDetail/updateInfo'  
import AppointApForm from 'components/FormDetail/appointAp'  
import TaskBar from 'components/TaskBar'  
function SubmissionDetails() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['form'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const queryClient = useQueryClient()  
 const { submissionId } = useParams()  
 const [userDialog, setUserDialog] = React.useState({ open: false })  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getSubmission', { submissionId }],  
 queryFn: async () => await getSubmission(submissionId),  
 })  
 const {  
 data: taskData,  
 error: taskError,  
 isError: taskIsError,  
 isLoading: taskIsLoading,  
 } = useQuery({  
 queryKey: ['myTask', { submissionId }],  
 queryFn: async () => await getMyTasks({ submission: submissionId }),  
 })  
 const createCaseMutation = useMutation({  
 mutationFn: () => createCase(submissionId),  
 onSuccess: (data) => {  
 queryClient.invalidateQueries({  
 queryKey: ['getSubmission', { submissionId }],  
 })  
 setSnackbar({  
 type: 'success',  
 message: 'Success',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const renderFormContent = () => {  
 if (!data) return null  
 switch (data.SubmissionType) {  
 case 'A2': {  
 return <A2Form data={data} mode="view" />  
 }  
 case 'E2': {  
 return <E2Form data={data} mode="view" />  
 }  
 case 'CCCAIC2': {  
 return <Cccaic2Form data={data} mode="view" />  
 }  
 case 'CCCAIMH/3': {  
 return <Cccaimh3Form data={data} mode="view" />  
 }  
 case 'JOKC02': {  
 return <Jokc02Form data={data} mode="view" />  
 }  
 case 'B2': {  
 return <B2Form data={data} mode="view" />  
 }  
 case 'C2': {  
 return <C2Form data={data} mode="view" />  
 }  
 case 'E2': {  
 return <E2Form data={data} mode="view" />  
 }  
 case 'F2': {  
 return <F2Form data={data} mode="view" />  
 }  
 case 'G2': {  
 return <G2Form data={data} mode="view" />  
 }  
 case 'update-info': {  
 return <UpdateInfoForm data={data} mode="view" />  
 }  
 case 'appoint-ap': {  
 return <AppointApForm data={data} mode="view" />  
 }  
 }  
 }  
 return (  
 <Box px={5} pb={2} overflow="auto">  
 <TaskBar type="SUBMISSION" />  
 {}  
 {  
}  
 <Grid container spacing={5} marginBottom={5}>  
 <Grid item xs={12} sm={6}>  
 <FormControl  
 title={'Submission Type'}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <TextField disabled value={data?.SubmissionType} />  
 </FormControl>  
 </Grid>  
 <Grid item xs={12} sm={6}>  
 <FormControl  
 title={'Assigned GR'}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 {data?.assignedGR ? (  
 <TextField  
 value={data?.assignedGR?.email}  
 disabled  
 />  
 ) : (  
 <CustomButton  
 onClick={() => setUserDialog({ open: true })}  
 >  
 Assign GR  
 </CustomButton>  
 )}  
 </FormControl>  
 </Grid>  
 <Grid item xs={12} sm={6}>  
 <FormControl  
 title={'Assigned BS'}  
 formLabelProps={{ variant: 'navyHeader' }}  
 labelPlacement="top"  
 fullWidth={true}  
 >  
 <TextField value={data?.assignedBS?.name} disabled />  
 </FormControl>  
 </Grid>  
 <Grid item xs={12} sm={6}>  
 <Box  
 display="flex"  
 alignItems="flex-end"  
 justifyContent="flex-end"  
 height="100%"  
 >  
 {data?.submissionCase ? (  
 <CustomButton  
 sx={{ mr: 1 }}  
 onClick={() =>  
 navigate(`/case/${data?.submissionCase}`)  
 }  
 >  
 Go To Case Detail  
 </CustomButton>  
 ) : (  
 <CustomButton  
 onClick={() => createCaseMutation.mutate()}  
 >  
 Create Case  
 </CustomButton>  
 )}  
 </Box>  
 </Grid>  
 </Grid>  
 {!['submit-doc', 'withdraw-app', 'inspect-req'].includes(  
 data?.SubmissionType  
 ) && (  
 <Box id="submission-details" mb={5}>  
 <Typography variant="title2" color="text.secondary">  
 Submission Details  
 </Typography>  
 <Box  
 mt={2}  
 display="flex"  
 flexDirection="column"  
 maxHeight="50vh"  
 >  
 {renderFormContent()}  
 </Box>  
 </Box>  
 )}  
 <Box id="submission-attachments">  
 <Typography variant="title2" color="text.secondary">  
 Attachments  
 </Typography>  
 <DataTable  
 disableRowSelectionOnClick  
 columns={[  
 {  
 field: 'type',  
 headerName: 'Attachment Type',  
 width: 300,  
 },  
 {  
 field: 'file\_ir',  
 headerName: 'Attachment Name',  
 width: 300,  
 renderCell: (params) => {  
 return (  
 <a  
 href={`${constants.apiEndPoint}attachments/${params.row.\_id}`}  
 download  
 >  
 {params.row.file?.originalname}  
 </a>  
 )  
 },  
 },  
 {  
 field: 'createdAt',  
 headerName: 'Uploaded At',  
 width: 300,  
 },  
 ]}  
 staticData={data?.attachments}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 mt: 2,  
 }}  
 />  
 </Box>  
 <UserDialog  
 open={userDialog.open}  
 handleClose={() => setUserDialog({ open: false })}  
 submissionId={submissionId}  
 />  
 </Box>  
 )  
}  
export default SubmissionDetails

## File: bd-scs-backend-web/src/routes/task/index.jsx

import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import dayjs from 'dayjs'  
import React, { useState } from 'react'  
import { useForm } from 'react-hook-form'  
import { useTranslation } from 'react-i18next'  
import { useNavigate, useParams } from 'react-router-dom'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Box from '@mui/material/Box'  
import Divider from '@mui/material/Divider'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Tab from '@mui/material/Tab'  
import Tabs from '@mui/material/Tabs'  
import EMinutes from 'components/Case/Eminutes'  
import CaseProgressItem from 'components/ListItem/CaseProgressItem'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import Typography from 'components/basic/Typography'  
import { getTask } from 'src/apis/task'  
import { TaskTitle } from 'src/constants/tasks'  
import P2 from 'routes/case/DeskStudy/detail'  
import P5 from 'routes/case/FRC/detail'  
import P7 from 'routes/case/MISC/detail'  
import P4 from 'routes/case/MOE/detail'  
import P1 from 'routes/case/P1/detail'  
import P3 from 'routes/case/Structural/bs/detail'  
import P3SE from 'routes/case/Structural/se/detail'  
import P6 from 'routes/case/UBW/detail'  
import P8 from 'routes/case/P8/detail'  
import P9 from 'routes/case/P9/detail'  
import SchForm from 'routes/case/SCH'  
import DVTable from 'routes/case/DvTable'  
import BsRecommendation from 'routes/case/BsRecommendation'  
import Dialog from 'src/components/basic/Dialog'  
import { ExpandLessOutlined, ExpandMoreOutlined } from '@mui/icons-material'  
import { getApplicationType } from 'routes/case/helper'  
import { fetchCaseDetail } from 'src/apis/case'  
import { TASK\_TAB\_MAP } from 'src/constants/tasks'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import TaskBar from 'components/TaskBar'  
import { updateTask } from 'src/apis/user'  
import { Button } from '@mui/material'  
function CustomTabPanel(props) {  
 const { children, value, index, ...other } = props  
 return (  
 <div  
 role="tabpanel"  
 hidden={value !== index}  
 id={`simple-tabpanel-${index}`}  
 aria-labelledby={`simple-tab-${index}`}  
 style={{ overflow: 'auto', height: '100%' }}  
 {...other}  
 >  
 {value === index && <Box height="100%">{children}</Box>}  
 </div>  
 )  
}  
function TaskDetailPage() {  
 const { t, i18n } = useTranslation(['p1Task'])  
 let navigate = useNavigate()  
 const [tabIndex, setTabIndex] = React.useState(0)  
 const [caseProgressExpanded, setCaseProgressExpanded] = React.useState(true)  
 const { applicationId, caseId, taskId } = useParams()  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const queryClient = useQueryClient()  
 const [completingTask, setCompletingTask] = useState(false)  
 const completeTask = () => {  
 setCompletingTask(true)  
 }  
 const cancelCompleteTask = () => {  
 setCompletingTask(false)  
 }  
 const completeTaskMutation = useMutation({  
 mutationFn: () =>  
 updateTask(taskId, {  
 status: 'COMPLETED',  
 }),  
 onSuccess: () => {  
 queryClient.invalidateQueries(['myTask'])  
 setCompletingTask(false)  
 setSnackbar({  
 type: 'success',  
 message: 'Task completed',  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const { data: task } = useQuery({  
 queryKey: ['task', taskId],  
 queryFn: () => getTask(taskId),  
 })  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['fetchCaseDetail', { caseId }],  
 queryFn: async () => await fetchCaseDetail(caseId),  
 })  
 const handleChange = (event, newValue) => {  
 setTabIndex(newValue)  
 }  
 const applicationType = getApplicationType(data?.application?.ApplicationType)  
 const TabList = React.useMemo(() => {  
 const Tabs = [  
 {  
 label: 'Initial Site Inspection',  
 component: (  
 <P1  
 data={{  
 ...data?.site\_inspection?.file,  
 \_id: data?.site\_inspection?.\_id,  
 }}  
 />  
 ),  
 },  
 {  
 label: 'Desk Study',  
 component: (  
 <P2  
 data={data?.deck\_study}  
 buildingData={data?.building\_information}  
 dvData={data?.dv}  
 />  
 ),  
 },  
 {  
 label: 'Structural(SE)',  
 component: <P3SE data={data?.structural\_schnlhkinds} />,  
 },  
 {  
 label: 'Structural(BS)',  
 component: (  
 <P3 data={data?.structural\_schnlhkinds} nature={data?.Nature} />  
 ),  
 },  
 {  
 label: 'Mean Of Escape',  
 component: (  
 <P4  
 data={data?.moe}  
 deskStudy={data?.deck\_study}  
 buildingInformation={data?.building\_information}  
 dv={data?.dv}  
 applicationType={applicationType}  
 />  
 ),  
 },  
 {  
 label: 'Fire Resisting Construction',  
 component: <P5 data={data?.frc} />,  
 },  
 {  
 label: 'Unauthorised Building Work',  
 component: <P6 data={data?.ubw} />,  
 },  
 {  
 label: 'Miscellaneous',  
 component: <P7 data={data?.misc} />,  
 },  
 {  
 label: 'Prepare Letter',  
 component: <P8 data={data?.prepareLetter} caseDetail={data} />,  
 },  
 {  
 label: 'Issue Letter',  
 component: (  
 <P9  
 data={[  
 ...(data?.prepareLetter || []),  
 ...(data?.issueLetter || []),  
 ]}  
 caseDetail={data}  
 />  
 ),  
 },  
 {  
 label: 'Document Checklist',  
 component: <SchForm data={data?.document\_checklist} />,  
 },  
 {  
 label: 'DV',  
 component: <DVTable data={data?.dv} />,  
 },  
 {  
 label: 'BS Recommendation',  
 component: <BsRecommendation data={data?.bs\_recommendation} />,  
 },  
 ]  
 const typeTabs = TASK\_TAB\_MAP?.[task?.taskType.toUpperCase()]  
 console.log('typeTabs', typeTabs)  
 if (!typeTabs) return []  
 return [  
 ...Tabs.filter((\_, index) => typeTabs.includes(index)),  
 ]  
 }, [data, task])  
 const progressTypes = [  
 ...new Set(data?.tasks.map((item) => item.progressType)),  
 ]  
 return (  
 <Box p={4} overflow="auto">  
 {}  
 <Stack  
 direction="row"  
 alignItems="right"  
 justifyContent="right"  
 position="sticky"  
 top={0}  
 zIndex={1}  
 gap={2}  
 sx={{ my: 2 }}  
 >  
 <GradientButton  
 bgColor="navy"  
 onClick={() =>  
 window.open(  
 `  
 /applications/${applicationId}  
 `,  
 '\_blank',  
 'rel=noopener,left=100,top=100,width=680,height=680 noreferrer'  
 )  
 }  
 >  
 Application  
 </GradientButton>  
 <GradientButton  
 bgColor="greenBlue"  
 onClick={() =>  
 window.open(  
 `  
 /applications/${applicationId}/cases/${caseId}  
 `,  
 '\_blank',  
 'rel=noopener,left=100,top=100,width=680,height=680 noreferrer'  
 )  
 }  
 >  
 Case  
 </GradientButton>  
 </Stack>  
 <Stack>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Grid container direction="column">  
 <ResponsiveGrid container direction="row" alignItems="center">  
 <ResponsiveGrid  
 item  
 container  
 isDesktop={isDesktop}  
 alignItems="center"  
 xs  
 sx={{ py: 2 }}  
 >  
 <Typography  
 variant="title2"  
 color="text.secondary"  
 onClick={() => setCaseProgressExpanded((prev) => !prev)}  
 sx={{ cursor: 'pointer' }}  
 >  
 {`${t('caseProgress', {  
 ns: 'commonTask',  
 })}`}{' '}  
 {caseProgressExpanded ? (  
 <ExpandMoreOutlined />  
 ) : (  
 <ExpandLessOutlined />  
 )}  
 </Typography>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 item  
 container  
 isDesktop={isDesktop}  
 alignItems="center"  
 justifyContent="flex-end"  
 xs  
 >  
 <Typography sx={{ color: '#4caf50', mr: 1 }}>{`${10} ${t(  
 'dayLeft'  
 )}`}</Typography>  
 <Typography sx={{ color: '#9e9e9e' }}>{`(${dayjs().format(  
 'DD/MM/YYYY'  
 )})`}</Typography>  
 </ResponsiveGrid>  
 </ResponsiveGrid>  
 {caseProgressExpanded && (  
 <Box maxHeight="38vh" overflow="auto">  
 {progressTypes.map((progressType) => {  
 return (  
 <Box>  
 <Typography  
 variant="title2"  
 color="text.secondary"  
 sx={{ cursor: 'pointer' }}  
 >  
 {`${progressType} Progress`}  
 </Typography>  
 <ResponsiveGrid container isDesktop={isDesktop}>  
 {data?.tasks  
 .filter(  
 (item) => item.progressType === progressType  
 )  
 .map((item) => {  
 return (  
 <Grid  
 item  
 xs={3}  
 sx={{  
 px: 1,  
 pb: 2,  
 }}  
 key={item?.\_id}  
 >  
 <CaseProgressItem  
 {...item}  
 onClick={(taskId) =>  
 navigate(  
 `/applications/${applicationId}/cases/${caseId}/tasks/${taskId}`  
 )  
 }  
 />  
 </Grid>  
 )  
 })}  
 </ResponsiveGrid>  
 </Box>  
 )  
 })}  
 </Box>  
 )}  
 </Grid>  
 )}  
 </ResponsiveForm>  
 </Stack>  
 <Divider variant="fullWidth" />  
 <Stack direction="row" justifyContent="space-between">  
 <Stack direction="column" gap={2} sx={{ my: 2 }}>  
 <Typography variant="title2" color="text.secondary">  
 Task: {TaskTitle[task?.taskType]}  
 </Typography>  
 <Typography variant="title" color="text.secondary">  
 Status: {task?.status}  
 </Typography>  
 </Stack>  
 {  
}  
 </Stack>  
 <Box sx={{ borderBottom: 1, borderColor: 'divider' }}>  
 {TabList?.length > 0 && (  
 <Tabs  
 value={tabIndex}  
 onChange={handleChange}  
 aria-label="basic tabs example"  
 sx={{ overflowX: 'auto' }}  
 scrollButtons  
 allowScrollButtonsMobile  
 variant="scrollable"  
 >  
 {TabList.map((item, index) => (  
 <Tab label={item.label} key={index} />  
 ))}  
 </Tabs>  
 )}  
 </Box>  
 <Box sx={{ maxHeight: 600, overflow: 'auto' }}>  
 {TabList.map((item, index) => (  
 <CustomTabPanel value={tabIndex} index={index}>  
 {item.component}  
 </CustomTabPanel>  
 ))}  
 </Box>  
 <Stack maxHeight="50vh" overflow="auto">  
 <EMinutes caseDetails={data} />  
 </Stack>  
 <Divider variant="fullWidth" sx={{ my: 4 }} />  
 <Button onClick={() => navigate(`/applications/${applicationId}`)}>  
 Back  
 </Button>  
 <Dialog  
 open={completingTask}  
 onClose={cancelCompleteTask}  
 title={  
 <Typography variant="title">{TaskTitle[task?.taskType]}</Typography>  
 }  
 message={  
 <Typography>  
 Are you sure you want to mark this task as complete?  
 </Typography>  
 }  
 onSubmit={() => completeTaskMutation.mutate()}  
 />  
 </Box>  
 )  
}  
export default TaskDetailPage

## File: bd-scs-backend-web/src/routes/user-management/add.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { Controller, useForm } from 'react-hook-form'  
import { useMutation } from '@tanstack/react-query'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import { useNavigate } from 'react-router-dom'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Typography from 'components/basic/Typography'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Autocomplete from 'src/components/basic/Autocomplete'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import colors from 'src/constants/colors'  
import { departments, roles, userTypes } from 'src/constants/options'  
import { createUser } from 'apis/user'  
export const schema = yup.object({  
 name: yup.string().required('Required'),  
 osdpLoginId: yup.string().required('Required'),  
 userType: yup  
 .object({ id: yup.string().required('Required') })  
 .required('Required'),  
 department: yup  
 .object({ id: yup.string().required('Required') })  
 .required('Required'),  
 role: yup  
 .object({ id: yup.string().required('Required') })  
 .required('Required'),  
})  
function AddUser() {  
 const navigate = useNavigate()  
 const { t, i18n } = useTranslation(['userManagement'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => createUser(payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 navigate('/user-management')  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 if (formData.position) formData.position = formData.position.id  
 if (formData.department) formData.department = formData.department.id  
 if (formData.role) formData.role = formData.role.id  
 if (formData.userType) formData.userType = formData.userType.id  
 mutation.mutate(formData)  
 }  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 {t('addUser')}  
 </Typography>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="name"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('name')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="osdpLoginId"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('osdpLoginId')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="password"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('password')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="password"  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="userType"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('userType')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.id?.message}  
 fullWidth  
 required  
 >  
 <Autocomplete  
 options={userTypes}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="osdpEmail"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('osdpEmail')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="email"  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="email"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('email')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="email"  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="notificationEmail"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('notificationEmail')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="email"  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="department"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('department')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.id?.message}  
 fullWidth  
 required  
 >  
 <Autocomplete  
 options={departments}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 {...field}  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="role"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => {  
 return (  
 <FormControl  
 title={t('role')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.id?.message}  
 fullWidth  
 required  
 >  
 <Autocomplete  
 options={roles}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 key={`${field.id}-${field?.value?.id}`}  
 {...field}  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="luPostName"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('luPostName')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="bdgis"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('bdgis')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="position"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('position')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="letterName"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('letterName')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="letterNameCn"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('letterNameCn')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="letterPosition"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('letterPosition')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="letterLongPosition"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('letterLongPosition')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="letterLongPositionCn"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('letterLongPositionCn')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <GradientButton  
 sx={{ width: 'fit-content' }}  
 onClick={() => handleSubmit(onSubmit)()}  
 loading={mutation.isLoading}  
 >  
 {t('button.submit')}  
 </GradientButton>  
 </Box>  
 )  
}  
export default AddUser

## File: bd-scs-backend-web/src/routes/user-management/detail.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { Controller, useForm } from 'react-hook-form'  
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import { useNavigate, useParams } from 'react-router-dom'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Select from '@mui/material/Select'  
import MenuItem from '@mui/material/MenuItem'  
import Typography from 'components/basic/Typography'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import KeyboardArrowDownIcon from '@mui/icons-material/KeyboardArrowDown'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Autocomplete from 'src/components/basic/Autocomplete'  
import Checkbox from 'src/components/basic/Checkbox'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import colors from 'src/constants/colors'  
import { departments, roles, positions, userTypes } from 'src/constants/options'  
import { updateUser, getUser, getUsers } from 'apis/user'  
import { schema } from './add'  
import UserPositionPicker from 'components/basic/UserPositionPicker'  
function UserDetails() {  
 const navigate = useNavigate()  
 const { userId } = useParams()  
 const { t, i18n } = useTranslation(['userManagement'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const queryClient = useQueryClient()  
 const {  
 data: users,  
 error: usersError,  
 isError: usersIsError,  
 isLoading: usersIsLoading,  
 } = useQuery({  
 queryKey: ['getUsers', {}],  
 queryFn: async () => await getUsers(),  
 })  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getUser', { userId }],  
 queryFn: async () => await getUser(userId),  
 })  
 console.log('user Type', data?.userType)  
 console.log('department', data?.department)  
 console.log('User locked', data?.lock)  
 const { handleSubmit, control, getValues, reset } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 osdpLoginId: data?.osdpLoginId,  
 userType: userTypes.filter((item) => item.id === data?.userType)[0],  
 name: data?.name,  
 email: data?.email,  
 password: 'DEFAULT\_PASSWORD',  
 osdpEmail: data?.osdpEmail,  
 notificationEmail: data?.notificationEmail,  
 luPostName: data?.luPostName,  
 bdgis: data?.bdgis,  
 team: data?.team,  
 letterName: data?.letterName,  
 letterNameCn: data?.letterNameCn,  
 letterPosition: data?.letterPosition,  
 letterPositionCn: data?.letterPositionCn,  
 letterLongPosition: data?.letterLongPosition,  
 letterLongPositionCn: data?.letterLongPositionCn,  
 notificationEmail: data?.notificationEmail,  
 position: data?.position,  
 department: departments.filter(  
 (item) => item.id === data?.department  
 )[0],  
 role: roles.filter((item) => item.id === data?.role)[0],  
 delegateTo: data?.delegateTo,  
 lock: data?.lock,  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateUser(userId, payload),  
 onSuccess: (data) => {  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 queryClient.invalidateQueries({ queryKey: ['getUser', { userId }] })  
 reset()  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 console.log('FORM DATA', formData)  
 if (formData.userType) formData.userType = formData.userType.id  
 if (formData.position) formData.position = formData.position.id  
 if (formData.department) formData.department = formData.department.id  
 if (formData.role) formData.role = formData.role.id  
 if (formData.password === 'DEFAULT\_PASSWORD') delete formData.password  
 mutation.mutate(formData)  
 }  
 const disabled = isLoading || mutation.isLoading  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 {t('userDetails')}  
 </Typography>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 spacing={4}  
 mb={4}  
 >  
 <Grid item md={6}>  
 <Controller  
 name="name"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('name')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="osdpLoginId"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('osdpLoginId')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="password"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('password')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="password"  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="userType"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => {  
 return (  
 <FormControl  
 title={t('userType')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.id?.message}  
 fullWidth  
 required  
 >  
 <Autocomplete  
 options={userTypes}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 key={`${field.id}-${field?.value?.id}`}  
 {...field}  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disabled={disabled}  
 />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 {  
}  
 <Grid item md={6}>  
 <Controller  
 name="email"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('email')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="email"  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {  
}  
 <Grid item md={6}>  
 <Controller  
 name="department"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => {  
 return (  
 <FormControl  
 title={t('department')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.id?.message}  
 fullWidth  
 required  
 >  
 <Autocomplete  
 options={departments}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 key={`${field.id}-${field?.value?.id}`}  
 {...field}  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disabled={disabled}  
 />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="role"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => {  
 return (  
 <FormControl  
 title={t('role')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.id?.message}  
 fullWidth  
 required  
 >  
 <Autocomplete  
 options={roles}  
 getOptionLabel={(option) =>  
 option.label  
 }  
 key={`${field.id}-${field?.value?.id}`}  
 {...field}  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disabled={disabled}  
 />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="luPostName"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('luPostName')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="bdgis"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('bdgis')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="team"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('team')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <UserPositionPicker  
 field={field}  
 disabled={disabled}  
 role="BS"  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="position"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={'Post'}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="letterName"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('letterName')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="letterNameCn"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('letterNameCn')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="letterPosition"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('letterPosition')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="letterPositionCn"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('letterPositionCn')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 {  
}  
 {  
}  
 <Grid item md={6}>  
 <Controller  
 name="delegateTo"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => {  
 return (  
 <FormControl  
 title={t('Delegate To')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth={true}  
 >  
 <Select  
 sx={{  
 height: 39,  
 bgcolor: 'white',  
 }}  
 IconComponent={  
 KeyboardArrowDownIcon  
 }  
 {...field}  
 >  
 {users?.map((u, index) => (  
 <MenuItem  
 key={index}  
 value={u.\_id}  
 disabled={  
 u.\_id ===  
 data?.\_id  
 }  
 >  
 {u.name}  
 </MenuItem>  
 ))}  
 </Select>  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 <Grid item md={6}>  
 <Controller  
 name="lock"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => {  
 console.log('CHECKBOX', field.value)  
 return (  
 <FormControl  
 title={t('lock')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="left"  
 error={invalid}  
 helperText={error?.id?.message}  
 >  
 <Checkbox  
 {...field}  
 checked={  
 field.value || false  
 }  
 onChange={(e, value) =>  
 field.onChange(value)  
 }  
 disabled={disabled}  
 />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <GradientButton  
 sx={{ width: 'fit-content' }}  
 onClick={() => handleSubmit(onSubmit)()}  
 loading={mutation.isLoading}  
 >  
 {t('button.save')}  
 </GradientButton>  
 </Box>  
 )  
}  
export default UserDetails

## File: bd-scs-backend-web/src/routes/user-management/list.jsx

import { useTranslation } from 'react-i18next'  
import dayjs from 'dayjs'  
import { useNavigate } from 'react-router-dom'  
import { useState, useEffect, useContext } from 'react'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import DataTable from 'components/basic/DataGrid'  
import Button from 'components/basic/Button'  
import Typography from 'components/basic/Typography'  
import colors from 'src/constants/colors'  
import { userContext } from 'src/context/UserProvider'  
function UserManagement() {  
 const navigate = useNavigate()  
 const [user, setUser] = useContext(userContext)  
 const { t, i18n } = useTranslation(['userManagement'])  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy">  
 {t('welcome')}  
 </Typography>  
 {  
}  
 <Stack  
 id="user-management-btns"  
 alignSelf="flex-end"  
 direction="row"  
 sx={{ width: 'fit-content' }}  
 mb={2}  
 spacing={2}  
 >  
 <Button width={20} onClick={() => navigate('add')}>  
 {t('button.createUser')}  
 </Button>  
 </Stack>  
 <DataTable  
 api={`/users${  
 user?.userType === 'SUPER\_ADMIN'  
 ? ''  
 : `?department=${user?.department}`  
 }`}  
 columns={[  
 {  
 field: 'name',  
 headerName: 'Name',  
 sortable: true,  
 width: 200,  
 },  
 {  
 field: 'email',  
 headerName: 'Email',  
 sortable: true,  
 width: 250,  
 },  
 {  
 field: 'position',  
 headerName: 'Position',  
 sortable: true,  
 width: 250,  
 },  
 {  
 field: 'department',  
 headerName: 'Department',  
 sortable: true,  
 width: 250,  
 },  
 {  
 field: 'role',  
 headerName: 'Role',  
 sortable: true,  
 width: 250,  
 },  
 {  
 field: 'team',  
 headerName: 'Team',  
 sortable: true,  
 width: 250,  
 },  
 {  
 field: 'lastLoginAt',  
 headerName: 'Last Login',  
 sortable: true,  
 width: 150,  
 valueFormatter: (params) => {  
 if (!params) return ''  
 return dayjs(params).format('YYYY-MM-DD-HH:mm')  
 },  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 color: '#fff',  
 },  
 '& .MuiDataGrid-columnHeaderCheckbox': {  
 color: '#fff',  
 },  
 '& .MuiButtonBase-root': {  
 color: '#000',  
 },  
 }}  
 onRowClick={(row) => {  
 navigate(`${row.id}`)  
 }}  
 />  
 </Box>  
 )  
}  
export default UserManagement

## File: bd-scs-backend-web/src/routes/dashboard.jsx

import React, { useContext } from 'react'  
import { useTranslation } from 'react-i18next'  
import useMediaQuery from '@mui/material/useMediaQuery'  
import Box from '@mui/material/Box'  
import Stack from '@mui/material/Stack'  
import InputAdornment from '@mui/material/InputAdornment'  
import DataTable from 'components/basic/DataGrid'  
import TextField from 'components/basic/TextField'  
import Typography from 'components/basic/Typography'  
import FormControl from 'components/basic/FormControl'  
import Autocomplete from 'components/basic/Autocomplete'  
import Button from 'components/basic/Button'  
import SquareButton from 'components/basic/Button/SquareIconButton'  
import BasicTable from 'components/basic/BasicTable'  
import UnderlineTabs from 'components/basic/UnderTabs'  
import GradientButton from 'components/basic/Button/GradientButton'  
import dayjs from 'dayjs'  
import SearchOutlinedIcon from '@mui/icons-material/SearchOutlined'  
import { TaskTitle } from 'src/constants/tasks'  
import { getCaseSummary } from 'src/apis/case'  
import { useQuery } from '@tanstack/react-query'  
import colors from 'src/constants/colors'  
import { Link, useNavigate } from 'react-router-dom'  
import { userContext } from 'src/context/UserProvider'  
import { submissionTypes } from 'src/constants/options'  
const TASKS = [  
 {  
 id: 1,  
 applicationNo: '2022-12-0002',  
 licensingCaseId: '00210814',  
 fileReferenceNumber: 'FRN-001',  
 category: 'CCC',  
 nature: 'Alt',  
 eApp: 'Y',  
 taskName: 'Desk study',  
 receivedDate: '17/06/2022',  
 targetReplyDate: '17/07/2022',  
 remainingDays: 6,  
 officer: 'SO-1',  
 address: '3 Fu Yue St., Lo Fu Ngam',  
 },  
 {  
 id: 2,  
 applicationNo: '2023-01-0045',  
 licensingCaseId: '00230845',  
 fileReferenceNumber: 'FRN-002',  
 category: 'BCA',  
 nature: 'New',  
 eApp: 'N',  
 taskName: 'Site inspection',  
 receivedDate: '20/01/2023',  
 targetReplyDate: '20/02/2023',  
 remainingDays: 12,  
 officer: 'SO-2',  
 address: '12 Shek Kip Mei St., Kowloon',  
 },  
 {  
 id: 3,  
 applicationNo: '2023-02-0098',  
 licensingCaseId: '00250988',  
 fileReferenceNumber: 'FRN-003',  
 category: 'DDA',  
 nature: 'Renovation',  
 eApp: 'Y',  
 taskName: 'Document review',  
 receivedDate: '05/02/2023',  
 targetReplyDate: '05/03/2023',  
 remainingDays: 8,  
 officer: 'SO-3',  
 address: '56 Nathan Rd., Tsim Sha Tsui',  
 },  
 {  
 id: 4,  
 applicationNo: '2023-03-0153',  
 licensingCaseId: '00271053',  
 fileReferenceNumber: 'FRN-004',  
 category: 'EEC',  
 nature: 'Modification',  
 eApp: 'N',  
 taskName: 'Preliminary assessment',  
 receivedDate: '10/03/2023',  
 targetReplyDate: '10/04/2023',  
 remainingDays: 15,  
 officer: 'SO-4',  
 address: '24 Hollywood Rd., Central',  
 },  
 {  
 id: 5,  
 applicationNo: '2023-04-0209',  
 licensingCaseId: '00291209',  
 fileReferenceNumber: 'FRN-005',  
 category: 'FBC',  
 nature: 'Upgrade',  
 eApp: 'Y',  
 taskName: 'Final review',  
 receivedDate: '25/04/2023',  
 targetReplyDate: '25/05/2023',  
 remainingDays: 10,  
 officer: 'SO-5',  
 address: '88 Queensway, Admiralty',  
 },  
]  
function Dashboard() {  
 const navigate = useNavigate()  
 const isDesktop = useMediaQuery('(min-width:600px)')  
 const [taskTab, setTaskTab] = React.useState(0)  
 const { t, i18n } = useTranslation(['dashboard'])  
 const [filter, setFilter] = React.useState({  
 filterType: 'FileReference',  
 filterValue: '',  
 })  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getCaseSummary'],  
 queryFn: async () => await getCaseSummary(),  
 })  
 const [user, setUser] = useContext(userContext)  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy">  
 {t('welcome')}  
 </Typography>  
 <Stack  
 direction={isDesktop ? 'row' : 'column'}  
 width="fit-content"  
 alignItems="center"  
 gap={1}  
 my={2}  
 >  
 <FormControl title={'Search case'} labelPlacement="start" fullWidth>  
 <Autocomplete  
 options={['FileReference', 'ApplicationNo', 'EmemoNo', '4+2No']}  
 getOptionLabel={(option) => t(`caseFilterAutocomplete.${option}`)}  
 value={filter.filterType}  
 onChange={(event, newValue) => {  
 if (newValue) {  
 setFilter({ ...filter, filterType: newValue })  
 }  
 }}  
 disableClearable  
 />  
 </FormControl>  
 <TextField  
 sx={{ minWidth: 400 }}  
 placeholder={t(`placeholder.input.${filter.filterType}`)}  
 value={filter.filterValue}  
 onChange={(event) => {  
 setFilter({  
 ...filter,  
 filterValue: event.target.value,  
 })  
 }}  
 InputProps={{  
 endAdornment: (  
 <InputAdornment position="end">  
 <Button  
 sx={{ p: `4px 10px`, minWidth: 10 }}  
 onClick={() =>  
 navigate(  
 '/advance-search/result?' +  
 filter.filterType +  
 '=' +  
 filter.filterValue  
 )  
 }  
 >  
 <SearchOutlinedIcon />  
 </Button>  
 </InputAdornment>  
 ),  
 }}  
 />  
 </Stack>  
 <Typography variant="title2" color="navy" sx={{ mt: 2 }}>  
 {t('outstandingCaseSummary')}  
 </Typography>  
 {!['TO', 'SE', 'SSE'].includes(user?.role) && (  
 <Box my={3}>  
 <BasicTable  
 onCellClick={(column, row) => {  
 navigate(  
 `/advance-search/result?Category=${row.category}${  
 row.nature ? `&Nature=${row.nature}` : ''  
 }${column.field == 'overdue' ? '&overdue=true' : ''}${  
 column.field == 'sevenDaysBeforeTargetReplyDate'  
 ? '&sevenDaysBeforeTargetReplyDate=true'  
 : ''  
 }`  
 )  
 }}  
 data={data}  
 row={[  
 {  
 id: 'sch',  
 label: 'SCH',  
 field: 'SCH',  
 category: 'SCH',  
 },  
 {  
 id: 'newsch',  
 category: 'SCH',  
 nature: 'NEW',  
 label: 'NEW',  
 field: 'NEW',  
 type: 'child',  
 },  
 {  
 id: 'revsch',  
 label: 'REV',  
 category: 'SCH',  
 nature: 'REV',  
 field: 'REV',  
 type: 'child',  
 },  
 {  
 id: 'altsch',  
 category: 'SCH',  
 nature: 'ALT',  
 label: 'ALT',  
 field: 'ALT',  
 type: 'child',  
 },  
 {  
 id: 'ccc',  
 label: 'CCC',  
 field: 'CCC',  
 category: 'CCC',  
 },  
 {  
 id: 'newccc',  
 label: 'NEW',  
 field: 'NEW',  
 type: 'child',  
 category: 'CCC',  
 nature: 'NEW',  
 },  
 {  
 id: 'revccc',  
 label: 'REV',  
 field: 'REV',  
 type: 'child',  
 category: 'CCC',  
 nature: 'REV',  
 },  
 {  
 id: 'altccc',  
 label: 'ALT',  
 field: 'ALT',  
 type: 'child',  
 category: 'CCC',  
 nature: 'ALT',  
 },  
 {  
 id: 'rnlccc',  
 label: 'RNL',  
 field: 'RNL',  
 type: 'child',  
 category: 'CCC',  
 nature: 'RNL',  
 },  
 {  
 id: 'nlhe',  
 label: 'NLHE',  
 field: 'NLHE',  
 category: 'NLHE',  
 },  
 {  
 id: 'newnlhe',  
 label: 'NEW',  
 field: 'NEW',  
 type: 'child',  
 category: 'NLHE',  
 nature: 'NEW',  
 },  
 {  
 id: 'revnlhe',  
 label: 'REV',  
 field: 'REV',  
 type: 'child',  
 category: 'NLHE',  
 nature: 'REV',  
 },  
 {  
 id: 'altnlhe',  
 label: 'ALT',  
 field: 'ALT',  
 type: 'child',  
 category: 'NLHE',  
 nature: 'ALT',  
 },  
 ]}  
 column={[  
 {  
 label: 'Total no. of case',  
 field: 'total',  
 headerStyle: {  
 backgroundColor: 'rgb(12, 50, 30)',  
 color: '#fff',  
 },  
 cellStyle: {  
 backgroundColor: 'rgba(12, 50, 30, 0.3)',  
 },  
 },  
 {  
 label: '7 days before target reply date',  
 field: 'sevenDaysBeforeTargetReplyDate',  
 headerStyle: {  
 backgroundColor: 'rgb(146, 102, 3)',  
 color: '#fff',  
 },  
 cellStyle: {  
 backgroundColor: 'rgba(146, 102, 3, 0.3)',  
 },  
 },  
 {  
 label: 'Overdue',  
 field: 'overdue',  
 headerStyle: {  
 backgroundColor: 'rgb(155, 33, 2)',  
 color: '#fff',  
 },  
 cellStyle: {  
 backgroundColor: 'rgba(155, 33, 2, 0.3)',  
 },  
 },  
 ]}  
 />  
 </Box>  
 )}  
 <UnderlineTabs  
 tabs={[  
 {  
 label: t('tabs.myTask'),  
 id: 'my-task',  
 },  
 {  
 label: t('tabs.myTeamTask'),  
 id: 'my-team-task',  
 },  
 ]}  
 onChange={(value) => setTaskTab(value)}  
 />  
 {  
}  
 <Box height={500} minHeight={500} overflow={'auto'}>  
 <DataTable  
 rows={TASKS}  
 columns={[  
 {  
 field: 'taskType',  
 headerName: 'Status',  
 width: 280,  
 valueGetter: (value, row) => {  
 return TaskTitle[row.taskType]  
 },  
 },  
 {  
 field: 'applicationNo',  
 headerName: 'Application No.',  
 width: 200,  
 valueGetter: (value, row) => {  
 return row.applicationDetails?.ApplicationNo  
 },  
 renderCell: (params) => (  
 <Link  
 to={`/applications/${params.row.applicationDetails?.\_id}`}  
 >  
 {params.row.applicationDetails?.ApplicationNo}  
 </Link>  
 ),  
 },  
 {  
 field: 'submissionCase',  
 headerName: 'Case No.',  
 width: 200,  
 renderCell: (params) => (  
 <Link  
 to={`/applications/${params.row.applicationDetails?.\_id}/cases/${params.row.submissionCase}`}  
 >  
 {params.row.submissionCase}  
 </Link>  
 ),  
 },  
 {  
 field: 'licensing',  
 headerName: 'Licensing Case ID',  
 width: 200,  
 renderCell: (params) => <dov></dov>,  
 },  
 {  
 field: 'fileRerence',  
 headerName: 'File Reference',  
 width: 180,  
 valueGetter: (value, row) => {  
 return row.fileReference?.[0]?.display  
 },  
 },  
 {  
 field: 'Category',  
 headerName: 'Category',  
 width: 180,  
 valueGetter: (value, row) => {  
 return row.caseDetails?.Category  
 },  
 },  
 {  
 field: 'Nature',  
 headerName: 'Nature',  
 width: 180,  
 valueGetter: (value, row) => {  
 return row.caseDetails?.Nature  
 },  
 },  
 {  
 field: 'seniorCaseOfficerReceive',  
 headerName: 'Senior Case Officer (Receive)',  
 width: 180,  
 valueGetter: (value, row) => {  
 return row.caseDetails?.seniorCaseOfficerReceive  
 },  
 },  
 {  
 field: 'caseOfficerReceive',  
 headerName: 'Case Officer (Receive)',  
 width: 180,  
 valueGetter: (value, row) => {  
 return row.caseDetails?.caseOfficerReceive  
 },  
 },  
 {  
 field: 'seniorCaseOfficerReply',  
 headerName: 'Senior Case Officer (Reply)',  
 width: 180,  
 valueGetter: (value, row) => {  
 return row.caseDetails?.seniorCaseOfficerReply  
 },  
 },  
 {  
 field: 'caseOfficerReply',  
 headerName: 'Case Officer (Reply)',  
 width: 180,  
 valueGetter: (value, row) => {  
 return row.caseDetails?.caseOfficerReply  
 },  
 },  
 {  
 field: 'receivedDate',  
 headerName: 'Received Date',  
 width: 200,  
 valueGetter: (value, row) => {  
 return (  
 row?.createdAt && dayjs(row?.createdAt).format('DD-MM-YYYY')  
 )  
 },  
 },  
 {  
 field: 'targetReplyDate',  
 headerName: 'Target Reply Date',  
 width: 200,  
 valueGetter: (value, row) => {  
 return (  
 row?.caseDetails?.TargetReplyDate &&  
 dayjs(row?.caseDetails?.TargetReplyDate).format('DD-MM-YYYY')  
 )  
 },  
 },  
 {  
 field: 'remainingDays',  
 headerName: 'Remaining Days',  
 width: 200,  
 valueGetter: (value, row) => {  
 return dayjs(row?.caseDetails?.TargetReplyDate).diff(  
 dayjs(row?.createdAt),  
 'days'  
 )  
 },  
 },  
 {  
 field: 'submissionType',  
 headerName: 'Submission Type',  
 width: 200,  
 valueGetter: (value, row) => {  
 return submissionTypes.find(  
 (item) => item.id == row?.caseDetails?.SubmissionType  
 )?.label  
 },  
 },  
 ]}  
 pageSize={5}  
 api={  
 (taskTab == 0 ? '/tasks/my-case-status?status=Active' : '/tasks/team')  
 }  
 rowsPerPageOptions={[5]}  
 checkboxSelection  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 color: '#fff',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 color: '#fff',  
 },  
 '& .MuiDataGrid-columnHeaderCheckbox': {  
 color: '#fff',  
 },  
 '& .MuiButtonBase-root': {  
 color: '#000',  
 },  
 }}  
 />  
 </Box>  
 {  
}  
 </Box>  
 )  
}  
export default Dashboard

## File: bd-scs-backend-web/src/routes/home.jsx

import Stack from '@mui/material/Stack';  
import Box from '@mui/material/Box';  
import TextField from 'components/basic/TextField';  
import GradientButton from 'components/basic/Button/GradientButton';  
import Button from 'components/basic/Button';  
import Autocomplete from 'components/basic/Autocomplete';  
import CaseProgressItem from 'components/ListItem/CaseProgressItem';  
function Home() {  
 return (  
 <Box m={2} overflow="auto">  
 <h3>Text Field</h3>  
 <Stack direction="row" spacing={2} width="fit-content">  
 <TextField value="active" />  
 <TextField disabled value="disabled" />  
 </Stack>  
 <h3>Multiline</h3>  
 <Stack direction="row" spacing={2} width="fit-content">  
 <TextField value="multiline" multiline rows={4} />  
 </Stack>  
 <h3>Gradient Button</h3>  
 <Stack direction="row" spacing={2} width="fit-content">  
 <GradientButton size="small">Small</GradientButton>  
 <GradientButton size="medium">Medium</GradientButton>  
 <GradientButton size="large">Large</GradientButton>  
 <GradientButton bgColor="red">Red</GradientButton>  
 <GradientButton disabled>Disabled</GradientButton>  
 </Stack>  
 <h3>Button</h3>  
 <Stack direction="row" spacing={2} width="fit-content">  
 <Button size="small">Small</Button>  
 <Button size="medium">Medium</Button>  
 <Button size="large">Large</Button>  
 <Button disabled>Disabled</Button>  
 <Button loading>Loading</Button>  
 </Stack>  
 <h3>Loading Button</h3>  
 <Stack direction="row" spacing={2} width="fit-content">  
 <Button size="small" loading>Small</Button>  
 <Button size="medium" loading>Medium</Button>  
 <Button size="large" loading>Large</Button>  
 </Stack>  
 <h3>Autocomplete</h3>  
 <Stack direction="row" spacing={2} width="fit-content">  
 <Autocomplete  
 options={[  
 { label: 'The Godfather', id: 1 },  
 { label: 'Pulp Fiction', id: 2 },  
 ]}  
 getOptionLabel={(option) => option.label}  
 />  
 </Stack>  
 <h3>ListItem</h3>  
 <Stack direction="row" spacing={2} >  
 <CaseProgressItem />  
 <CaseProgressItem disabled />  
 </Stack>  
 </Box>  
 )  
}  
export default Home;

## File: bd-scs-backend-web/src/routes/index.jsx

import { lazy, Suspense } from 'react'  
import { Routes, Route, Navigate } from 'react-router-dom'  
import Layout from 'src/components/Layout'  
import Home from './home'  
import UserManagement from './user-management/list'  
import AddUserPage from './user-management/add'  
import UserDetailPage from './user-management/detail'  
import Dashboard from './dashboard'  
import CreatePage from './application'  
import CreateApplicationPage from './application/add'  
import ApplicationDetailPage from './application/detail'  
import CreateCasePage from './case/add'  
import BuildingInformationSearch from './building-search'  
import Report from './report'  
import AdvanceSearch from './advance-search'  
import AdvanceSearchResult from './advance-search/result'  
import Forms from './forms'  
import A2Form from './forms/a2'  
import Jokc02Form from './forms/jokc02'  
import Cccaimh3Form from './forms/cccaimh3'  
import Cccaic2Form from './forms/cccaic2'  
import F2Form from './forms/f2'  
import G2Form from './forms/g2'  
import E2Form from './forms/e2'  
import B2Form from './forms/b2'  
import C2Form from './forms/c2'  
import FormDetail from './legacy\_formDetail'  
import ApplicationDetails from './applicationDetails/old\_index'  
import SubmissionDetails from './submissionDetails'  
import SubmitDocForm from './forms/submitDoc'  
import InspectReqForm from './forms/inspectReq'  
import WithdrawAppForm from './forms/withdrawApp'  
import UpdateInfoForm from './forms/updateInfo'  
import AppointApForm from './forms/appointAp'  
import Profile from './profile'  
import TaskDetail from './task'  
import CaseDetailPage from './case/detail'  
import EfolioSearch from './efolio-search'  
function AppRouter() {  
 return (  
 <Suspense fallback={<div />}>  
 <Routes>  
 <Route exact path="/" element={<Layout />}>  
 <Route index element={<Navigate to="dashboard" replace />} />  
 <Route path="home" element={<Home />} />  
 <Route path="profile" element={<Profile />} />  
 <Route path="dashboard" element={<Dashboard />} />  
 <Route path="applications">  
 <Route index element={<CreatePage />} />  
 <Route exact path="add" element={<CreateApplicationPage />} />  
 <Route path=":applicationId">  
 <Route index element={<ApplicationDetailPage />} />  
 <Route exact path="cases/add" element={<CreateCasePage />} />  
 <Route path="cases/:caseId">  
 <Route index element={<CaseDetailPage />} />  
 <Route path="tasks/:taskId">  
 <Route index element={<TaskDetail />} />  
 </Route>  
 </Route>  
 </Route>  
 </Route>  
 <Route path="report" element={<Report />} />  
 <Route path="building-search" element={<BuildingInformationSearch />} />  
 <Route path="efolio-search" element={<EfolioSearch />} />  
 <Route path="advance-search">  
 <Route index element={<AdvanceSearch />} />  
 <Route path="result" element={<AdvanceSearchResult />} />  
 <Route path="forms/:formId" element={<FormDetail />} />  
 </Route>  
 <Route path="user-management">  
 <Route index element={<UserManagement />} />  
 <Route path="add" element={<AddUserPage />} />  
 <Route path=":userId" element={<UserDetailPage />} />  
 </Route>  
 <Route path="forms/:formId" element={<FormDetail />} />  
 <Route path="forms">  
 <Route index element={<Forms />} />  
 <Route path="jokc02" exact element={<Jokc02Form />} />  
 <Route path="cccaimh3" exact element={<Cccaimh3Form />} />  
 <Route path="cccaic2" exact element={<Cccaic2Form />} />  
 <Route path="e2" exact element={<E2Form />} />  
 <Route path="a2" exact element={<A2Form />} />  
 </Route>  
 </Route>  
 </Routes>  
 </Suspense>  
 )  
}  
export default AppRouter

## File: bd-scs-backend-web/src/routes/login.jsx

import React from 'react'  
import styled from '@emotion/styled'  
import muiStyled from '@mui/material/styles/styled'  
import { useNavigate } from 'react-router-dom'  
import { yupResolver } from '@hookform/resolvers/yup'  
import \* as yup from 'yup'  
import { Controller, useForm } from 'react-hook-form'  
import { useMutation } from '@tanstack/react-query'  
import Box from '@mui/material/Box'  
import Card from '@mui/material/Card'  
import Stack from '@mui/material/Stack'  
import colors from 'src/constants/colors'  
import Typography from 'src/components/basic/Typography'  
import StarIcon from '@mui/icons-material/Star'  
import Button from '@mui/material/Button'  
import FormControl from 'src/components/basic/FormControl'  
import TextField from 'src/components/basic/TextField'  
import Autocomplete from 'src/components/basic/Autocomplete'  
import GradientButton from 'src/components/basic/Button/GradientButton'  
import Header from 'src/components/Login/Header'  
import LoadingCard from 'src/components/Login/LoadingCard'  
import DelegateDialog from 'src/components/Dialog/DelegateDialog'  
import { userContext } from 'src/context/UserProvider'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import constants from 'src/constants'  
import { departments } from 'src/constants/options'  
import { login } from 'src/apis/auth'  
const schema = yup.object({  
 username: yup.string().required('Required'),  
 password: yup.string().required('Required'),  
 department: yup  
 .object({ id: yup.string().required('Required') })  
 .required('Required'),  
})  
function Login() {  
 const [isLoading, setIsLoading] = React.useState(false)  
 const [user, setUser] = React.useContext(userContext)  
 const [delegateList, setDelegateList] = React.useState([])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const [open, setOpen] = React.useState(false)  
 const [selectedValue, setSelectedValue] = React.useState('')  
 const handleOpen = () => {  
 console.log('open')  
 setOpen(true)  
 }  
 const handleClose = (value) => {  
 setOpen(false)  
 setSelectedValue(value)  
 console.log(selectedValue)  
 }  
 const {  
 handleSubmit,  
 control,  
 formState: { errors },  
 watch,  
 setValue,  
 } = useForm({  
 resolver: yupResolver(schema),  
 defaultValues: {  
 username: '',  
 password: '',  
 department: {  
 label: 'Buildings Department',  
 id: 'building-department',  
 },  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => login(payload),  
 onSuccess: (data) => {  
 localStorage.setItem('accessToken', data?.accessToken)  
 localStorage.setItem('refreshToken', data?.refreshToken)  
 console.log("Login data", data)  
 if (Array.isArray(data)) {  
 setDelegateList(data)  
 setIsLoading(false)  
 handleOpen()  
 } else {  
 setUser(data?.user)  
 }  
 },  
 onError: (error) => {  
 console.log('error', error)  
 setIsLoading(false)  
 setSnackbar({  
 type: 'error',  
 message: error?.error,  
 })  
 },  
 })  
 const onLogin = (formValue) => {  
 setIsLoading(true)  
 mutation.mutate({  
 username: formValue.username,  
 password: formValue.password,  
 grant\_type: 'password',  
 client\_id: constants.client\_id,  
 client\_secret: constants.client\_secret,  
 })  
 }  
 React.useEffect(() => {  
 if (selectedValue) {  
 console.log('selectedValue', selectedValue)  
 setIsLoading(true)  
 mutation.mutate({  
 username: watch('username'),  
 password: watch('password'),  
 grant\_type: 'password',  
 client\_id: constants.client\_id,  
 client\_secret: constants.client\_secret,  
 chosen\_user: selectedValue,  
 })  
 setSelectedValue('')  
 }  
 }, [selectedValue])  
 if (isLoading) {  
 return <LoadingCard />  
 }  
 return (  
 <Container>  
 <CardContainer>  
 <Header />  
 <Box sx={{ my: 1 }}>  
 <Typography variant="h5">User Login</Typography>  
 <Typography>  
 Please sign in with your Username and Password.  
 </Typography>  
 </Box>  
 <Controller  
 name="username"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Username"  
 labelPlacement="top"  
 sx={{ mb: 2 }}  
 error={invalid}  
 helperText={error?.message}  
 >  
 <TextField {...field} />  
 </FormControl>  
 )}  
 />  
 <Controller  
 name="password"  
 control={control}  
 render={({ field, fieldState: { invalid, error } }) => (  
 <FormControl  
 title="Password"  
 labelPlacement="top"  
 sx={{ mb: 2 }}  
 error={invalid}  
 helperText={error?.message}  
 >  
 <TextField {...field} type="password" />  
 </FormControl>  
 )}  
 />  
 <DelegateDialog  
 selectedValue={selectedValue}  
 delegateList={delegateList}  
 open={open}  
 onClose={handleClose}  
 />  
 <GradientButton onClick={() => handleSubmit(onLogin)()}>  
 Login  
 </GradientButton>  
 </CardContainer>  
 </Container>  
 )  
}  
export const Container = styled(Box)(({ theme }) => ({  
 display: 'flex',  
 justifyContent: 'center',  
 alignItems: 'center',  
 height: '100vh',  
 width: '100%',  
 backgroundColor: colors.main,  
 overflow: 'hidden',  
}))  
export const CardContainer = muiStyled(Card)(({ theme }) => ({  
 display: 'flex',  
 flexDirection: 'column',  
 padding: theme.spacing(4),  
 borderRadius: theme.shape.borderRadius,  
 boxShadow: theme.shadows[3],  
 backgroundColor: theme.palette.background.paper,  
 width: '550px',  
 overflow: 'auto',  
}))  
export default Login

## File: bd-scs-backend-web/src/routes/profile.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import { Controller, useForm } from 'react-hook-form'  
import { useMutation, useQuery, useQueryClient } from '@tanstack/react-query'  
import \* as yup from 'yup'  
import { yupResolver } from '@hookform/resolvers/yup'  
import { useNavigate, useParams } from 'react-router-dom'  
import Box from '@mui/material/Box'  
import Grid from '@mui/material/Grid'  
import Stack from '@mui/material/Stack'  
import Typography from 'components/basic/Typography'  
import ResponsiveForm from 'components/basic/ResponsiveForm'  
import ResponsiveGrid from 'components/basic/ResponsiveGrid'  
import FormControl from 'components/basic/FormControl'  
import TextField from 'components/basic/TextField'  
import GradientButton from 'components/basic/Button/GradientButton'  
import Autocomplete from 'src/components/basic/Autocomplete'  
import { snackbarContext } from 'src/context/SnackbarProvider'  
import { userContext } from 'src/context/UserProvider'  
import colors from 'src/constants/colors'  
import { departments, roles, positions } from 'src/constants/options'  
import { updateUser, getUser } from 'apis/user'  
export const schema = yup.object({  
 osdpLoginId: yup.string().required('Required'),  
 name: yup.string().required('Required'),  
})  
function MyProfile() {  
 const navigate = useNavigate()  
 const [user, setUser] = React.useContext(userContext)  
 const { t, i18n } = useTranslation(['userManagement'])  
 const [snackbar, setSnackbar] = React.useContext(snackbarContext)  
 const queryClient = useQueryClient()  
 const { data, error, isError, isLoading } = useQuery({  
 queryKey: ['getUser', { userId: user?.\_id }],  
 queryFn: async () => await getUser(user.\_id),  
 })  
 const { handleSubmit, control } = useForm({  
 resolver: yupResolver(schema),  
 values: {  
 name: data?.name,  
 osdpLoginId: data?.osdpLoginId,  
 email: data?.email,  
 password: 'DEFAULT\_PASSWORD',  
 },  
 })  
 const mutation = useMutation({  
 mutationFn: (payload) => updateUser(user?.\_id, payload),  
 onSuccess: (data) => {  
 setUser((prev) => ({  
 ...prev,  
 ...data,  
 }))  
 setSnackbar({  
 type: 'success',  
 message: t('success', { ns: 'common' }),  
 })  
 queryClient.invalidateQueries({  
 queryKey: ['getUser', { userId: user?.\_id }],  
 })  
 },  
 onError: (error) => {  
 setSnackbar({  
 type: 'error',  
 message: error?.message,  
 })  
 },  
 })  
 const onSubmit = (formData) => {  
 let changes = {}  
 if (formData.password != 'DEFAULT\_PASSWORD') {  
 changes['password'] = formData.password  
 }  
 mutation.mutate(changes)  
 }  
 const disabled = isLoading || mutation.isLoading  
 return (  
 <Box  
 p={4}  
 bgcolor={colors.bgColor}  
 height="100%"  
 display="flex"  
 flexDirection="column"  
 overflow="auto"  
 >  
 <Typography variant="title2" color="navy" align="center" mb={2}>  
 Profile  
 </Typography>  
 <ResponsiveForm>  
 {(isDesktop) => (  
 <Stack direction="column" my={4}>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="name"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('name')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 disabled={disabled}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="email"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('email')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="email"  
 disabled={true}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 <ResponsiveGrid  
 container  
 isDesktop={isDesktop}  
 gap={3}  
 mb={4}  
 >  
 <Grid item xs>  
 <Controller  
 name="osdpLoginId"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => (  
 <FormControl  
 title={t('osdpLoginId')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 required  
 >  
 <TextField  
 {...field}  
 disabled={true}  
 />  
 </FormControl>  
 )}  
 />  
 </Grid>  
 <Grid item xs>  
 <Controller  
 name="password"  
 control={control}  
 render={({  
 field,  
 fieldState: { invalid, error },  
 }) => {  
 return (  
 <FormControl  
 title={t('password')}  
 formLabelProps={{  
 variant: 'navyHeader',  
 }}  
 labelPlacement="top"  
 error={invalid}  
 helperText={error?.message}  
 fullWidth  
 >  
 <TextField  
 {...field}  
 type="password"  
 disabled={disabled}  
 />  
 </FormControl>  
 )  
 }}  
 />  
 </Grid>  
 </ResponsiveGrid>  
 </Stack>  
 )}  
 </ResponsiveForm>  
 <GradientButton  
 sx={{ width: 'fit-content' }}  
 onClick={() => handleSubmit(onSubmit)()}  
 loading={mutation.isLoading}  
 >  
 {t('button.save')}  
 </GradientButton>  
 </Box>  
 )  
}  
export default MyProfile

## File: bd-scs-backend-web/src/routes/report.jsx

import React from 'react'  
import { useTranslation } from 'react-i18next'  
import useMediaQuery from '@mui/material/useMediaQuery'  
import Box from '@mui/material/Box'  
import Typography from 'components/basic/Typography'  
import DataTable from 'components/basic/DataGrid'  
import CustomDatePicker from 'components/basic/DatePicker'  
import \* as XLSX from 'xlsx-js-style'  
import GradientButton from 'components/basic/Button/GradientButton'  
import colors from 'src/constants/colors'  
import { BorderAll, Gradient } from '@mui/icons-material'  
function Report() {  
 const { t, i18n } = useTranslation(['report'])  
 const isDesktop = useMediaQuery('(min-width:800px)')  
 const isTemplateAvailable = (templateNo) => {  
 return [1, 2, 3, 4, 12, 13].includes(templateNo)  
 }  
 const getTemplateData = (templateNo) => {  
 let data = []  
 const perMonthTemplate = [  
 [],  
 [],  
 [  
 '',  
 'Case Officer',  
 'Jan',  
 'Feb',  
 'Mar',  
 'Apr',  
 'May',  
 'Jun',  
 'Jul',  
 'Aug',  
 'Sep',  
 'Oct',  
 'Nov',  
 'Dec',  
 'Total',  
 ],  
 ['', '', '', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ['', 'Total', '', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ]  
 if (templateNo < 5) {  
 data = perMonthTemplate  
 // Template-specific headers are added in downloadTemplate  
 } else if (templateNo === 12) {  
 data = [  
 [],  
 [  
 '',  
 'Case officer',  
 'No. of cases with Nth Reminder Received',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 ],  
 [  
 '',  
 '',  
 '1st',  
 '2nd',  
 '3rd',  
 '4th',  
 '5th',  
 '6th',  
 '7th',  
 '8th',  
 '9th',  
 '10th',  
 '11th',  
 'Total',  
 ],  
 ['', 'BSLIC1', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ['', 'NEW', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ['', 'REV', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ['', 'ALT', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ['', 'Total', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ]  
 } else if (templateNo === 13) {  
 data = [  
 [],  
 ['', '', '', 'No. of cases', '', '', ' '],  
 [  
 '',  
 'Month',  
 'Cumulative licensing cases received',  
 'Cumulative licensing cases replied',  
 'Licensing cases received',  
 'Licensing cases replied',  
 'Cumulative overdue licensing cases ',  
 ],  
 ['', 'Jan', '', '', '', '', ''],  
 ['', 'Feb', '', '', '', '', ''],  
 ['', 'Mar', '', '', '', '', ''],  
 ['', 'Apr', '', '', '', '', ''],  
 ['', 'May', '', '', '', '', ''],  
 ['', 'Jun', '', '', '', '', ''],  
 ['', 'Jul', '', '', '', '', ''],  
 ['', 'Aug', '', '', '', '', ''],  
 ['', 'Sep', '', '', '', '', ''],  
 ['', 'Oct', '', '', '', '', ''],  
 ['', 'Nov', '', '', '', '', ''],  
 ['', 'Dec', '', '', '', '', ''],  
 ]  
 }  
 return data  
 }  
 const downloadTemplate = (templateNo) => {  
 let data = []  
 const perMonthTemplate = [  
 [],  
 [],  
 [  
 '',  
 'Case Officer',  
 'Jan',  
 'Feb',  
 'Mar',  
 'Apr',  
 'May',  
 'Jun',  
 'Jul',  
 'Aug',  
 'Sep',  
 'Oct',  
 'Nov',  
 'Dec',  
 'Total',  
 ],  
 ['', '', '', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ['', 'Total', '', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ]  
 if (templateNo < 5) {  
 data = perMonthTemplate  
 } else if (templateNo === 12) {  
 data = [  
 [],  
 [  
 '',  
 'Case officer',  
 'No. of cases with Nth Reminder Received',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 ],  
 [  
 '',  
 '',  
 '1st',  
 '2nd',  
 '3rd',  
 '4th',  
 '5th',  
 '6th',  
 '7th',  
 '8th',  
 '9th',  
 '10th',  
 '11th',  
 'Total',  
 ],  
 ['', 'BSLIC1', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ['', 'NEW', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ['', 'REV', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ['', 'ALT', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ['', 'Total', '', '', '', '', '', '', '', '', '', '', '', ''],  
 ]  
 } else if (templateNo === 13) {  
 data = [  
 [],  
 ['', '', '', 'No. of cases', '', '', ' '],  
 [  
 '',  
 'Month',  
 'Cumulative licensing cases received',  
 'Cumulative licensing cases replied',  
 'Licensing cases received',  
 'Licensing cases replied',  
 'Cumulative overdue licensing cases ',  
 ],  
 ['', 'Jan', '', '', '', '', ''],  
 ['', 'Feb', '', '', '', '', ''],  
 ['', 'Mar', '', '', '', '', ''],  
 ['', 'Apr', '', '', '', '', ''],  
 ['', 'May', '', '', '', '', ''],  
 ['', 'Jun', '', '', '', '', ''],  
 ['', 'Jul', '', '', '', '', ''],  
 ['', 'Aug', '', '', '', '', ''],  
 ['', 'Sep', '', '', '', '', ''],  
 ['', 'Oct', '', '', '', '', ''],  
 ['', 'Nov', '', '', '', '', ''],  
 ['', 'Dec', '', '', '', '', ''],  
 ]  
 }  
 console.log('templateNo', templateNo)  
 switch (templateNo) {  
 case 1:  
 data[1] = [  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 'Case Received in Month',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 ]  
 break  
 case 2:  
 data[1] = [  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 'Case Replied in Month',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 ]  
 break  
 case 3:  
 data[1] = [  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 'Outstanding Cases in Month',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 ]  
 break  
 case 4:  
 data[1] = [  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 'Overdue Cases in Month',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 '',  
 ]  
 break  
 default:  
 data[0] = []  
 }  
 // Create a worksheet from the data  
 const worksheet = XLSX.utils.aoa\_to\_sheet(data)  
 // Merge cells for "Case Received in Year" title  
 if (templateNo < 5) {  
 worksheet['!merges'] = [{ s: { r: 1, c: 7 }, e: { r: 1, c: 9 } }]  
 worksheet['H2'].s = {  
 font: { bold: true },  
 alignment: { horizontal: 'center' },  
 fill: { fgColor: { rgb: 'DCE6F1' } },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 const headerStyle = {  
 font: { bold: true },  
 fill: { fgColor: { rgb: 'DCE6F1' } },  
 alignment: { horizontal: 'center' },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 const headers = [  
 'H2',  
 'I2',  
 'J2',  
 'B3',  
 'C3',  
 'D3',  
 'E3',  
 'F3',  
 'G3',  
 'H3',  
 'I3',  
 'J3',  
 'K3',  
 'L3',  
 'M3',  
 'N3',  
 'O3',  
 ]  
 headers.forEach((header) => {  
 worksheet[header].s = headerStyle  
 })  
 const contentStyle = {  
 alignment: { horizontal: 'center' },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 const content = [  
 'B4',  
 'C4',  
 'D4',  
 'E4',  
 'F4',  
 'G4',  
 'H4',  
 'I4',  
 'J4',  
 'K4',  
 'L4',  
 'M4',  
 'N4',  
 'O4',  
 ]  
 content.forEach((cell) => {  
 worksheet[cell].s = contentStyle  
 })  
 const total = [  
 'B5',  
 'C5',  
 'D5',  
 'E5',  
 'F5',  
 'G5',  
 'H5',  
 'I5',  
 'J5',  
 'K5',  
 'L5',  
 'M5',  
 'N5',  
 'O5',  
 ]  
 const totalStyle = {  
 alignment: { horizontal: 'center' },  
 fill: { fgColor: { rgb: 'DCE6F1' } },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 total.forEach((cell) => {  
 worksheet[cell].s = totalStyle  
 })  
 worksheet['!cols'] = Array(data[1].length).fill({ wch: 13 })  
 } else if (templateNo === 12) {  
 worksheet['!merges'] = [  
 { s: { r: 1, c: 2 }, e: { r: 1, c: 13 } },  
 { s: { r: 1, c: 1 }, e: { r: 2, c: 1 } },  
 ]  
 const blueStyle = {  
 alignment: { horizontal: 'center' },  
 fill: { fgColor: { rgb: 'DCE6F1' } },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 const whiteStyle = {  
 alignment: { horizontal: 'center' },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 const yellowStyle = {  
 alignment: { horizontal: 'center' },  
 fill: { fgColor: { rgb: 'FFFF00' } },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 const greyStyle = {  
 alignment: { horizontal: 'center' },  
 fill: { fgColor: { rgb: 'DDDDDD' } },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 for (let r = 1; r < data.length; r++) {  
 for (let c = 1; c < data[r].length; c++) {  
 const cellAddress = XLSX.utils.encode\_cell({ r, c })  
 if (!worksheet[cellAddress]) worksheet[cellAddress] = {}  
 if (r == 1 || r == 2 || r == 7) {  
 worksheet[cellAddress].s = blueStyle  
 }  
 }  
 }  
 for (let c = 1; c < data[3].length - 1; c++) {  
 let r = 3  
 const cellAddress = XLSX.utils.encode\_cell({ r, c })  
 if (!worksheet[cellAddress]) worksheet[cellAddress] = {}  
 worksheet[cellAddress].s = whiteStyle  
 }  
 for (let r = 3; r < data.length - 1; r++) {  
 let c = data[3].length - 1  
 const cellAddress = XLSX.utils.encode\_cell({ r, c })  
 if (!worksheet[cellAddress]) worksheet[cellAddress] = {}  
 worksheet[cellAddress].s = whiteStyle  
 }  
 for (let c = 1; c < data[4].length - 1; c++) {  
 let r = 4  
 const cellAddress = XLSX.utils.encode\_cell({ r, c })  
 if (!worksheet[cellAddress]) worksheet[cellAddress] = {}  
 worksheet[cellAddress].s = yellowStyle  
 }  
 for (let r = 5; r < 7; r++) {  
 for (let c = 1; c < data[r].length - 1; c++) {  
 const cellAddress = XLSX.utils.encode\_cell({ r, c })  
 if (!worksheet[cellAddress]) worksheet[cellAddress] = {}  
 worksheet[cellAddress].s = greyStyle  
 }  
 }  
 worksheet['B2'].s = {  
 alignment: { vertical: 'center' },  
 fill: { fgColor: { rgb: 'DCE6F1' } },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 worksheet['C2'].s = {  
 alignment: { horizontal: 'center' },  
 fill: { fgColor: { rgb: 'DCE6F1' } },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 worksheet['!cols'] = Array(data[1].length).fill({ wch: 10 })  
 } else if (templateNo === 13) {  
 worksheet['!merges'] = [{ s: { r: 1, c: 3 }, e: { r: 1, c: 4 } }]  
 worksheet['D2'].s = {  
 alignment: { horizontal: 'center' },  
 }  
 const cellStyle = {  
 alignment: { horizontal: 'center' },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 }  
 for (let r = 2; r < data.length; r++) {  
 for (let c = 1; c < data[r].length; c++) {  
 const cellAddress = XLSX.utils.encode\_cell({ r, c })  
 if (!worksheet[cellAddress]) worksheet[cellAddress] = {}  
 worksheet[cellAddress].s = cellStyle  
 }  
 }  
 const headerStyle = {  
 alignment: { horizontal: 'center' },  
 fill: { fgColor: { rgb: 'DCE6F1' } },  
 border: {  
 top: { style: 'thin', color: { rgb: '000000' } },  
 bottom: { style: 'thin', color: { rgb: '000000' } },  
 left: { style: 'thin', color: { rgb: '000000' } },  
 right: { style: 'thin', color: { rgb: '000000' } },  
 },  
 width: 30,  
 }  
 const headers = ['B3', 'C3', 'D3', 'E3', 'F3', 'G3']  
 headers.forEach((header) => {  
 worksheet[header].s = headerStyle  
 })  
 worksheet['!cols'] = [  
 { wch: 8 },  
 { wch: 30 },  
 { wch: 30 },  
 { wch: 30 },  
 { wch: 30 },  
 { wch: 30 },  
 { wch: 30 },  
 ]  
 }  
 const workbook = XLSX.utils.book\_new()  
 XLSX.utils.book\_append\_sheet(workbook, worksheet, 'Sheet1')  
 const excelBuffer = XLSX.write(workbook, {  
 bookType: 'xlsx',  
 type: 'binary',  
 })  
 const buffer = new ArrayBuffer(excelBuffer.length)  
 const view = new Uint8Array(buffer)  
 for (let i = 0; i < excelBuffer.length; i++) {  
 view[i] = excelBuffer.charCodeAt(i) & 0xff  
 }  
 const dataBlob = new Blob([buffer], {  
 type: 'application/octet-stream',  
 })  
 const link = document.createElement('a')  
 link.href = URL.createObjectURL(dataBlob)  
 switch (templateNo) {  
 case 1:  
 link.download = 'Total received cases per month.xlsx'  
 break  
 case 2:  
 link.download = 'Total replied cases per month.xlsx'  
 break  
 case 3:  
 link.download = 'Total outstanding cases per month.xlsx'  
 break  
 case 4:  
 link.download = 'Total overdue cases per month.xlsx'  
 break  
 case 12:  
 link.download = 'Outstanding cases with reminder.xlsx'  
 break  
 case 13:  
 link.download = 'Performance report.xlsx'  
 break  
 default:  
 link.download = 'report.xlsx'  
 }  
 document.body.appendChild(link)  
 link.click()  
 document.body.removeChild(link)  
 }  
 const printTemplate = (templateNo) => {  
 const data = getTemplateData(templateNo)  
 const printableContent = generatePrintableHTML(data, templateNo)  
 const printWindow = window.open('', '\_blank')  
 printWindow.document.write(`  
 <html>  
 <head>  
 <title>${getTemplateTitle(templateNo)}</title>  
 <style>  
 body { font-family: Arial; margin: 20px; }  
 table { border-collapse: collapse; width: 100%; }  
 th, td { border: 1px solid #ddd; padding: 8px; text-align: center; }  
 th { background-color: #f2f2f2; }  
 </style>  
 </head>  
 <body>  
 <h2>${getTemplateTitle(templateNo)}</h2>  
 ${printableContent}  
 <script>  
 window.onload = function() {  
 window.print();  
 window.close();  
 }  
 </script>  
 </body>  
 </html>  
 `)  
 printWindow.document.close()  
 }  
 const previewTemplate = (templateNo) => {  
 const data = getTemplateData(templateNo)  
 const printableContent = generatePrintableHTML(data, templateNo)  
 const previewWindow = window.open('', '\_blank')  
 previewWindow.document.write(`  
 <html>  
 <head>  
 <title>Preview: ${getTemplateTitle(templateNo)}</title>  
 <style>  
 body { font-family: Arial; margin: 20px; }  
 table { border-collapse: collapse; width: 100%; }  
 th, td { border: 1px solid #ddd; padding: 8px; text-align: center; }  
 th { background-color: #f2f2f2; }  
 </style>  
 </head>  
 <body>  
 <h2>Preview: ${getTemplateTitle(templateNo)}</h2>  
 ${printableContent}  
 <div style="margin-top: 20px;">  
 <button onclick="window.print()">Print</button>  
 <button onclick="window.close()" style="margin-left: 10px;">Close</button>  
 </div>  
 </body>  
 </html>  
 `)  
 previewWindow.document.close()  
 }  
 const getTemplateTitle = (templateNo) => {  
 const titles = {  
 1: 'Total received cases per month',  
 2: 'Total replied cases per month',  
 3: 'Total outstanding cases per month',  
 4: 'Total overdue cases per month',  
 12: 'Outstanding cases with reminder',  
 13: 'Performance report',  
 }  
 return titles[templateNo] || 'Report'  
 }  
 const generatePrintableHTML = (data, templateNo) => {  
 let html = '<table>'  
 html += '<thead><tr>'  
 data[2].forEach((cell) => {  
 html += `<th>${cell}</th>`  
 })  
 html += '</tr></thead>'  
 html += '<tbody>'  
 for (let i = 3; i < data.length; i++) {  
 html += '<tr>'  
 data[i].forEach((cell) => {  
 html += `<td>${cell}</td>`  
 })  
 html += '</tr>'  
 }  
 html += '</tbody></table>'  
 return html  
 }  
 return (  
 <Box p={4} bgcolor={colors.bgColor} height="100%" overflow="auto">  
 <Box marginBottom="2%">  
 <Typography variant="title2" color="navy">  
 {t('Report')}  
 </Typography>  
 </Box>  
 <Box marginBottom="2%" display="flex" alignItems="center" gap={2}>  
 <Typography variant="title" color="navy">  
 {t('From')}  
 </Typography>  
 <CustomDatePicker />  
 <Typography variant="title" color="navy">  
 {t('To')}  
 </Typography>  
 <CustomDatePicker />  
 </Box>  
 <DataTable  
 staticData={[  
 { \_id: 1, reportType: 'Total received cases per month' },  
 { \_id: 2, reportType: 'Total replied cases per month' },  
 { \_id: 3, reportType: 'Total outstanding cases per month' },  
 { \_id: 4, reportType: 'Total overdue cases per month' },  
 {  
 \_id: 5,  
 reportType: 'Total Audit Cases and corresponding results',  
 },  
 {  
 \_id: 6,  
 reportType:  
 'Total applications through e-submission and paper submission',  
 },  
 { \_id: 7, reportType: 'Total opted and Non-opt cases' },  
 {  
 \_id: 8,  
 reportType: 'Total accepted and rejected PTS cases',  
 },  
 {  
 \_id: 9,  
 reportType: 'Cumulative received cases per month',  
 },  
 {  
 \_id: 10,  
 reportType: 'Cumulative replied cases per month',  
 },  
 {  
 \_id: 11,  
 reportType: 'Cumulative Outstanding cases per month',  
 },  
 { \_id: 12, reportType: 'Outstanding cases with reminder' },  
 { \_id: 13, reportType: 'Performance report' },  
 ]}  
 columns={[  
 {  
 field: 'taskType',  
 headerName: 'Report',  
 flex: 1,  
 headerAlign: 'left',  
 cellClassName: 'report-cell',  
 valueGetter: (value, row) => {  
 return row.reportType  
 },  
 sortable: false,  
 renderCell: (params) => (  
 <div  
 style={{  
 display: 'flex',  
 justifyContent: 'space-between',  
 width: '100%',  
 alignItems: 'center',  
 }}  
 >  
 <span style={{ fontSize: '16px', color: '#5b6e86' }}>  
 {params.row.reportType}  
 </span>  
 <div  
 style={{ display: 'flex', gap: '10px', marginRight: '50px' }}  
 >  
 <GradientButton  
 sx={{  
 width: 120,  
 color: isTemplateAvailable(params.row.\_id)  
 ? '#FFFFFF !important'  
 : '#000000 !important',  
 opacity: isTemplateAvailable(params.row.\_id) ? 1 : 0.5,  
 }}  
 onClick={() => previewTemplate(params.row.\_id)}  
 disabled={!isTemplateAvailable(params.row.\_id)}  
 >  
 Preview  
 </GradientButton>  
 <GradientButton  
 sx={{  
 width: 120,  
 color: isTemplateAvailable(params.row.\_id)  
 ? '#FFFFFF !important'  
 : '#000000 !important',  
 opacity: isTemplateAvailable(params.row.\_id) ? 1 : 0.5,  
 }}  
 onClick={() => printTemplate(params.row.\_id)}  
 disabled={!isTemplateAvailable(params.row.\_id)}  
 >  
 Print  
 </GradientButton>  
 <GradientButton  
 sx={{  
 width: 120,  
 color: isTemplateAvailable(params.row.\_id)  
 ? '#FFFFFF !important'  
 : '#000000 !important',  
 opacity: isTemplateAvailable(params.row.\_id) ? 1 : 0.5,  
 }}  
 onClick={() => downloadTemplate(params.row.\_id)}  
 disabled={!isTemplateAvailable(params.row.\_id)}  
 >  
 Download  
 </GradientButton>  
 </div>  
 </div>  
 ),  
 },  
 ]}  
 sx={{  
 '& .MuiDataGrid-container--top [role=row]': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 },  
 '& .MuiDataGrid-columnHeader': {  
 backgroundColor: 'rgb(23, 137, 76)',  
 fontSize: '16px',  
 color: '#fff',  
 marginLeft: '20px',  
 },  
 '& .MuiDataGrid-sortIcon': {  
 color: '#fff',  
 display: 'none',  
 },  
 '& .report-cell': {  
 paddingLeft: '30px',  
 },  
 '& .MuiButtonBase-root': {  
 color: '#000',  
 },  
 '& .MuiDataGrid-footerContainer': {  
 display: 'none',  
 },  
 }}  
 />  
 </Box>  
 )  
}  
export default Report

## File: bd-scs-backend-web/src/App.jsx

import { useContext, useEffect, useState } from 'react'  
import { Route, Routes } from 'react-router-dom'  
import styled from '@mui/material/styles/styled'  
import { createTheme, ThemeProvider } from '@mui/material/styles'  
import { QueryClient, QueryClientProvider } from '@tanstack/react-query'  
import Menu from 'components/Menu'  
import Snackbar from 'components/basic/Snackbar'  
import CircularProgress from '@mui/material/CircularProgress'  
import Box from '@mui/material/Box'  
import AppRoutes from './routes'  
import Login from 'routes/login'  
import { fetchProfile } from 'apis/user'  
import UserProvider, { userContext } from 'src/context/UserProvider'  
import SnackbarProvider from 'src/context/SnackbarProvider'  
import colors from 'src/constants/colors'  
import 'src/i18n'  
const theme = createTheme({  
 palette: {  
 primary: {  
 main: colors.main,  
 },  
 secondary: {  
 main: colors.secondary,  
 },  
 text: {  
 primary: '#000',  
 secondary: '#2E4765',  
 disabled: '#9e9e9e',  
 },  
 },  
 typography: {  
 title: {  
 color: '#000',  
 fontSize: '18px',  
 },  
 titleWithGreyBg: {  
 color: '#2E4765',  
 fontSize: '24px',  
 backgroundColor: '#F5F5F5',  
 fontWeight: 'bold',  
 padding: 8,  
 },  
 subtitle: {  
 color: '#616161',  
 fontSize: '14px',  
 },  
 title2: {  
 color: '#000',  
 fontSize: '24px',  
 fontWeight: 'bold',  
 },  
 title3: {  
 fontSize: '14px',  
 fontWeight: 500,  
 },  
 large: {  
 fontSize: '18px',  
 },  
 medium: {  
 fontSize: '14px',  
 },  
 small: {  
 fontSize: '12px',  
 },  
 warning: {  
 textDecoration: 'underline',  
 color: colors.red,  
 },  
 link: {  
 textDecoration: 'underline',  
 color: 'rgb(59 130 246)',  
 cursor: 'pointer',  
 },  
 },  
})  
const AppContainer = styled('div')({  
 display: 'flex',  
 width: '100vw',  
})  
const PageContainer = styled('div')({  
 height: '100vh',  
 flexDirection: 'column',  
 width: `calc(100% - 100px)`,  
 backgroundColor: `#ffffff`,  
 overflow: 'hidden',  
})  
function Main() {  
 const [user, setUser] = useContext(userContext);  
 const [isLoading, setIsLoading] = useState(true);  
 useEffect(() => {  
 const accessToken = localStorage.getItem('accessToken');  
 if (accessToken) {  
 fetchProfile().then((res) => {  
 setIsLoading(false);  
 if (res) setUser(res);  
 });  
 } else {  
 setIsLoading(false);  
 }  
 }, [])  
 if (isLoading) {  
 return (  
 <Box  
 display="flex"  
 flexDirection="column"  
 alignItems="center"  
 justifyContent="center"  
 position="absolute"  
 height="100%"  
 width="100%"  
 >  
 <CircularProgress />  
 </Box>  
 )  
 }  
 if (!user) {  
 return (  
 <Routes>  
 <Route path="\*" element={<Login />} />  
 </Routes>  
 )  
 } else {  
 return (  
 <AppContainer>  
 <Menu />  
 <PageContainer>  
 <AppRoutes />  
 </PageContainer>  
 </AppContainer>  
 )  
 }  
}  
const queryClient = new QueryClient()  
function App() {  
 return (  
 <ThemeProvider theme={theme}>  
 <QueryClientProvider client={queryClient}>  
 <UserProvider>  
 <SnackbarProvider>  
 <Main />  
 <Snackbar />  
 </SnackbarProvider>  
 </UserProvider>  
 </QueryClientProvider>  
 </ThemeProvider>  
 )  
}  
export default App

## File: bd-scs-backend-web/package.json

{  
 "name": "bs-app",  
 "version": "0.1.0",  
 "private": true,  
 "dependencies": {  
 "@emotion/css": "^11.13.0",  
 "@emotion/react": "^11.13.0",  
 "@emotion/styled": "^11.13.0",  
 "@hookform/resolvers": "^3.9.0",  
 "@mui/icons-material": "^5.16.4",  
 "@mui/lab": "6.0.0-beta.30",  
 "@mui/material": "^5.16.4",  
 "@mui/x-data-grid": "^7.11.1",  
 "@mui/x-date-pickers": "^7.11.0",  
 "@tanstack/react-query": "^5.51.15",  
 "@testing-library/jest-dom": "^5.14.1",  
 "@testing-library/react": "^13.0.0",  
 "@testing-library/user-event": "^13.2.1",  
 "@tinymce/tinymce-react": "^5.1.1",  
 "axios": "^1.7.2",  
 "dayjs": "^1.11.12",  
 "i18next": "^23.12.2",  
 "lodash": "^4.17.21",  
 "prop-types": "^15.8.1",  
 "react": "^18.3.1",  
 "react-dom": "^18.3.1",  
 "react-dropzone": "^14.2.3",  
 "react-hook-form": "^7.52.1",  
 "react-i18next": "^15.0.0",  
 "react-router-dom": "^6.25.1",  
 "react-scripts": "5.0.1",  
 "tinymce": "^7.3.0",  
 "uuid": "^11.1.0",  
 "web-vitals": "^2.1.0",  
 "xlsx-js-style": "^1.2.0",  
 "yup": "^1.4.0"  
 },  
 "scripts": {  
 "dev": "vite",  
 "start": "react-scripts start",  
 "build": "vite build",  
 "test": "react-scripts test",  
 "eject": "react-scripts eject"  
 },  
 "eslintConfig": {  
 "extends": [  
 "react-app",  
 "react-app/jest"  
 ]  
 },  
 "browserslist": {  
 "production": [  
 ">0.2%",  
 "not dead",  
 "not op\_mini all"  
 ],  
 "development": [  
 "last 1 chrome version",  
 "last 1 firefox version",  
 "last 1 safari version"  
 ]  
 },  
 "devDependencies": {  
 "@vitejs/plugin-react-swc": "^3.7.0",  
 "vite": "^5.3.4"  
 }  
}

## File: bd-scs-backend-web/README.md

# Getting Started with Create React App  
  
This project was bootstrapped with [Create React App](https://github.com/facebook/create-react-app).  
  
## Available Scripts  
  
In the project directory, you can run:  
  
### `yarn start`  
  
Runs the app in the development mode.\  
Open [http://localhost:3000](http://localhost:3000) to view it in your browser.  
  
The page will reload when you make changes.\  
You may also see any lint errors in the console.  
  
### `yarn test`  
  
Launches the test runner in the interactive watch mode.\  
See the section about [running tests](https://facebook.github.io/create-react-app/docs/running-tests) for more information.  
  
### `yarn build`  
  
Builds the app for production to the `build` folder.\  
It correctly bundles React in production mode and optimizes the build for the best performance.  
  
The build is minified and the filenames include the hashes.\  
Your app is ready to be deployed!  
  
See the section about [deployment](https://facebook.github.io/create-react-app/docs/deployment) for more information.  
  
### `yarn eject`  
  
\*\*Note: this is a one-way operation. Once you `eject`, you can't go back!\*\*  
  
If you aren't satisfied with the build tool and configuration choices, you can `eject` at any time. This command will remove the single build dependency from your project.  
  
Instead, it will copy all the configuration files and the transitive dependencies (webpack, Babel, ESLint, etc) right into your project so you have full control over them. All of the commands except `eject` will still work, but they will point to the copied scripts so you can tweak them. At this point you're on your own.  
  
You don't have to ever use `eject`. The curated feature set is suitable for small and middle deployments, and you shouldn't feel obligated to use this feature. However we understand that this tool wouldn't be useful if you couldn't customize it when you are ready for it.  
  
## Learn More  
  
You can learn more in the [Create React App documentation](https://facebook.github.io/create-react-app/docs/getting-started).  
  
To learn React, check out the [React documentation](https://reactjs.org/).  
  
### Code Splitting  
  
This section has moved here: [https://facebook.github.io/create-react-app/docs/code-splitting](https://facebook.github.io/create-react-app/docs/code-splitting)  
  
### Analyzing the Bundle Size  
  
This section has moved here: [https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size](https://facebook.github.io/create-react-app/docs/analyzing-the-bundle-size)  
  
### Making a Progressive Web App  
  
This section has moved here: [https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app](https://facebook.github.io/create-react-app/docs/making-a-progressive-web-app)  
  
### Advanced Configuration  
  
This section has moved here: [https://facebook.github.io/create-react-app/docs/advanced-configuration](https://facebook.github.io/create-react-app/docs/advanced-configuration)  
  
### Deployment  
  
This section has moved here: [https://facebook.github.io/create-react-app/docs/deployment](https://facebook.github.io/create-react-app/docs/deployment)  
  
### `yarn build` fails to minify  
  
This section has moved here: [https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify](https://facebook.github.io/create-react-app/docs/troubleshooting#npm-run-build-fails-to-minify)

## File: bd-scs-nodejs-frontend/src/config/database.js

const { Sequelize } = require("sequelize");  
require("dotenv").config();  
console.log("DB\_NAME", process.env.DB\_NAME, typeof process.env.DB\_NAME);  
console.log("DB\_USER", process.env.DB\_USER, typeof process.env.DB\_USER);  
console.log("DB\_PASSWORD", process.env.DB\_PASSWORD, typeof process.env.DB\_PASSWORD);  
console.log("DB\_HOST", process.env.DB\_HOST, typeof process.env.DB\_HOST);  
console.log("DB\_DIALECT", process.env.DB\_DIALECT, typeof process.env.DB\_DIALECT);  
const sequelize = new Sequelize(  
 process.env.DB\_NAME,  
 process.env.DB\_USER,  
 process.env.DB\_PASSWORD,  
 {  
 host: process.env.DB\_HOST,  
 dialect: process.env.DB\_DIALECT,  
 dialectOptions: {  
 options: {  
 requestTimeout: 300000,  
 trustServerCertificate: true  
 },  
 },  
 logging: false,  
 }  
);  
module.exports = sequelize;

## File: bd-scs-nodejs-frontend/src/models/AdrBlk\_T.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class AdrBlkModel extends Model {}  
AdrBlkModel.init(  
 {  
 ADR\_BLK\_ID: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 BLK\_TYPE\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 BLDG\_CAT\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 BLDG\_USAGE\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 SYS\_REGION\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 SYS\_DISTRICT\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 AREA\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 BLK\_DESC\_E\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 BLK\_NO\_NUM: {  
 type: DataTypes.INTEGER,  
 allowNull: true,  
 },  
 BLK\_NO\_ALPHA: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_E3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_C1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_C2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_C3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E5: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_ST\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_ST\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_ST\_C: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C5: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OBSOLETE: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "AdrBlkModel\_T",  
 tableName: "AdrBlk\_T",  
 timestamps: true,  
 }  
);  
module.exports = AdrBlkModel;

## File: bd-scs-nodejs-frontend/src/models/AdrBlk.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class AdrBlkModel extends Model {}  
AdrBlkModel.init(  
 {  
 ADR\_BLK\_ID: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 BLK\_TYPE\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: false,  
 },  
 BLDG\_CAT\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: false,  
 },  
 BLDG\_USAGE\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: false,  
 },  
 SYS\_REGION\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: false,  
 },  
 SYS\_DISTRICT\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: false,  
 },  
 AREA\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: false,  
 },  
 BLK\_DESC\_E\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: false,  
 },  
 BLK\_NO\_NUM: {  
 type: DataTypes.INTEGER,  
 allowNull: false,  
 },  
 BLK\_NO\_ALPHA: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_E3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_C1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_C2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 BLDG\_NAME\_C3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_E5: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_ST\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_ST\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_E4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_ST\_C: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_LOT\_C4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C3: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C4: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OSADR\_C5: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OBSOLETE: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "AdrBlkModel",  
 tableName: "AdrBlk",  
 timestamps: true,  
 }  
);  
module.exports = AdrBlkModel;

## File: bd-scs-nodejs-frontend/src/models/ApplicationCase.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class ApplicationCaseModel extends Model {}  
ApplicationCaseModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 FromId: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 ApplicationNo: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FileDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 CreatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 UpdatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "ApplicationCaseModel",  
 tableName: "ApplicationCase",  
 timestamps: true,  
 }  
);  
module.exports = ApplicationCaseModel;

## File: bd-scs-nodejs-frontend/src/models/ApplicationFile.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
const IamSmartModel = require("./IamSmart")  
class ApplicationFileModel extends Model {  
}  
ApplicationFileModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 FromId: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 ApplicationNo: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 SubmissionId: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FileId: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FileName: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 unique: true,  
 },  
 FileSize: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 DocumentType: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 SignDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 SignType: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FilePath: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FileDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 RemoveStatus: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CreatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 UpdatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "ApplicationFileModel",  
 tableName: "ApplicationFiles",  
 timestamps: true,  
 }  
);  
module.exports = ApplicationFileModel;

## File: bd-scs-nodejs-frontend/src/models/ApRse.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
const { stubFalse } = require("lodash");  
class ApRseModel extends Model {}  
ApRseModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 autoIncrement: true,  
 primaryKey: true,  
 allowNull: false,  
 },  
 UUID: {  
 type: DataTypes.STRING(100),  
 allowNull: true,  
 },  
 Name: {  
 type: DataTypes.STRING(200),  
 allowNull: true,  
 },  
 Name\_tc: {  
 type: DataTypes.STRING(200),  
 allowNull: true,  
 },  
 RegistrationNumber: {  
 type: DataTypes.STRING(100),  
 allowNull: true,  
 },  
 RegistrationType: {  
 type: DataTypes.STRING(100),  
 allowNull: true,  
 },  
 ExpiryDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 CreatedAt: {  
 type: DataTypes.DATE,  
 allowNull: false,  
 defaultValue: DataTypes.NOW,  
 },  
 CreatedBy: {  
 type: DataTypes.STRING(100),  
 allowNull: true,  
 },  
 UpdatedAt: {  
 type: DataTypes.DATE,  
 allowNull: false,  
 defaultValue: DataTypes.NOW,  
 },  
 UpdatedBy: {  
 type: DataTypes.STRING(100),  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "ApRseModel",  
 tableName: "ApRse",  
 timestamps: false,  
 }  
);  
module.exports = ApRseModel;

## File: bd-scs-nodejs-frontend/src/models/Attachment.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class AttachmentModel extends Model {}  
AttachmentModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 SubmissionId: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 ApplicationNo: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 DocumentType: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 RemoveStatus: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FromId: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FileId: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 FileName: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 FilePath: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 FileDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 FileSize: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 signDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 signType: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CreatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 UpdatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "AttachmentModel",  
 tableName: "Attachment",  
 timestamps: true,  
 }  
);  
module.exports = AttachmentModel;

## File: bd-scs-nodejs-frontend/src/models/BackendUpdate.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class BackendUpdateModel extends Model {}  
BackendUpdateModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 ApplicationNo: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 UpdateType: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Status: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FileSize: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 DocumentType: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FilePath: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FileDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 FileMimeType: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 File: {  
 type: DataTypes.TEXT,  
 allowNull: true,  
 },  
 UpdatedAt: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 defaultValue: DataTypes.NOW,  
 },  
 },  
 {  
 sequelize,  
 modelName: "BackendUpdate",  
 tableName: "BackendUpdate",  
 timestamps: false,  
 }  
);  
module.exports = BackendUpdateModel;

## File: bd-scs-nodejs-frontend/src/models/GenOtp.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class GenOtpModel extends Model {}  
GenOtpModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 ApplicationNo: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 UserId: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 Otp: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Used: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 EffectiveUntil: {  
 type: DataTypes.DATE,  
 allowNull: false,  
 },  
 CreatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 UpdatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "GenOtpModel",  
 tableName: "GenOtp",  
 timestamps: true,  
 }  
);  
module.exports = GenOtpModel;

## File: bd-scs-nodejs-frontend/src/models/IamSmart.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
const ApplicationFileModel = require("./ApplicationFile")  
class IamSmartModel extends Model {  
}  
IamSmartModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 BusinessId: {  
 type: DataTypes.STRING,  
 },  
 FileId: {  
 type: DataTypes.STRING,  
 },  
 CurrentUrl: {  
 type: DataTypes.STRING  
 }  
 },  
 {  
 sequelize,  
 modelName: "IamSmartModel",  
 tableName: "IamSmart",  
 timestamps: true,  
 }  
);  
module.exports = IamSmartModel;

## File: bd-scs-nodejs-frontend/src/models/LogEvents.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class LogEventsModel extends Model {}  
LogEventsModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 LogAlias: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 IpAddress: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 PlatformId: {  
 type: DataTypes.INTEGER,  
 allowNull: false,  
 },  
 FromUrl: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CallUrlApi: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 StatusCode: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 LoginInfo: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 RequestBody: {  
 type: DataTypes.TEXT,  
 allowNull: true,  
 },  
 Response: {  
 type: DataTypes.TEXT,  
 allowNull: true,  
 },  
 PrgError: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CreatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "LogEventsModel",  
 tableName: "LogEvents",  
 timestamps: true,  
 }  
);  
module.exports = LogEventsModel;

## File: bd-scs-nodejs-frontend/src/models/SchoolAppInfo.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class SchoolAppInfoModel extends Model {}  
SchoolAppInfoModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 FromId: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 ApplicationNo: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FormName: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicationType: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantAddress: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantName: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantEmail: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantMobile: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPName: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPEmail: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPMobile: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPTel: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPFax: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Date: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 BlockID: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseCN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseCNFloor: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseCNFloorUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseCNUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseCNUnitUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseENFloor: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseENFloorUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseENUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseENUnitUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 NameOfSchoolCN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 NameOfSchoolEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 DescriptionOfSchool: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OtherSchoolDesc: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AgeOfStudent: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EstimatedNoOfStudent: {  
 type: DataTypes.INTEGER,  
 allowNull: true,  
 },  
 Sign: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ChineseNameOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EnglishNameOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EmailOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 MobileOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 RegTypeOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CertRegNoOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ChineseNameOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EnglishNameOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EmailOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 MobileOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 RegTypeOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CertRegNoOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ChineseNameOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EnglishNameOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EmailOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 MobileOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 RegTypeOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CertRegNoOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Status: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Stage: {  
 type: DataTypes.INTEGER,  
 allowNull: true,  
 defaultValue: 1,  
 },  
 SubmittedDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 InspectReq: {  
 type: DataTypes.INTEGER,  
 allowNull: true,  
 },  
 WithdrawReq: {  
 type: DataTypes.INTEGER,  
 allowNull: true,  
 },  
 BDLetterRefDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 BDLetterRefNo: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 SubmitRefDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 ApplicantPostTitle: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantAddressEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantTitle: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantNameEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantHKIC: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantTel: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantRepOrg: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantRepOrgEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPAddress: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPTitle: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 PhoneOfSchool: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantRefDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 FileDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 DateOfInspection: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 ApplicantDated: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 Category1Requirement: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Category2Requirement: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 RegExpiryDateOfAp2: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 FaxOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 IdentityOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApList: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 SelfCertification: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CreatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 UpdatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "SchoolAppInfoModel",  
 tableName: "SchoolApp\_Infos",  
 timestamps: true,  
 }  
);  
module.exports = SchoolAppInfoModel;

## File: bd-scs-nodejs-frontend/src/models/SchoolAppSubmission.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class SchoolAppSubmissionModel extends Model {}  
SchoolAppSubmissionModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 FromId: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 ApplicationNo: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 SubmissionId: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 FormName: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicationType: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantAddress: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantName: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantEmail: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantMobile: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPName: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPEmail: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPMobile: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPTel: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPFax: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Date: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 BlockID: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseCN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseCNFloor: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseCNFloorUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseCNUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseCNUnitUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseENFloor: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseENFloorUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseENUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfPremiseENUnitUnit: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 NameOfSchoolCN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 NameOfSchoolEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 DescriptionOfSchool: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 OtherSchoolDesc: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AgeOfStudent: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EstimatedNoOfStudent: {  
 type: DataTypes.INTEGER,  
 allowNull: true,  
 },  
 Sign: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ChineseNameOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EnglishNameOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EmailOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 MobileOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 RegTypeOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CertRegNoOfAp1: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ChineseNameOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EnglishNameOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EmailOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 MobileOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 RegTypeOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CertRegNoOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ChineseNameOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EnglishNameOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EmailOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 MobileOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 RegTypeOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CertRegNoOfRse: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Status: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 SubmittedDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 InspectReq: {  
 type: DataTypes.INTEGER,  
 allowNull: true,  
 },  
 WithdrawReq: {  
 type: DataTypes.INTEGER,  
 allowNull: true,  
 },  
 BDLetterRefDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 BDLetterRefNo: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 SubmitRefDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 ApplicantPostTitle: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantAddressEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantTitle: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantNameEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantHKIC: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantTel: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantRepOrg: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantRepOrgEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPAddress: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ContactPTitle: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 PhoneOfSchool: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApplicantRefDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 FileDate: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 RemoveStatus: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 DateOfInspection: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 ApplicantDated: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 Category1Requirement: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Category2Requirement: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 RegExpiryDateOfAp2: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 FaxOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 AddressOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 IdentityOfAp2: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ApList: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 SelfCertification: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CreatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 UpdatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Synced: {  
 type: DataTypes.BOOLEAN,  
 defaultValue: false,  
 },  
 },  
 {  
 sequelize,  
 modelName: "SchoolAppSubmissionModel",  
 tableName: "SchoolApp\_Submissions",  
 timestamps: true,  
 }  
);  
module.exports = SchoolAppSubmissionModel;

## File: bd-scs-nodejs-frontend/src/models/ScsMasterTable.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class ScsMasterTableModel extends Model {}  
ScsMasterTableModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 Type: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 Code: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 Ordering: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 DataValue: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CaptionEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CaptionTC: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CaptionSC: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Remarks: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 LongTextEN: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 LongTextTC: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 LongTextSC: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "ScsMasterTableModel",  
 tableName: "ScsMasterTable",  
 timestamps: true,  
 }  
);  
module.exports = ScsMasterTableModel;

## File: bd-scs-nodejs-frontend/src/models/Staff.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class StaffModel extends Model {}  
StaffModel.init(  
 {  
 Id: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 UserId: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Password: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 EnglishName: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ChineseName: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Department: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Post: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Telephone: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Email: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Role: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 Status: {  
 type: DataTypes.INTEGER,  
 allowNull: true,  
 },  
 CreatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 UpdatedBy: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "StaffModel",  
 tableName: "Staff",  
 timestamps: true,  
 }  
);  
module.exports = StaffModel;

## File: bd-scs-nodejs-frontend/src/models/Sys\_Meta\_Data\_T.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class SysMetaDataModel extends Model {}  
SysMetaDataModel.init(  
 {  
 SYS\_META\_DATA\_ID: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 REC\_TYPE: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 CODE: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 PARENT\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 ENGLISH\_DESCRIPTION: {  
 type: DataTypes.STRING(2500),  
 allowNull: true,  
 },  
 CHINESE\_DESCRIPTION: {  
 type: DataTypes.STRING(2500),  
 allowNull: true,  
 },  
 IS\_ACTIVE: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ORDERING: {  
 type: DataTypes.BIGINT,  
 allowNull: true,  
 },  
 CREATED\_DT: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 CREATED\_POST: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CREATED\_NAME: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CREATED\_SECTION: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 LAST\_MODIFIED\_DT: {  
 type: DataTypes.DATE,  
 allowNull: false,  
 },  
 LAST\_MODIFIED\_POST: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 LAST\_MODIFIED\_NAME: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 LAST\_MODIFIED\_SECTION: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 MULTI\_INSP\_FOR\_LSO\_IND: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 REC\_SRC: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "SysMetaDataT",  
 tableName: "Sys\_Meta\_Data\_T",  
 timestamps: false,  
 }  
);  
module.exports = SysMetaDataModel;

## File: bd-scs-nodejs-frontend/src/models/Sys\_Meta\_Data.js

const { DataTypes, Model } = require("sequelize");  
const sequelize = require("../config/database");  
class SysMetaDataModel extends Model {}  
SysMetaDataModel.init(  
 {  
 SYS\_META\_DATA\_ID: {  
 type: DataTypes.BIGINT,  
 primaryKey: true,  
 autoIncrement: true,  
 },  
 REC\_TYPE: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 CODE: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 PARENT\_ID: {  
 type: DataTypes.BIGINT,  
 allowNull: false,  
 },  
 ENGLISH\_DESCRIPTION: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CHINESE\_DESCRIPTION: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 IS\_ACTIVE: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 ORDERING: {  
 type: DataTypes.BIGINT,  
 allowNull: false,  
 },  
 CREATED\_DT: {  
 type: DataTypes.DATE,  
 allowNull: true,  
 },  
 CREATED\_POST: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CREATED\_NAME: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 CREATED\_SECTION: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 LAST\_MODIFIED\_DT: {  
 type: DataTypes.DATE,  
 allowNull: false,  
 },  
 LAST\_MODIFIED\_POST: {  
 type: DataTypes.STRING,  
 allowNull: false,  
 },  
 LAST\_MODIFIED\_NAME: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 LAST\_MODIFIED\_SECTION: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 MULTI\_INSP\_FOR\_LSO\_IND: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 REC\_SRC: {  
 type: DataTypes.STRING,  
 allowNull: true,  
 },  
 },  
 {  
 sequelize,  
 modelName: "SysMetaData",  
 tableName: "Sys\_Meta\_Data",  
 timestamps: false,  
 }  
);  
module.exports = SysMetaDataModel;

## File: bd-scs-nodejs-frontend/src/routes/ApplicationController.js

var express = require("express");  
var router = express.Router();  
const SchoolAppSubmissionModel = require("../models/SchoolAppSubmission");  
const SchoolAppInfoModel = require("../models/SchoolAppInfo");  
const ScsMasterTableModel = require("../models/ScsMasterTable");  
const BackendUpdateModel = require("../models/BackendUpdate");  
const AttachmentModel = require("../models/Attachment");  
const ApplicationFileModel = require("../models/ApplicationFile");  
const ApRseModel = require("../models/ApRse");  
const AdrBlkModel = require("../models/AdrBlk");  
const SysMetaDataModel = require("../models/Sys\_Meta\_Data");  
const sequelize = require("../config/database");  
const { generateApplicationNo } = require("../utils/applicationUtils");  
const { getUserJsonWithToken } = require("../utils/loginUtils");  
const { v4: uuidv4 } = require("uuid");  
const multer = require("multer");  
const path = require("path");  
const fs = require("fs");  
const zip = require("node-zip");  
const { QueryTypes } = require("sequelize");  
const { camelize } = require("../utils/on9Dotnet");  
const sendEmail = require("../utils/sendEmail");  
const storage = multer.diskStorage({  
 destination: (req, file, cb) => {  
 cb(null, "wwwroot/uploads");  
 },  
 filename: (req, file, cb) => {  
 const uniqueSuffix = uuidv4();  
 const fileExtension = path.extname(file.originalname);  
 cb(null, `${uniqueSuffix}${fileExtension}`);  
 },  
});  
const upload = multer({ storage });  
const capitalizeKeys = (obj) => {  
 const capitalizedObj = {};  
 for (let key in obj) {  
 if (obj.hasOwnProperty(key)) {  
 let capitalizedKey = "";  
 if (key.startsWith("bd")) {  
 capitalizedKey =  
 key.charAt(0).toUpperCase() +  
 key.charAt(1).toUpperCase() +  
 key.slice(2);  
 } else {  
 capitalizedKey = key.charAt(0).toUpperCase() + key.slice(1);  
 }  
 if (typeof obj[key] === "object" && !Array.isArray(obj[key])) {  
 capitalizedObj[capitalizedKey] = capitalizeKeys(obj[key]);  
 } else {  
 capitalizedObj[capitalizedKey] = obj[key];  
 }  
 }  
 }  
 return capitalizedObj;  
};  
const lowercaseKeys = (obj) => {  
 const lowercasedObj = {};  
 for (let key in obj) {  
 if (obj.hasOwnProperty(key)) {  
 let lowercasedKey = "";  
 if (key.startsWith("BD")) {  
 lowercasedKey =  
 key.charAt(0).toLowerCase() +  
 key.charAt(1).toLowerCase() +  
 key.slice(2);  
 } else {  
 lowercasedKey = key.charAt(0).toLowerCase() + key.slice(1);  
 }  
 if (obj[key] instanceof Date) {  
 lowercasedObj[lowercasedKey] = obj[key];  
 } else if (  
 obj[key] !== null &&  
 typeof obj[key] === "object" &&  
 !Array.isArray(obj[key])  
 ) {  
 lowercasedObj[lowercasedKey] = lowercaseKeys(obj[key]);  
 } else {  
 lowercasedObj[lowercasedKey] = obj[key];  
 }  
 }  
 }  
 return lowercasedObj;  
};  
router.post("/newschoolsubmission", async (req, res, next) => {  
 try {  
 let applicationReq = req.body;  
 applicationReq = capitalizeKeys(applicationReq);  
 if (applicationReq.ApplicationNo === "") {  
 // new application create new application no  
 const newApplicationNo = await generateApplicationNo(  
 applicationReq.Form.ApplicationType  
 );  
 applicationReq.Form.ApplicationNo = newApplicationNo;  
 } else {  
 applicationReq.Form.ApplicationNo = applicationReq.ApplicationNo;  
 }  
 if (applicationReq.Form.ApList) {  
 applicationReq.Form.ApList = JSON.stringify(applicationReq.Form.ApList);  
 }  
 if (applicationReq.Form.SelfCertification) {  
 applicationReq.Form.SelfCertification = JSON.stringify(  
 applicationReq.Form.SelfCertification  
 );  
 }  
 if (  
 applicationReq.Form.FormName === "inspect-req" ||  
 applicationReq.Form.FormName === "update-info" ||  
 applicationReq.Form.FormName === "appoint-ap" ||  
 applicationReq.Form.FormName === "withdraw-app"  
 ) {  
 applicationReq.Form.Date = new Date();  
 applicationReq.Form.SubmittedDate = new Date();  
 }  
 if (applicationReq.Form.FormName === "update-info") {  
 if (!applicationReq.Form.ApplicantNameEN) {  
 applicationReq.Form.ApplicantNameEN = applicationReq.Form.ApplicantName;  
 }  
 }  
 const newSubmssionId = uuidv4();  
 applicationReq.Form.SubmissionId = newSubmssionId;  
 if (  
 applicationReq.Form.FormName === "A2" ||  
 applicationReq.Form.FormName === "E2" ||  
 applicationReq.Form.FormName === "CCCAI-C2" ||  
 applicationReq.Form.FormName === "EDB-JOKC02" ||  
 applicationReq.Form.FormName === "CCCAI-MH3"  
 ) {  
 if (!applicationReq.Form.ApplicantNameEN) {  
 applicationReq.Form.ApplicantNameEN = applicationReq.Form.ApplicantName;  
 }  
 const applicationInfo = await SchoolAppInfoModel.create(  
 applicationReq.Form  
 );  
 if (!applicationInfo) {  
 return res.status(400).json({ error: "Create application fail" });  
 }  
 }  
 const getUser = await getUserJsonWithToken({  
 applicationNo: applicationReq.Form.ApplicationNo,  
 userId: applicationReq.Form.UpdatedBy,  
 });  
 applicationReq.Form.UpdatedBy = getUser.role;  
 await SchoolAppInfoModel.update(applicationReq.Form, {  
 where: { ApplicationNo: applicationReq.Form.ApplicationNo },  
 });  
 const applicationResponse = await SchoolAppSubmissionModel.create(  
 applicationReq.Form  
 );  
 if (!applicationResponse) {  
 return res.status(400).json({ error: "Create application fail" });  
 }  
 if (applicationReq.Form.FormName === "withdraw-app") {  
 await SchoolAppInfoModel.update(  
 { Status: "Abandoned" },  
 { where: { ApplicationNo: applicationReq.Form.ApplicationNo } }  
 );  
 }  
 let successResponse = {};  
 if (  
 applicationReq.Form.FormName === "A2" ||  
 applicationReq.Form.FormName === "E2" ||  
 applicationReq.Form.FormName === "CCCAI-C2" ||  
 applicationReq.Form.FormName === "EDB-JOKC02" ||  
 applicationReq.Form.FormName === "CCCAI-MH3"  
 ) {  
 successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 applicationNo: applicationReq.Form.ApplicationNo,  
 submissionId: newSubmssionId,  
 msg: "Saved Successfully",  
 userId: getUser.userId,  
 role: getUser.role,  
 token: getUser.token,  
 },  
 };  
 } else {  
 successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 applicationNo: applicationReq.Form.ApplicationNo,  
 submissionId: newSubmssionId,  
 msg: "Saved Successfully",  
 },  
 };  
 }  
 res.status(200).json(successResponse);  
 } catch (err) {  
 next(err);  
 }  
});  
router.post("/updateschoolsubmission", async (req, res, next) => {  
 try {  
 let applicationReq = req.body;  
 applicationReq = capitalizeKeys(applicationReq);  
 if (applicationReq.Form.Status === "Submitted") {  
 applicationReq.Form.SubmittedDate = new Date();  
 const getUser = await getUserJsonWithToken({  
 applicationNo: applicationReq.ApplicationNo,  
 userId: applicationReq.Form.UpdatedBy,  
 });  
 applicationReq.Form.UpdatedBy = getUser.role;  
 }  
 if (applicationReq.Form.ApList) {  
 applicationReq.Form.ApList = JSON.stringify(applicationReq.Form.ApList);  
 }  
 if (applicationReq.Form.SelfCertification) {  
 applicationReq.Form.SelfCertification = JSON.stringify(  
 applicationReq.Form.SelfCertification  
 );  
 }  
 const appInfo = await SchoolAppInfoModel.update(applicationReq.Form, {  
 where: { ApplicationNo: applicationReq.ApplicationNo },  
 returning: true,  
 plain: true,  
 });  
 const applicationResponse = await SchoolAppSubmissionModel.update(  
 applicationReq.Form,  
 {  
 where: {  
 ApplicationNo: applicationReq.ApplicationNo,  
 SubmissionId: applicationReq.Form.SubmissionId,  
 },  
 }  
 );  
 if (!applicationResponse) {  
 return res.status(400).json({ error: "Application update failed" });  
 }  
 const updatedApplication = await SchoolAppSubmissionModel.findOne({  
 where: { SubmissionId: applicationReq.Form.SubmissionId },  
 });  
 console.log("App info", appInfo);  
 if (!updatedApplication) {  
 return res  
 .status(400)  
 .json({ error: "Application not found, update failed" });  
 }  
 if (applicationReq.Form.Status === "Submitted") {  
 await sendEmail(  
 appInfo?.[1]?.dataValues?.ApplicantEmail,  
 `Application received: ${appInfo?.[1]?.dataValues?.ApplicationNo}`,  
 `  
 Dear Sir/Madam,  
 Your application number is ${appInfo?.[1]?.dataValues?.ApplicationNo}. Please use this application number to login the Licensing Self-Certification Portal in the future.  
 For enquiry, please contact us at telephone no. 2626 1616 (Handled by “1823”) or via email lscphelpdesk@bd.gov.hk.  
 Please do not reply to this email as it comes from an automated mailbox.  
 Buildings Department  
 `  
 );  
 await sendEmail(  
 process.env.HELPDESK\_EMAIL,  
 `New application received: ${appInfo?.[1]?.dataValues?.ApplicationNo}`,  
 `  
 Dear Helpdesk,  
 A new application has been received with the application number ${appInfo?.[1]?.dataValues?.ApplicationNo}.  
 Please follow up with the applicant accordingly.  
 Regards,  
 BD E-Submission System  
 `  
 );  
 }  
 const successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 applicationNo: applicationReq.ApplicationNo,  
 submissionId: updatedApplication.dataValues.SubmissionId,  
 msg: "Saved Successfully",  
 userId: null,  
 role: null,  
 token: null,  
 },  
 };  
 res.status(200).json(successResponse);  
 } catch (err) {  
 next(err);  
 }  
});  
router.get("/getapplicationcasealldata", async (req, res, next) => {  
 try {  
 const applicationNo = req.query.applicationNo.trim();  
 const applicationResponse = await SchoolAppSubmissionModel.findAll({  
 where: { ApplicationNo: applicationNo },  
 raw: true,  
 });  
 const applicationInfo = await SchoolAppInfoModel.findOne({  
 where: { ApplicationNo: applicationNo },  
 });  
 const lowerCasedApplication = applicationResponse.map((application) => {  
 const lowerCasedDataValues = {};  
 Object.keys(application).forEach((key) => {  
 lowerCasedDataValues[key.charAt(0).toLowerCase() + key.slice(1)] =  
 application[key];  
 });  
 return lowerCasedDataValues;  
 });  
 const backendUpdates = await BackendUpdateModel.findAll({  
 where: { ApplicationNo: applicationNo },  
 });  
 let appInfoData = lowercaseKeys(applicationInfo?.dataValues);  
 let appSubmissionData = lowerCasedApplication;  
 try {  
 if (appInfoData.apList)  
 appInfoData.apList = JSON.parse(appInfoData.apList);  
 if (appInfoData.selfCertification)  
 appInfoData.selfCertification = JSON.parse(  
 appInfoData.selfCertification  
 );  
 if (appSubmissionData.apList)  
 appSubmissionData.apList = JSON.parse(appSubmissionData.apList);  
 if (appSubmissionData.selfCertification)  
 appSubmissionData.selfCertification = JSON.parse(  
 appSubmissionData.selfCertification  
 );  
 } catch (e) {}  
 const successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 applicationNo: applicationNo,  
 appSubmissions: appSubmissionData,  
 appInfo: appInfoData,  
 backendUpdates: backendUpdates.map((update) =>  
 lowercaseKeys(update.dataValues)  
 ),  
 },  
 };  
 return res.status(200).json(successResponse);  
 } catch (err) {  
 next(err);  
 }  
});  
router.get("/getapplicationsubmissionandinfo", async (req, res, next) => {  
 try {  
 const submisstionId = req.query.submissionId.trim();  
 const applicationResponse = await SchoolAppSubmissionModel.findOne({  
 where: { SubmissionId: submisstionId },  
 });  
 if (!applicationResponse) {  
 return res.status(400).json({ error: "Application not found!" });  
 }  
 const applicationInfo = await SchoolAppInfoModel.findOne({  
 where: { ApplicationNo: applicationResponse.dataValues.ApplicationNo },  
 });  
 let appInfoData = lowercaseKeys(applicationInfo?.dataValues);  
 let appSubmissionData = lowercaseKeys(applicationResponse?.dataValues);  
 try {  
 if (appInfoData.apList)  
 appInfoData.apList = JSON.parse(appInfoData.apList);  
 if (appInfoData.selfCertification)  
 appInfoData.selfCertification = JSON.parse(  
 appInfoData.selfCertification  
 );  
 if (appSubmissionData.apList)  
 appSubmissionData.apList = JSON.parse(appSubmissionData.apList);  
 if (appSubmissionData.selfCertification)  
 appSubmissionData.selfCertification = JSON.parse(  
 appSubmissionData.selfCertification  
 );  
 } catch (e) {}  
 const successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 applicationNo: applicationResponse.dataValues.ApplicationNo,  
 appInfo: appInfoData,  
 appSubmission: appSubmissionData,  
 },  
 };  
 return res.status(200).json(successResponse);  
 } catch (err) {  
 next(err);  
 }  
});  
router.get("/get-file-list/:submissionId", async (req, res, next) => {  
 try {  
 const { submissionId } = req.params;  
 const applicationFiles = await ApplicationFileModel.findAll({  
 where: { SubmissionId: submissionId },  
 raw: true,  
 });  
 if (!applicationFiles) {  
 return res.status(400).json({ error: "ApplicationFile not found!" });  
 }  
 const lowerCasedAttachments = applicationFiles.map((applicationFile) => {  
 const lowerCasedDataValues = {};  
 Object.keys(applicationFile).forEach((key) => {  
 lowerCasedDataValues[key.charAt(0).toLowerCase() + key.slice(1)] =  
 applicationFile[key];  
 if (key === "FileSize") {  
 lowerCasedDataValues["fileSize"] = parseInt(applicationFile[key]);  
 }  
 });  
 return lowerCasedDataValues;  
 });  
 return res.status(200).json({  
 meta: { code: 200, message: "OK" },  
 data: lowerCasedAttachments,  
 });  
 } catch (err) {  
 console.log(err);  
 next(err);  
 }  
});  
router.get("/get-file/:fileId", async (req, res, next) => {  
 try {  
 const { fileId } = req.params;  
 const applicationFile = await ApplicationFileModel.findOne({  
 where: { FileId: fileId },  
 });  
 if (fs.existsSync(applicationFile.FilePath)) {  
 res.setHeader(  
 "Content-Disposition",  
 'inline; filename="' + path.basename(applicationFile.FilePath) + '"'  
 );  
 res.setHeader("Content-Type", "application/pdf");  
 const fileStream = fs.createReadStream(applicationFile.FilePath);  
 fileStream.pipe(res);  
 } else {  
 res.status(404).json({ error: "File not found" });  
 }  
 } catch (err) {  
 console.log(err);  
 next(err);  
 }  
});  
router.get("/download-update/:backendUpdateId", async (req, res, next) => {  
 try {  
 const { backendUpdateId } = req.params;  
 const backendUpdate = await BackendUpdateModel.findOne({  
 where: { Id: backendUpdateId },  
 });  
 if (backendUpdate.dataValues.File) {  
 let file = JSON.parse(backendUpdate.dataValues.File);  
 if (!fs.existsSync(file.path)) {  
 res.status(404).json({ error: "File not found" });  
 res.status(404).json({ error: "File not found" });  
 }  
 res.setHeader(  
 "Content-Disposition",  
 `attachment; filename=${file.originalname}`  
 );  
 res.setHeader("Content-Type", backendUpdate.dataValues.FileMimeType);  
 const fileStream = fs.createReadStream(file.path);  
 fileStream.pipe(res);  
 } else {  
 res.status(404).json({ error: "File not found" });  
 }  
 } catch (err) {  
 console.log(err);  
 next(err);  
 }  
});  
router.get("/download-file/:submissionId", async (req, res, next) => {  
 try {  
 const { submissionId } = req.params;  
 const attachments = await ApplicationFileModel.findAll({  
 where: { SubmissionId: submissionId },  
 });  
 if (!attachments || attachments.length === 0) {  
 return res.status(404).json({ error: "Attachments not found" });  
 }  
 const newZip = new zip();  
 attachments.forEach((attachment) => {  
 const filePath = attachment.FilePath;  
 const fileName = attachment.FileName;  
 if (fs.existsSync(filePath)) {  
 const pdfContent = fs.readFileSync(filePath);  
 newZip.file(`${fileName}.pdf`, pdfContent);  
 } else {  
 console.log(`File not found: ${filePath}`);  
 }  
 });  
 const data = newZip.generate({ base64: false, compression: "DEFLATE" });  
 const zipFileName = `ApplicationCaseFiles\_${submissionId}.zip`;  
 res.setHeader("Content-Type", "application/zip");  
 res.setHeader("Content-Disposition", `attachment; filename=${zipFileName}`);  
 res.status(200).send(Buffer.from(data, "binary"));  
 } catch (err) {  
 console.log(err);  
 next(err);  
 }  
});  
router.post("/upload-file", upload.single("file"), async (req, res, next) => {  
 try {  
 const {  
 formName,  
 applicationNo,  
 applicationType,  
 submissionId,  
 documentType,  
 signDate,  
 signType,  
 createdBy,  
 } = req.body;  
 const file = req.file;  
 let applicationResponse = {};  
 if (formName === "submit-doc") {  
 try {  
 const submissionId = uuidv4();  
 const application = {  
 ApplicationNo: applicationNo,  
 SubmissionId: submissionId,  
 FormName: formName,  
 ApplicationType: applicationType,  
 Status: "Draft",  
 };  
 applicationResponse = await SchoolAppSubmissionModel.create(  
 application  
 );  
 } catch (error) {  
 console.log(error);  
 return res  
 .status(400)  
 .json({ error: "Fail to create upload submission" });  
 }  
 } else {  
 applicationResponse = await SchoolAppSubmissionModel.findOne({  
 where: { SubmissionId: submissionId },  
 });  
 }  
 if (!applicationResponse) {  
 return res.status(400).json({ error: "Application not found!" });  
 }  
 if (documentType === "form") {  
 const existingAttachment = await ApplicationFileModel.findOne({  
 where: { SubmissionId: submissionId, DocumentType: "form" },  
 });  
 if (existingAttachment) {  
 try {  
 const oldFilePath = `wwwroot/uploads/${existingAttachment.FileName}`;  
 await fs.promises.unlink(oldFilePath).catch(() => {});  
 await ApplicationFileModel.destroy({  
 where: { FileId: existingAttachment.FileId },  
 });  
 } catch (error) {  
 console.error("Update failed:", error);  
 return res.status(500).json({ error: "Failed to update attachment" });  
 }  
 }  
 }  
 const newApplicationFile = await ApplicationFileModel.create({  
 ApplicationNo: applicationResponse.dataValues.ApplicationNo,  
 SubmissionId: applicationResponse.dataValues.SubmissionId,  
 DocumentType: documentType,  
 FileId: file.filename.slice(0, -4),  
 FileName: file.filename,  
 FilePath: file.path,  
 FileSize: file.size,  
 SignDate: signDate,  
 SignType: signType,  
 CreatedBy: createdBy,  
 });  
 const successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 applicationFile: lowercaseKeys(newApplicationFile.dataValues),  
 returnMsg: null,  
 },  
 };  
 res.status(200).json(successResponse);  
 } catch (err) {  
 console.log(err, "error");  
 next(err);  
 }  
});  
router.delete("/delete-file/:fileId", async (req, res, next) => {  
 try {  
 const { fileId } = req.params;  
 const attachment = await ApplicationFileModel.destroy({  
 where: { FileId: fileId },  
 });  
 if (!attachment) {  
 return res.status(404).json({ error: "File not found" });  
 } else {  
 return res  
 .status(200)  
 .json({ meta: { code: 200, message: "OK" }, data: { success: true } });  
 }  
 } catch (err) {  
 console.log(err);  
 next(err);  
 }  
});  
router.post("/getmasterdata", async (req, res, next) => {  
 try {  
 let applicationReq = req.body;  
 applicationReq = capitalizeKeys(applicationReq);  
 const { Type, Code } = applicationReq;  
 if (Type == null || Code == null) {  
 return res.status(400).json({ error: "Type and Code are required" });  
 }  
 const masterData = await ScsMasterTableModel.findAll({  
 where: { Type, Code },  
 raw: true,  
 });  
 let response = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: masterData.map((data) => lowercaseKeys(data)),  
 };  
 return res.status(200).json(response);  
 } catch (err) {  
 next(err);  
 }  
});  
router.post("/find-aprse", async (req, res, next) => {  
 try {  
 const { chineseName, englishName, regType, certRegNo } = req.body;  
 const regNo = regType + " " + certRegNo.trim();  
 const apRse = await ApRseModel.findOne({  
 where: {  
 Name\_tc: chineseName,  
 Name: englishName,  
 RegistrationType: regType,  
 RegistrationNumber: regNo,  
 },  
 });  
 if (!apRse) {  
 return res.status(404).json({ error: "AP/RSE not found" });  
 }  
 const response = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: lowercaseKeys(apRse.dataValues),  
 };  
 return res.status(200).json(response);  
 } catch (err) {  
 next(err);  
 }  
});  
router.post("/find-Address", async (req, res, next) => {  
 try {  
 let applicationReq = req.body;  
 applicationReq = capitalizeKeys(applicationReq);  
 const {  
 BuildingId = "",  
 BuildingName = "",  
 StreetName = "",  
 DD = "",  
 Lot = "",  
 } = applicationReq;  
 let addresses = [];  
 if (BuildingId.trim() !== "") {  
 try {  
 const \_id = parseInt(BuildingId, 10);  
 const query = await sequelize.query(  
 `  
 SELECT  
 TOP 10  
 a.\*,  
 district.ENGLISH\_DESCRIPTION AS district\_e,  
 district.CHINESE\_DESCRIPTION AS district\_c,  
 (SELECT ENGLISH\_DESCRIPTION AS region\_e FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.SYS\_REGION\_ID) AS region\_e,  
 (SELECT CHINESE\_DESCRIPTION AS region\_c FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.SYS\_REGION\_ID) AS region\_c,  
 area.ENGLISH\_DESCRIPTION AS area\_e,  
 area.CHINESE\_DESCRIPTION AS area\_c,  
 (SELECT ENGLISH\_DESCRIPTION AS bt\_e FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.BLK\_TYPE\_ID) AS bt\_e,  
 (SELECT CHINESE\_DESCRIPTION AS bt\_c FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.BLK\_TYPE\_ID) AS bt\_c  
 FROM AdrBlk\_T a  
 LEFT JOIN Sys\_Meta\_Data\_T district  
 ON a.SYS\_DISTRICT\_ID = district.SYS\_META\_DATA\_ID  
 LEFT JOIN Sys\_Meta\_Data\_T region  
 ON a.SYS\_REGION\_ID = district.SYS\_META\_DATA\_ID  
 LEFT JOIN Sys\_Meta\_Data\_T area  
 ON a.AREA\_ID = area.SYS\_META\_DATA\_ID  
 WHERE a.ADR\_BLK\_ID = :id  
 AND a.OBSOLETE = 'N'  
 `,  
 {  
 replacements: { id: \_id },  
 type: QueryTypes.SELECT,  
 }  
 );  
 addresses = query;  
 } catch (e) {  
 console.error(e);  
 return [];  
 }  
 } else {  
 const query = await sequelize.query(  
 `  
 SELECT TOP 10  
 a.\*,  
 district.ENGLISH\_DESCRIPTION AS district\_e,  
 district.CHINESE\_DESCRIPTION AS district\_c,  
 (SELECT ENGLISH\_DESCRIPTION AS region\_e FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.SYS\_REGION\_ID) AS region\_e,  
 (SELECT CHINESE\_DESCRIPTION AS region\_c FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.SYS\_REGION\_ID) AS region\_c,  
 area.ENGLISH\_DESCRIPTION AS area\_e,  
 area.CHINESE\_DESCRIPTION AS area\_c,  
 (SELECT ENGLISH\_DESCRIPTION AS bt\_e FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.BLK\_TYPE\_ID) AS bt\_e,  
 (SELECT CHINESE\_DESCRIPTION AS bt\_c FROM Sys\_Meta\_Data\_T WHERE SYS\_META\_DATA\_ID = a.BLK\_TYPE\_ID) AS bt\_c  
 FROM AdrBlk\_T a  
 LEFT JOIN Sys\_Meta\_Data\_T district  
 ON a.SYS\_DISTRICT\_ID = district.SYS\_META\_DATA\_ID  
 LEFT JOIN Sys\_Meta\_Data\_T area  
 ON a.AREA\_ID = area.SYS\_META\_DATA\_ID  
 WHERE (  
 a.BLDG\_NAME\_E1 LIKE :buildingName  
 OR a.BLDG\_NAME\_E2 LIKE :buildingName  
 OR a.BLDG\_NAME\_E3 LIKE :buildingName  
 OR a.BLDG\_NAME\_C1 LIKE :buildingName  
 OR a.BLDG\_NAME\_C2 LIKE :buildingName  
 OR a.BLDG\_NAME\_C3 LIKE :buildingName  
 ) AND (  
 a.OSADR\_ST\_E1 LIKE :streetName  
 OR a.OSADR\_ST\_E2 LIKE :streetName  
 OR a.OSADR\_ST\_C LIKE :streetName  
 ) AND (  
 a.OSADR\_LOT\_E1 LIKE :dd  
 OR a.OSADR\_LOT\_E2 LIKE :dd  
 OR a.OSADR\_LOT\_E3 LIKE :dd  
 OR a.OSADR\_LOT\_E4 LIKE :dd  
 OR a.OSADR\_LOT\_C1 LIKE :dd  
 OR a.OSADR\_LOT\_C2 LIKE :dd  
 OR a.OSADR\_LOT\_C3 LIKE :dd  
 OR a.OSADR\_LOT\_C4 LIKE :dd  
 ) AND (  
 a.OSADR\_LOT\_E1 LIKE :lot  
 OR a.OSADR\_LOT\_E2 LIKE :lot  
 OR a.OSADR\_LOT\_E3 LIKE :lot  
 OR a.OSADR\_LOT\_E4 LIKE :lot  
 OR a.OSADR\_LOT\_C1 LIKE :lot  
 OR a.OSADR\_LOT\_C2 LIKE :lot  
 OR a.OSADR\_LOT\_C3 LIKE :lot  
 OR a.OSADR\_LOT\_C4 LIKE :lot  
 ) AND a.OBSOLETE = 'N'  
 `,  
 {  
 replacements: {  
 buildingName: `%${BuildingName}%`,  
 streetName: `%${StreetName}%`,  
 dd: `%${DD}%`,  
 lot: `%${Lot}%`,  
 },  
 type: QueryTypes.SELECT,  
 }  
 );  
 addresses = query;  
 }  
 let response = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: camelize(addresses),  
 };  
 return res.status(200).json(response);  
 } catch (err) {  
 next(err);  
 }  
});  
router.post("/checkschoolappstatus", async (req, res, next) => {  
 try {  
 const applicationNo = req.query.appNo;  
 const formName = req.query.formName;  
 const successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 applicationNo: applicationNo,  
 formName: formName,  
 status: "Submitted",  
 },  
 };  
 return res.status(200).json(successResponse);  
 } catch (err) {  
 next(err);  
 }  
});  
router.get("/unsynced-submissions", async (req, res, next) => {  
 try {  
 const submissions = await SchoolAppSubmissionModel.findAll({  
 where: { Synced: false, Status: "Submitted" },  
 raw: true,  
 limit: 5,  
 });  
 const submissionIds = submissions.map((submission) => submission.Id);  
 console.log("Getting unsynced submissions", submissionIds);  
 await SchoolAppSubmissionModel.update(  
 { Synced: true },  
 { where: { Id: submissionIds } }  
 );  
 return res.status(200).json(  
 submissions.map((submission) => {  
 try {  
 if (submission.ApList)  
 submission.APP13 = JSON.parse(submission.ApList);  
 if (submission.SelfCertification)  
 submission.SelfCertification = JSON.parse(  
 submission.SelfCertification  
 );  
 } catch (e) {}  
 return submission;  
 })  
 );  
 } catch (err) {  
 console.log(err);  
 next(err);  
 }  
});  
router.post(  
 "/backend-update",  
 upload.single("file"),  
 async (req, res, next) => {  
 try {  
 const { applicationNo, stage, updateType } = req.body;  
 console.log("Update Body", req.body);  
 let newUpdate = {  
 ApplicationNo: applicationNo,  
 Stage: stage,  
 UpdateType: updateType,  
 };  
 console.log("Update File", req.file);  
 if (req.file) {  
 let fileObj = req.file;  
 newUpdate.FileMimeType = req.file.mimetype;  
 delete fileObj.mimetype;  
 newUpdate.File = JSON.stringify(fileObj);  
 }  
 const backendUpdate = await BackendUpdateModel.create(newUpdate);  
 if (stage) {  
 await SchoolAppInfoModel.update(  
 { Stage: stage },  
 { where: { ApplicationNo: applicationNo } }  
 );  
 }  
 if (!backendUpdate) {  
 return res.status(400).json({ error: "Update failed" });  
 }  
 const successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 updateId: backendUpdate.dataValues.Id,  
 },  
 };  
 return res.status(200).json(successResponse);  
 } catch (err) {  
 next(err);  
 }  
 }  
);  
module.exports = router;

## File: bd-scs-nodejs-frontend/src/routes/AuthController.js

var express = require("express");  
const axios = require("axios");  
var router = express.Router();  
const GenOtpModel = require("../models/GenOtp");  
const { getUserJsonWithToken } = require("../utils/loginUtils");  
const ScsMasterTableModel = require("../models/ScsMasterTable");  
const sendEmail = require("../utils/sendEmail");  
require("dotenv").config();  
router.post("/getUserLoginWithToken", async (req, res, next) => {  
 try {  
 const user = await getUserJsonWithToken(req.body);  
 if (user) {  
 const successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 userId: user.userId,  
 role: user.role,  
 token: user.token,  
 },  
 };  
 res.status(200).json(successResponse);  
 } else {  
 const errorResponse = {  
 meta: {  
 code: 400,  
 message: "Invalid applicationNo or userId",  
 },  
 data: {},  
 };  
 res.status(400).json(errorResponse);  
 }  
 } catch (err) {  
 next(err);  
 }  
});  
router.post("/getotp", async (req, res, next) => {  
 try {  
 const { applicationNo, userId } = req.body;  
 await GenOtpModel.destroy({  
 where: {  
 ApplicationNo: applicationNo,  
 UserId: userId,  
 },  
 });  
 const randomNumber =  
 Math.floor(Math.random() \* (999999 - 100000 + 1)) + 100000;  
 const currentDate = new Date();  
 const newOtp = await GenOtpModel.create({  
 ApplicationNo: applicationNo,  
 UserId: userId,  
 Otp: randomNumber,  
 EffectiveUntil: new Date(currentDate.getTime() + 5 \* 60000),  
 Used: 0,  
 });  
 if (!newOtp) {  
 return res.status(400).json({ error: "Create application fail" });  
 }  
 if (/^\d+$/.test(userId)) {  
 let message = "";  
 const query = await ScsMasterTableModel.findOne({  
 where: { Type: "otp\_login\_msg", Code: "sms\_message" },  
 });  
 message = query?.CaptionEN || ""; // Default to English  
 message = message.replace("[otp]", randomNumber);  
 // console.log(message);  
 // Send SMS  
 const smsUserName = process.env.SMS\_USERNAME;  
 const smsPassword = process.env.SMS\_PASSWORD;  
 // Build URL  
 const url = `${  
 process.env.SMS\_API\_URL  
 }/?username=${smsUserName}&password=${smsPassword}&destinatingAddress=852${userId.trim()}&sms=${message}&type=1`;  
 try {  
 // Make GET request  
 const response = await axios.get(url);  
 console.log("SMS sent successfully:", response.data);  
 } catch (error) {  
 console.error("Error sending SMS:", error.message);  
 throw error; // Rethrow error for the caller to handle  
 }  
 } else {  
 // Send Email  
 // Fetch email subject and body from database  
 const query = await ScsMasterTableModel.findOne({  
 where: { Type: "otp\_login\_msg", Code: "email\_message" },  
 });  
 const subject = query?.CaptionEN || ""; // Default to English  
 let body = query?.LongTextEN || ""; // Default to English  
 body = body.replace("[otp]", randomNumber);  
 // Fetch OTP effective time from database  
 const effectiveQuery = await ScsMasterTableModel.findOne({  
 where: { Type: "admin\_setting", Code: "otp\_effective\_min" },  
 });  
 const effectiveMin = effectiveQuery?.DataValue || "";  
 body = body.replace("[effective\_min]", effectiveMin);  
 // Send Email  
 await sendEmail(userId, subject, body);  
 }  
 const successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 result: "One Time Password (OTP) has been generated.",  
 },  
 };  
 res.status(200).json(successResponse);  
 } catch (err) {  
 next(err);  
 }  
});  
router.post("/getotp-bysms", async (req, res, next) => {  
 try {  
 const mobile = req.query.mobile.trim();  
 await GenOtpModel.destroy({  
 where: {  
 UserId: mobile,  
 },  
 });  
 const randomNumber =  
 Math.floor(Math.random() \* (999999 - 100000 + 1)) + 100000;  
 const currentDate = new Date();  
 const newOtp = await GenOtpModel.create({  
 ApplicationNo: "",  
 UserId: mobile,  
 Otp: randomNumber,  
 EffectiveUntil: new Date(currentDate.getTime() + 5 \* 60000),  
 Used: 0,  
 });  
 if (!newOtp) {  
 return res.status(400).json({ error: "Create application fail" });  
 }  
 let message = "";  
 const query = await ScsMasterTableModel.findOne({  
 where: { Type: "otp\_login\_msg", Code: "sms\_message" },  
 });  
 message = query?.CaptionEN || ""; // Default to English  
 message = message.replace("[otp]", randomNumber);  
 // check mobile is email address or not  
 if (mobile.includes("@")) {  
 // Send Email  
 const messageEmailOtp = `  
 Dear Sir/Madam,<br><br>  
 Please use the One-time Password below to login.<br><br>  
 One-time Password (OTP): ${randomNumber}<br><br>  
 This OTP is valid for 5 minutes only.<br><br>  
 Please do not disclose it for security purpose.<br><br>  
 For enquiries, please contact us by phone 2626 1616 (handled by 1823) or via email lscphelpdesk@bd.gov.hk.<br><br>  
 Please <u>do not</u> reply to this email as it comes from an automated mailbox.<br><br>  
 Buildings Department  
 `;  
 const subject =  
 "Buildings Department Licensing Self-certification Portal (LSCP) – One-time Password for Login 屋宇署牌照申請自行核證平台 – 一次性登入密碼";  
 await sendEmail(mobile, subject, messageEmailOtp);  
 } else {  
 // Send SMS  
 const smsUserName = process.env.SMS\_USERNAME;  
 const smsPassword = process.env.SMS\_PASSWORD;  
 // Build URL  
 const url = `${  
 process.env.SMS\_API\_URL  
 }/?username=${smsUserName}&password=${smsPassword}&destinatingAddress=852${mobile.trim()}&sms=${message}&type=1`;  
 try {  
 // Make GET request  
 const response = await axios.get(url);  
 console.log("SMS sent successfully:", response.data);  
 } catch (error) {  
 console.error("Error sending SMS:", error.message);  
 throw error; // Rethrow error for the caller to handle  
 }  
 }  
 const successResponse = {  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 result: "One Time Password (OTP) has been generated.",  
 },  
 };  
 res.status(200).json(successResponse);  
 } catch (err) {  
 next(err);  
 }  
});  
router.post("/otp-verification", async (req, res, next) => {  
 try {  
 const { applicationNo, userId, verificationCode } = req.body;  
 if (userId === process.env.ADMIN\_BYPASS\_ID) {  
 return res.status(200).json({  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 result: "Verification success",  
 },  
 });  
 }  
 await GenOtpModel.findOne({  
 where: {  
 ApplicationNo: applicationNo,  
 UserId: userId,  
 },  
 }).then(async (otp) => {  
 if (otp) {  
 if (otp.Otp === verificationCode && new Date() < otp.EffectiveUntil) {  
 await GenOtpModel.destroy({  
 where: {  
 ApplicationNo: applicationNo,  
 UserId: userId,  
 },  
 });  
 res.status(200).json({  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 result: "Verification success",  
 },  
 });  
 } else {  
 res.status(400).json({  
 meta: {  
 code: 400,  
 message: "Invalid verification code",  
 },  
 data: {},  
 });  
 }  
 } else {  
 res.status(400).json({  
 meta: {  
 code: 400,  
 message: "Invalid applicationNo or userId",  
 },  
 data: {},  
 });  
 }  
 });  
 } catch (err) {  
 next(err);  
 }  
});  
router.post("/sendverifyotpbysms", async (req, res, next) => {  
 try {  
 const mobile = req.query.mobile.trim();  
 const code = req.query.code.trim();  
 const otp = await GenOtpModel.findOne({  
 where: {  
 UserId: mobile,  
 },  
 });  
 if (otp) {  
 if (otp.Otp === code && new Date() < otp.EffectiveUntil) {  
 await GenOtpModel.destroy({  
 where: {  
 UserId: mobile,  
 },  
 });  
 res.status(200).json({  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 result: "Verification success",  
 },  
 });  
 } else {  
 res.status(400).json({  
 meta: {  
 code: 400,  
 message: "Invalid verification code",  
 },  
 data: {},  
 });  
 }  
 } else {  
 res.status(400).json({  
 meta: {  
 code: 400,  
 message: "Invalid mobile",  
 },  
 data: {},  
 });  
 }  
 } catch (err) {  
 next(err);  
 }  
});  
module.exports = router;

## File: bd-scs-nodejs-frontend/src/routes/ESignController.js

var express = require("express");  
var router = express.Router();  
const IamSmartModel = require("../models/IamSmart");  
const sequelize = require("../config/database");  
const multer = require("multer");  
const { hkpostSign } = require("../utils/hkpostUtils");  
const upload = multer({ dest: "temp\_uploads/" });  
const {  
 initiateIamsmartRequest,  
 getAccessTokenOpenID,  
 getSignResult,  
 insertSignature,  
} = require("../utils/iamSmartUtils");  
function toCamelCase(string) {  
 return string[0].toLowerCase() + string.slice(1);  
}  
router.post("/hkpost", upload.single("certFile"), async (req, res, next) => {  
 try {  
 const { fileId, certPassword } = req.body;  
 const file = await hkpostSign(fileId, req.file.path, certPassword);  
 const fileRes = Object.keys(file).reduce((acc, cur) => {  
 acc[toCamelCase(cur)] = file[cur];  
 return acc;  
 }, {});  
 res.status(200).json({  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: fileRes,  
 });  
 } catch (err) {  
 next(err);  
 }  
});  
router.post("/iamsmart", async (req, res, next) => {  
 try {  
 const { fileId, hkic, documentName, lang, currentUrl } = req.body;  
 const { signCode, qrcodeUrl } = await initiateIamsmartRequest(  
 fileId,  
 hkic,  
 documentName,  
 lang,  
 currentUrl  
 );  
 res.status(200).json({  
 meta: {  
 code: 200,  
 message: "OK",  
 },  
 data: {  
 signCode,  
 qrcodeUrl,  
 },  
 });  
 } catch (err) {  
 next(err);  
 }  
});  
router.get("/iamsmart/callback", async (req, res, next) => {  
 try {  
 const { code, state, businessID, error\_code } = req.query;  
 console.log("Callback res", req.query);  
 if (error\_code) {  
 const file = (  
 await sequelize.query(  
 `SELECT \* FROM IamSmart I INNER JOIN ApplicationFiles A ON I.FileId = A.FileId WHERE I.BusinessId = '${businessID}'`  
 )  
 )[0][0];  
 const currentUrl = file.CurrentUrl  
 if (currentUrl.includes("?")) {  
 if (currentUrl.includes("&msg")) {  
 const modifyUrl = currentUrl.split("&msg")[0];  
 return res.redirect(modifyUrl + "&msg=" + error\_code);  
 }  
 return res.redirect(currentUrl + "&msg=" + error\_code);  
 } else {  
 return res.redirect(currentUrl + "?msg=" + error\_code);  
 }  
 }  
 const { accessToken, openID } = await getAccessTokenOpenID(code);  
 const signature = await getSignResult(accessToken, openID);  
 const currentUrl = await insertSignature(businessID, signature);  
 if (currentUrl.includes("&msg")) {  
 const modifyUrl = currentUrl.split("&msg")[0];  
 return res.redirect(modifyUrl);  
 }  
 return res.redirect(currentUrl);  
 } catch (err) {  
 next(err);  
 }  
});  
module.exports = router;

## File: bd-scs-nodejs-frontend/src/utils/aes256gcm.js

const crypto = require("crypto");  
const IV\_LENGTH = 12;  
const aes256gcm = (key) => {  
 const encrypt = (str) => {  
 const iv = new crypto.randomBytes(IV\_LENGTH);  
 const cipher = crypto.createCipheriv("aes-256-gcm", key, iv);  
 let enc1 = cipher.update(str);  
 let enc2 = cipher.final();  
 const ivLength = Buffer.alloc(4);  
 ivLength.writeUInt32BE(IV\_LENGTH, 0);  
 console.log(  
 Buffer.concat([ivLength, iv, enc1, cipher.getAuthTag()]).length  
 );  
 return Buffer.concat([ivLength, iv, enc1, cipher.getAuthTag()]).toString(  
 "base64"  
 );  
 };  
 const decrypt = (enc) => {  
 enc = Buffer.from(enc, "base64");  
 const iv = enc.slice(4, 4 + IV\_LENGTH);  
 const tag = enc.slice(enc.length - 16);  
 const decipher = crypto.createDecipheriv("aes-256-gcm", key, iv);  
 decipher.setAuthTag(tag);  
 const str = decipher.update(enc.slice(4 + IV\_LENGTH, enc.length - 16));  
 return str.toString("utf8");  
 };  
 return {  
 encrypt,  
 decrypt,  
 };  
};  
module.exports = aes256gcm;

## File: bd-scs-nodejs-frontend/src/utils/applicationUtils.js

const SchoolAppSubmissionModel = require("../models/SchoolAppSubmission");  
const { Op } = require("sequelize");  
const APPLICATION\_NO\_TYPES = {  
 NEWSCH: 12,  
 EXTSCH: 12,  
 NLHE: 12,  
 NEWCCC: 13,  
 NEWJOKC: 13,  
 NEWMUT: 13,  
};  
async function generateApplicationNo(applicationType) {  
 console.log("generateApplicationNo", applicationType);  
 const year = new Date().getFullYear();  
 if (!Object.keys(APPLICATION\_NO\_TYPES).includes(applicationType)) {  
 return null;  
 }  
 const type = APPLICATION\_NO\_TYPES[applicationType];  
 const regexPattern = `F${year}${type}%`;  
 try {  
 const lastApplication = await SchoolAppSubmissionModel.findOne({  
 where: {  
 ApplicationNo: {  
 [Op.like]: regexPattern,  
 },  
 },  
 order: [["ApplicationNo", "DESC"]],  
 });  
 let count = 1;  
 if (lastApplication) {  
 const lastCount = parseInt(lastApplication.ApplicationNo.slice(-5));  
 count = lastCount + 1;  
 }  
 console.log("count", count);  
 const paddedCount = count.toString().padStart(5, "0");  
 const newApplicationNo = `F${year}${type}${paddedCount}`;  
 return newApplicationNo;  
 } catch (error) {  
 console.log(error);  
 return null;  
 }  
}  
module.exports = { generateApplicationNo };

## File: bd-scs-nodejs-frontend/src/utils/ExternalSigner.js

import \* as nodeCrypto from 'crypto';  
import \* as asn1js from 'asn1js';  
import \* as pkijs from 'pkijs';  
import {ISigner, SignPdfError} from '@signpdf/utils';  
import {Signer} from "./Signer"  
export class ExternalSigner extends Signer {  
 async getSignature(hash, data) {  
 throw new SignPdfError(  
 `getSignature() is not implemented on ${this.constructor.name}`,  
 SignPdfError.TYPE\_INPUT,  
 );  
 }  
 getCrypto() {  
 const crypto = super.getCrypto();  
 crypto.sign = async (\_algo, \_key, data) => {  
 const hash = await crypto.digest({name: this.hashAlgorithm}, data);  
 const signature = await this.getSignature(Buffer.from(hash), Buffer.from(data));  
 return signature;  
 };  
 return crypto;  
 }  
 async obtainKey() {  
 const algorithmParams = this.crypto.getAlgorithmParameters(this.signAlgorithm, 'generatekey').algorithm;  
 const keypair = await this.crypto.generateKey({  
 name: this.signAlgorithm,  
 ...algorithmParams,  
 hash: {name: this.hashAlgorithm},  
 }, false, ['sign', 'verify']);  
 return keypair.privateKey;  
 }  
}

## File: bd-scs-nodejs-frontend/src/utils/hkpostUtils.js

const { signWithPlaceholder } = require("./signUtils");  
const fs = require("fs");  
var P12Signer = require("@signpdf/signer-p12").P12Signer;  
const ApplicationFiles = require("../models/ApplicationFile");  
const sequelize = require("../config/database");  
const { signConfig } = require("./signConfig");  
const MOMENT = require("moment");  
const hkpostSign = async (fileId, certFilePath, certPassword) => {  
 const certFile = fs.readFileSync(certFilePath);  
 console.log(certFile.length, certPassword);  
 var signer = new P12Signer(certFile, {  
 passphrase: certPassword,  
 });  
 const UPLOAD\_DIR = "wwwroot/uploads/";  
 const targetFile = UPLOAD\_DIR + fileId + ".pdf";  
 const applicantName = (  
 await sequelize.query(  
 `SELECT SI.ApplicantName FROM ApplicationFiles AF JOIN SchoolApp\_Infos SI ON AF.ApplicationNo = SI.ApplicationNo WHERE AF.FileId = '${fileId}'`  
 )  
 )[0][0]["ApplicantName"];  
 const DocumentType = (  
 await sequelize.query(  
 `SELECT AF.DocumentType FROM ApplicationFiles AF JOIN SchoolApp\_Submissions SS ON AF.SubmissionId = SS.SubmissionId WHERE AF.FileId = '${fileId}'`  
 )  
 )[0][0]["DocumentType"];  
 let signType = "";  
 if (DocumentType == "form") {  
 signType = (  
 await sequelize.query(  
 `SELECT SS.FormName FROM ApplicationFiles AF JOIN SchoolApp\_Submissions SS ON AF.SubmissionId = SS.SubmissionId WHERE AF.FileId = '${fileId}'`  
 )  
 )[0][0]["FormName"];  
 } else if (DocumentType == "layout-plan") {  
 const pdf = fs.readFileSync(targetFile);  
 const pdfjs = await import("pdfjs-dist/legacy/build/pdf.mjs");  
 const pdfData = new Uint8Array(pdf);  
 const pdfDoc = await pdfjs.getDocument(pdfData).promise;  
 const page = await pdfDoc.getPage(1);  
 const viewport = page.getViewport({ scale: 1 });  
 const pdfWidth = viewport.width;  
 const pdfHeight = viewport.height;  
 if (  
 pdfHeight >= 595 &&  
 pdfHeight <= 596 &&  
 pdfWidth >= 841 &&  
 pdfWidth <= 842  
 ) {  
 signType = "A4";  
 } else {  
 signType = "A3";  
 }  
 }  
 const signX = signConfig[signType]?.x || 400;  
 const signY = signConfig[signType]?.y || 200;  
 const signPage = signConfig[signType]?.page || 0;  
 let visibleSignatures = [];  
 if (signType == "A3" || signType == "A4") {  
 visibleSignatures.push({  
 x: signX,  
 y: signY,  
 pageNo: signPage,  
 fontSize: 9,  
 displayText: `Digitally Signed By:\n ${applicantName}\nDated: ${MOMENT().format(  
 "DD MMMM YYYY"  
 )}`,  
 });  
 visibleSignatures.push({  
 x: signX,  
 y: signY,  
 pageNo: signPage,  
 fontSize: 9,  
 displayText: `Digitally Signed By:\n ${applicantName}\nDated: ${MOMENT().format(  
 "DD MMMM YYYY"  
 )}`,  
 });  
 visibleSignatures.push({  
 x: signX,  
 y: signY - 200,  
 pageNo: signPage,  
 maxWidth: 150,  
 fontSize: 9,  
 displayText: `RECEIVED ON - ${MOMENT().format(  
 "DD MMMM YYYY"  
 )} BUILDINGS DEPT. (Lic. Unit)`,  
 });  
 } else {  
 visibleSignatures.push({  
 x: signX,  
 y: signY,  
 pageNo: signPage,  
 fontSize: 12,  
 displayText: "Signed By " + applicantName + " (hkpost)",  
 });  
 }  
 await signWithPlaceholder(  
 targetFile,  
 targetFile,  
 {  
 signatureLength: 22000,  
 },  
 signer,  
 visibleSignatures  
 );  
 const file = (  
 await ApplicationFiles.update(  
 {  
 SignDate: MOMENT().format("YYYY-MM-DD HH:mm:ss.0000000"),  
 SignType: "hkpost",  
 },  
 {  
 where: {  
 FileId: fileId,  
 },  
 returning: true,  
 plain: true,  
 }  
 )  
 )[1]["dataValues"];  
 return file;  
};  
module.exports = { hkpostSign };

## File: bd-scs-nodejs-frontend/src/utils/iamSmartUtils.js

const axios = require("axios");  
const crypto = require("crypto");  
const fs = require("fs");  
const forge = require("node-forge");  
const NodeRSA = require("node-rsa");  
const aes256gcm = require("./aes256gcm");  
const querystring = require("querystring");  
const IamSmartModel = require("../models/IamSmart");  
const ApplicationFileModel = require("../models/ApplicationFile");  
const ApplicationFiles = require("../models/ApplicationFile");  
const sequelize = require("../config/database");  
const {  
 signWithPlaceholder,  
 externalSign,  
 prepareDocDigest,  
 getSignCode,  
} = require("./signUtils");  
const { signPdf } = require("pdf-signatures");  
const { signConfig } = require("./signConfig");  
require("dotenv").config();  
const MOMENT = require("moment");  
function decryptWithPrivateKey(privateKey, encryptedData) {  
 const keyRSA = new NodeRSA(privateKey, "private", {  
 encryptionScheme: "pkcs1",  
 });  
 keyRSA.setOptions({ environment: "browser", encryptionScheme: "pkcs1" });  
 return keyRSA.decrypt(encryptedData);  
}  
function createHmacSha256Signature(secret, data) {  
 const hmac = crypto.createHmac("sha256", secret);  
 const hash = hmac.update(data).digest("base64");  
 const signature = encodeURIComponent(hash);  
 return signature;  
}  
function initHttpClient(requestBodyJson) {  
 const client = axios.create();  
 const timeStamp = Date.now().toString();  
 const nonce = crypto.randomUUID().replace(/-/g, ""); // Generate a nonce without dashes  
 // Default headers  
 const headers = {  
 clientID: process.env.IAM\_SMART\_CLIENT\_ID,  
 signatureMethod: "HmacSHA256",  
 timestamp: timeStamp,  
 nonce: nonce,  
 };  
 let paramStr;  
 if (requestBodyJson) {  
 paramStr = `${headers.clientID}${headers.signatureMethod}${timeStamp}${nonce}${requestBodyJson}`;  
 } else {  
 paramStr = `${headers.clientID}${headers.signatureMethod}${timeStamp}${nonce}`;  
 }  
 const signature = createHmacSha256Signature(  
 process.env.IAM\_SMART\_CLIENT\_SECRET,  
 paramStr  
 );  
 headers.signature = signature;  
 client.defaults.headers = headers;  
 return client;  
}  
async function getKey() {  
 const client = initHttpClient();  
 const uri = process.env.IAM\_SMART\_DOMAIN + "api/v1/security/getKey";  
 try {  
 const response = await client.post(uri, null);  
 let result = response.data;  
 if (result.code !== "D00000") {  
 return null;  
 }  
 return result;  
 } catch (error) {  
 console.error(error);  
 return null;  
 }  
}  
async function getCek() {  
 const result = await getKey();  
 if (result === null) {  
 console.error("获取CEK失败");  
 return;  
 }  
 const secretKeyBytes = Buffer.from(result.content.secretKey, "base64");  
 const privateKeyFilePath = process.env.IAM\_SMART\_CERT\_FILE;  
 const keyFile = fs.readFileSync(privateKeyFilePath, "binary");  
 const p12Asn1 = forge.asn1.fromDer(keyFile);  
 const p12 = forge.pkcs12.pkcs12FromAsn1(  
 p12Asn1,  
 process.env.IAM\_SMART\_CERT\_PASSWORD  
 );  
 var keyBag = p12.getBags({ bagType: forge.pki.oids.pkcs8ShroudedKeyBag })[  
 forge.pki.oids.pkcs8ShroudedKeyBag  
 ][0];  
 var privateKey = forge.pki.privateKeyToPem(keyBag.key);  
 var certBag = p12.getBags({ bagType: forge.pki.oids.certBag })[  
 forge.pki.oids.certBag  
 ][0];  
 const publicKey = forge.pki.publicKeyToPem(certBag.cert.publicKey);  
 const cek = decryptWithPrivateKey(privateKey, secretKeyBytes);  
 const encryptedData = crypto.publicEncrypt(  
 {  
 key: publicKey,  
 padding: crypto.constants.RSA\_PKCS1\_PADDING,  
 },  
 cek  
 );  
 const decryptedData2 = decryptWithPrivateKey(privateKey, encryptedData);  
 return cek;  
}  
const encryptRequestBody = async (cek, body) => {  
 const aes = aes256gcm(cek);  
 const encryptedBody = aes.encrypt(JSON.stringify(body));  
 return {  
 ...body,  
 content: encryptedBody,  
 };  
};  
const decryptResponse = async (cek, response) => {  
 const decryptedRes = aes256gcm(cek).decrypt(response);  
 return JSON.parse(decryptedRes);  
};  
const getQRCodePage = async (ticketId, lang = "en-US") => {  
 const queryParams = {  
 clientID: process.env.IAM\_SMART\_CLIENT\_ID,  
 responseType: "code",  
 source: "PC\_Browser",  
 redirectURI: encodeURIComponent(process.env.IAM\_SMART\_RedirectURI),  
 scope: encodeURIComponent("eidapi\_sign"),  
 ticketID: ticketId,  
 lang: lang,  
 state: require("crypto").randomUUID(),  
 brokerPage: "false",  
 };  
 const qr\_code\_url = `${  
 process.env.IAM\_SMART\_QR\_DOMAIN  
 }api/v1/auth/getQR?${querystring.stringify(queryParams)}`;  
 return qr\_code\_url;  
};  
const initiateIamsmartRequest = async (  
 fileId,  
 hkic,  
 documentName,  
 lang,  
 currentUrl  
) => {  
 const cek = await getCek();  
 const uuid = require("uuid");  
 const businessId = uuid.v4();  
 try {  
 await IamSmartModel.create({  
 FileId: fileId,  
 BusinessId: businessId,  
 CurrentUrl: currentUrl,  
 });  
 } catch (err) {  
 console.log(err);  
 }  
 const UPLOAD\_DIR = "wwwroot/uploads/";  
 const targetFile = UPLOAD\_DIR + fileId + ".pdf";  
 const applicantName = (  
 await sequelize.query(  
 `SELECT SI.ApplicantName FROM ApplicationFiles AF JOIN SchoolApp\_Infos SI ON AF.ApplicationNo = SI.ApplicationNo WHERE AF.FileId = '${fileId}'`  
 )  
 )[0][0]["ApplicantName"];  
 const DocumentType = (  
 await sequelize.query(  
 `SELECT AF.DocumentType FROM ApplicationFiles AF JOIN SchoolApp\_Submissions SS ON AF.SubmissionId = SS.SubmissionId WHERE AF.FileId = '${fileId}'`  
 )  
 )[0][0]["DocumentType"];  
 let signType = "";  
 if (DocumentType == "form") {  
 signType = (  
 await sequelize.query(  
 `SELECT SS.FormName FROM ApplicationFiles AF JOIN SchoolApp\_Submissions SS ON AF.SubmissionId = SS.SubmissionId WHERE AF.FileId = '${fileId}'`  
 )  
 )[0][0]["FormName"];  
 } else if (DocumentType == "layout-plan") {  
 const pdf = fs.readFileSync(targetFile);  
 const pdfjs = await import("pdfjs-dist");  
 const pdfData = new Uint8Array(pdf);  
 const pdfDoc = await pdfjs.getDocument(pdfData).promise;  
 const page = await pdfDoc.getPage(1);  
 const viewport = page.getViewport({ scale: 1 });  
 const pdfWidth = viewport.width;  
 const pdfHeight = viewport.height;  
 if (  
 pdfHeight >= 595 &&  
 pdfHeight <= 596 &&  
 pdfWidth >= 841 &&  
 pdfWidth <= 842  
 ) {  
 signType = "A4";  
 } else {  
 signType = "A3";  
 }  
 }  
 const signX = signConfig[signType]?.x || 400;  
 const signY = signConfig[signType]?.y || 200;  
 const pdfWithPlaceholder = await signWithPlaceholder(  
 targetFile,  
 targetFile,  
 {  
 signatureLength: 20998,  
 },  
 null,  
 [  
 {  
 x: signX,  
 y: signY,  
 pageNo: 0,  
 fontSize: 12,  
 displayText: "Signed By " + applicantName + " (iamsmart)",  
 },  
 ]  
 );  
 const docDigest = await prepareDocDigest(pdfWithPlaceholder);  
 const hkicHash = require("crypto")  
 .createHash("sha256")  
 .update(hkic)  
 .digest("base64");  
 const DEPARTMENT = "Buildings Department";  
 const SERVICE\_NAME = "LSCP";  
 const req = {  
 businessID: businessId,  
 docDigest: docDigest,  
 HKICHash: hkicHash,  
 department: DEPARTMENT,  
 serviceName: SERVICE\_NAME,  
 documentName: documentName,  
 };  
 const signCode = getSignCode(  
 docDigest,  
 hkicHash,  
 process.env.IAM\_SMART\_CLIENT\_ID  
 );  
 const encryptedReq = await encryptRequestBody(cek, req);  
 const client = initHttpClient(JSON.stringify(encryptedReq));  
 const init\_sign\_url = `${process.env.IAM\_SMART\_DOMAIN}api/v1/anonymous/pdfsigning/initiateRequest`;  
 const result = await client.post(init\_sign\_url, encryptedReq);  
 const decryptedRes = await decryptResponse(cek, result.data.content);  
 console.log("decryptedRes:", decryptedRes);  
 const ticketId = decryptedRes.ticketID;  
 return { signCode: signCode, qrcodeUrl: await getQRCodePage(ticketId, lang) };  
};  
const getAccessTokenOpenID = async (code) => {  
 const cek = await getCek();  
 const req = {  
 code,  
 grantType: "authorization\_code",  
 };  
 const encryptedReq = await encryptRequestBody(cek, req);  
 const client = initHttpClient(JSON.stringify(encryptedReq));  
 const access\_token\_url = `${process.env.IAM\_SMART\_DOMAIN}api/v1/auth/getToken`;  
 const result = await client.post(access\_token\_url, encryptedReq);  
 console.log("Result", result.data);  
 const decryptedRes = await decryptResponse(cek, result.data.content);  
 return {  
 accessToken: decryptedRes.accessToken,  
 openID: decryptedRes.openID,  
 };  
};  
const getSignResult = async (accessToken, openID) => {  
 const cek = await getCek();  
 const req = {  
 accessToken,  
 openID,  
 };  
 const encryptedReq = await encryptRequestBody(cek, req);  
 const client = initHttpClient(JSON.stringify(encryptedReq));  
 const sign\_result\_url = `${process.env.IAM\_SMART\_DOMAIN}api/v1/anonymous/pdfsigning/getResult`;  
 const result = await client.post(sign\_result\_url, encryptedReq);  
 const decryptedRes = await decryptResponse(cek, result.data.content);  
 return decryptedRes.pdfSignature;  
};  
const insertSignature = async (businessId, signature) => {  
 const file = (  
 await sequelize.query(  
 `SELECT \* FROM IamSmart I INNER JOIN ApplicationFiles A ON I.FileId = A.FileId WHERE I.BusinessId = '${businessId}'`  
 )  
 )[0][0];  
 let pdfBuffer = fs.readFileSync(file.FilePath);  
 const binarySignature = Buffer.from(signature, "base64").toString("binary");  
 pdfBuffer = await externalSign(pdfBuffer, binarySignature);  
 console.log("Signed pdf length: " + pdfBuffer.length);  
 fs.writeFileSync(file.FilePath, pdfBuffer);  
 await ApplicationFiles.update(  
 {  
 SignDate: MOMENT().format("YYYY-MM-DD HH:mm:ss.0000000"),  
 SignType: "iamsmart",  
 },  
 {  
 where: {  
 FileId: file.FileId,  
 },  
 }  
 );  
 return file.CurrentUrl;  
};  
module.exports = {  
 getKey,  
 getCek,  
 initHttpClient,  
 encryptRequestBody,  
 decryptResponse,  
 initiateIamsmartRequest,  
 getAccessTokenOpenID,  
 getSignResult,  
 insertSignature,  
};

## File: bd-scs-nodejs-frontend/src/utils/loginUtils.js

const jwt = require("jsonwebtoken");  
const { Op } = require("sequelize");  
const SchoolAppInfoModel = require("../models/SchoolAppInfo");  
const \_jwtConfigAudience = process.env.JWT\_AUDIENCE;  
const \_jwtConfigIssuer = process.env.JWT\_ISSUER;  
const \_jwtConfigSecret = process.env.JWT\_SECRET;  
require("dotenv").config();  
async function checkApplicantLogin(user) {  
 return true;  
}  
async function getUserJsonWithToken(user) {  
 const schoolExisted = await checkUserDataInDB(  
 user.applicationNo,  
 user.userId  
 );  
 if (schoolExisted.isExisted) {  
 const token = generateToken(user.userId, schoolExisted.role);  
 return {  
 userId: user.userId,  
 role: schoolExisted.role,  
 token: token,  
 };  
 } else {  
 return null;  
 }  
}  
async function generateToken(userId, role) {  
 const payload = { userId, role };  
 const token = jwt.sign(payload, \_jwtConfigSecret, {  
 issuer: \_jwtConfigIssuer,  
 audience: \_jwtConfigAudience,  
 expiresIn: "3h",  
 });  
 return token;  
}  
async function getTokenByUser(user) {  
 if (user) {  
 const token = generateToken(user.userId, null);  
 return {  
 userId: user.userId,  
 token: token,  
 };  
 } else {  
 return null;  
 }  
}  
async function getTokenByStaff(staff) {  
 if (staff) {  
 const token = generateToken(staff.userId, staff.role);  
 return {  
 userId: staff.userId,  
 englishName: staff.englishName,  
 chineseName: staff.chineseName,  
 department: staff.department,  
 post: staff.post,  
 email: staff.email,  
 role: staff.role,  
 token: token,  
 };  
 } else {  
 return null;  
 }  
}  
async function checkUserDataInDB(applicationNo, userId) {  
 if (userId === process.env.ADMIN\_BYPASS\_ID) {  
 return {  
 isExisted: true,  
 role: ["APPLICANT", "CONTACTP", "AP1", "AP2", "RSE"],  
 };  
 }  
 const schoolAppInfo = await SchoolAppInfoModel.findOne({  
 where: {  
 ApplicationNo: applicationNo,  
 [Op.or]: [  
 { ApplicantEmail: userId },  
 { ApplicantMobile: userId },  
 { ContactPEmail: userId },  
 { ContactPMobile: userId },  
 { EmailOfAp1: userId },  
 { MobileOfAp1: userId },  
 { EmailOfAp2: userId },  
 { MobileOfAp2: userId },  
 { EmailOfRse: userId },  
 { MobileOfRse: userId },  
 ],  
 },  
 raw: true,  
 });  
 if (schoolAppInfo) {  
 const roles = [];  
 console.log(schoolAppInfo);  
 if (  
 schoolAppInfo.ApplicantEmail === userId ||  
 schoolAppInfo.ApplicantMobile === userId  
 ) {  
 roles.push("APPLICANT");  
 }  
 if (  
 schoolAppInfo.ContactPEmail === userId ||  
 schoolAppInfo.ContactPMobile === userId  
 ) {  
 roles.push("CONTACTP");  
 }  
 if (  
 schoolAppInfo.EmailOfAp1 === userId ||  
 schoolAppInfo.MobileOfAp1 === userId  
 ) {  
 roles.push("AP1");  
 }  
 if (  
 schoolAppInfo.EmailOfAp2 === userId ||  
 schoolAppInfo.MobileOfAp2 === userId  
 ) {  
 roles.push("AP2");  
 }  
 return { isExisted: true, role: roles.join(",") };  
 }  
 return { isExisted: false, role: "" };  
}  
module.exports = {  
 checkApplicantLogin,  
 getUserJsonWithToken,  
 generateToken,  
 getTokenByUser,  
 getTokenByStaff,  
 checkUserDataInDB,  
};

## File: bd-scs-nodejs-frontend/src/utils/on9Dotnet.js

const { isArray, transform, isObject, isDate } = require("lodash");  
function convertToCamelCase(name) {  
 if (!name || !isUpperCase(name[0])) {  
 return name;  
 }  
 const chars = name.split("");  
 fixCasing(chars);  
 return chars.join("");  
}  
const camelize = (obj) =>  
 transform(obj, (result, value, key, target) => {  
 const camelKey = isArray(target) ? key : convertToCamelCase(key);  
 result[camelKey] =  
 isObject(value) && !isDate(value) ? camelize(value) : value;  
 });  
function fixCasing(chars) {  
 for (let i = 0; i < chars.length; i++) {  
 if (i === 1 && !isUpperCase(chars[i])) {  
 break;  
 }  
 const hasNext = i + 1 < chars.length;  
 // Stop when the next character is already lowercase.  
 if (i > 0 && hasNext && !isUpperCase(chars[i + 1])) {  
 // If the next character is a space, lowercase the current character before exiting.  
 if (chars[i + 1] === " ") {  
 chars[i] = chars[i].toLowerCase();  
 }  
 break;  
 }  
 chars[i] = chars[i].toLowerCase();  
 }  
}  
function isUpperCase(char) {  
 return char === char.toUpperCase() && char !== char.toLowerCase();  
}  
module.exports = { camelize };

## File: bd-scs-nodejs-frontend/src/utils/signConfig.js

module.exports.signConfig = {  
 A2: {  
 x: 390,  
 y: 200,  
 },  
 B2: {  
 x: 350,  
 y: 305,  
 },  
 C2: {  
 x: 370,  
 y: 295,  
 },  
 A3: {  
 x: 1000,  
 y: 300,  
 },  
 A4: {  
 x: 650,  
 y: 290,  
 },  
 E2: {  
 x: 360,  
 y: 200,  
 },  
 F2: {  
 x: 360,  
 y: 300,  
 },  
 G2: {  
 x: 340,  
 y: 285,  
 },  
 "CCCAI-C2": {  
 x: 150,  
 y: 100,  
 },  
 "CCCAI-MH3": {  
 x: 150,  
 y: 115,  
 },  
 "EDB-JOKC02": {  
 x: 150,  
 y: 100,  
 },  
 "SCS-1": {  
 x: 150,  
 y: 400,  
 page: 3,  
 },  
};

## File: bd-scs-nodejs-frontend/src/utils/Signer.js

import \* as nodeCrypto from 'crypto';  
import \* as asn1js from 'asn1js';  
import \* as pkijs from 'pkijs';  
import {ISigner, SignPdfError} from '@signpdf/utils';  
const oids = {  
 data: '1.2.840.113549.1.7.1',  
 signedData: '1.2.840.113549.1.7.2',  
 contentType: '1.2.840.113549.1.9.3',  
 messageDigest: '1.2.840.113549.1.9.4',  
 signingTime: '1.2.840.113549.1.9.5',  
 signingCertificateV2: '1.2.840.113549.1.9.16.2.47',  
};  
export class Signer extends ISigner {  
 signAlgorithm = 'RSASSA-PKCS1-v1\_5';  
 hashAlgorithm = 'SHA-256';  
 async getSignAlgorithm() {  
 return this.signAlgorithm;  
 }  
 async getHashAlgorithm() {  
 return this.hashAlgorithm;  
 }  
 async getCertificate() {  
 throw new SignPdfError(  
 `getCertificate() is not implemented on ${this.constructor.name}`,  
 SignPdfError.TYPE\_INPUT,  
 );  
 }  
 async getKey() {  
 throw new SignPdfError(  
 `getKey() is not implemented on ${this.constructor.name}`,  
 SignPdfError.TYPE\_INPUT,  
 );  
 }  
 getCrypto() {  
 const crypto = new pkijs.CryptoEngine({name: 'SignerCrypto', crypto: nodeCrypto});  
 return crypto;  
 }  
 async obtainCertificates() {  
 let certBytes = await this.getCertificate();  
 if (!Array.isArray(certBytes)) {  
 certBytes = [certBytes];  
 }  
 return certBytes.map((cb) => pkijs.Certificate.fromBER(cb));  
 }  
 async obtainKey() {  
 const keyBytes = await this.getKey();  
 const algorithmParams = this.crypto.getAlgorithmParameters(this.signAlgorithm, 'importkey').algorithm;  
 return this.crypto.importKey('pkcs8', keyBytes, {  
 name: this.signAlgorithm,  
 ...algorithmParams,  
 hash: {name: this.hashAlgorithm},  
 }, false, ['sign']);  
 }  
 async obtainSignedAttributes(signingTime, data, signCert) {  
 const digest = await this.crypto.digest({name: this.hashAlgorithm}, data);  
 const signedAttrs = [  
 new pkijs.Attribute({  
 type: oids.contentType,  
 values: [new asn1js.ObjectIdentifier({value: oids.data})],  
 }),  
 new pkijs.Attribute({  
 type: oids.signingTime,  
 values: [new asn1js.UTCTime({valueDate: signingTime ?? new Date()})],  
 }),  
 new pkijs.Attribute({  
 type: oids.messageDigest,  
 values: [new asn1js.OctetString({valueHex: digest})],  
 }),  
 ];  
 const hashOid = this.crypto.getOIDByAlgorithm({name: this.hashAlgorithm}, true, 'hashAlgorithm');  
 const signCertHash = await this.crypto.digest(  
 {name: this.hashAlgorithm},  
 signCert.toSchema(true).toBER(false),  
 );  
 const essCertIDv2 = new asn1js.Sequence({  
 value: [  
 new asn1js.Sequence({value: [new asn1js.ObjectIdentifier({value: hashOid})]}),  
 new asn1js.OctetString({valueHex: signCertHash}),  
 ],  
 });  
 const signingCertificateV2 = new asn1js.Sequence({  
 value: [  
 new asn1js.Sequence({value: [essCertIDv2]}),  
 ],  
 });  
 signedAttrs.push(new pkijs.Attribute({  
 type: oids.signingCertificateV2,  
 values: [signingCertificateV2],  
 }));  
 return signedAttrs;  
 }  
 async obtainUnsignedAttributes(signature) {  
 return [];  
 }  
 async sign(pdfBuffer, signingTime = undefined) {  
 if (!(pdfBuffer instanceof Buffer)) {  
 throw new SignPdfError(  
 'PDF expected as Buffer.',  
 SignPdfError.TYPE\_INPUT,  
 );  
 }  
 this.signAlgorithm = await this.getSignAlgorithm();  
 this.hashAlgorithm = await this.getHashAlgorithm();  
 this.crypto = this.getCrypto();  
 const certificates = await this.obtainCertificates();  
 const signCert = certificates[0];  
 const key = await this.obtainKey();  
 const signerInfo = new pkijs.SignerInfo({  
 version: 1,  
 sid: new pkijs.IssuerAndSerialNumber({  
 issuer: signCert.issuer,  
 serialNumber: signCert.serialNumber,  
 }),  
 signedAttrs: new pkijs.SignedAndUnsignedAttributes({  
 type: 0,  
 attributes: await this.obtainSignedAttributes(signingTime, pdfBuffer, signCert),  
 }),  
 });  
 const cmsSignedData = new pkijs.SignedData({  
 version: 1,  
 encapContentInfo: new pkijs.EncapsulatedContentInfo({  
 eContentType: oids.data,  
 }),  
 signerInfos: [signerInfo],  
 certificates,  
 });  
 await cmsSignedData.sign(key, 0, this.hashAlgorithm, undefined, this.crypto);  
 const unsignedAttrs = await this.obtainUnsignedAttributes(signerInfo.signature.getValue());  
 if (unsignedAttrs.length > 0) {  
 signerInfo.unsignedAttrs = new pkijs.SignedAndUnsignedAttributes({  
 type: 1,  
 attributes: unsignedAttrs,  
 });  
 }  
 const cmsContentWrap = new pkijs.ContentInfo({  
 contentType: oids.signedData,  
 content: cmsSignedData.toSchema(true),  
 });  
 return Buffer.from(cmsContentWrap.toSchema().toBER(false));  
 }  
 async verify(cmsSignedBuffer, pdfBuffer) {  
 const cmsContentSimpl = pkijs.ContentInfo.fromBER(cmsSignedBuffer);  
 const cmsSignedSimpl = new pkijs.SignedData({schema: cmsContentSimpl.content});  
 return cmsSignedSimpl.verify({  
 signer: 0,  
 trustedCerts: [],  
 data: pdfBuffer,  
 }, this.getCrypto());  
 }  
}

## File: bd-scs-nodejs-frontend/src/utils/signUtils.js

const fs = require("fs");  
var {  
 PDFDocument,  
 StandardFonts,  
 rgb,  
 CustomFontSubsetEmbedder,  
} = require("pdf-lib");  
var pdflibAddPlaceholder =  
 require("@signpdf/placeholder-pdf-lib").pdflibAddPlaceholder;  
var signpdf = require("@signpdf/signpdf").default;  
var \_utils = require("@signpdf/utils");  
const crypto = require("crypto");  
const fontkit = require("@pdf-lib/fontkit");  
const hexDigits = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"];  
function base64Decode(input) {  
 return Buffer.from(input, "base64");  
}  
function digestToByte(data, algorithm) {  
 const hash = crypto.createHash(algorithm);  
 hash.update(data);  
 return hash.digest();  
}  
function getSignCode(documentHash, hkiCHash, clientID) {  
 const documentHashBytes = base64Decode(documentHash);  
 const hkiCHashBytes = base64Decode(hkiCHash);  
 const clientIDBytes = digestToByte(clientID, "sha512");  
 const source = Buffer.concat([  
 documentHashBytes,  
 hkiCHashBytes,  
 clientIDBytes,  
 ]);  
 const inData = digestToByte(source, "sha512");  
 const digestMD5Data = digestToByte(inData, "md5");  
 const code = [];  
 for (let i = 0; i < 4; i++) {  
 code[i] = hexDigits[((digestMD5Data[i \* 4] >>> 4) & 0xf) % 10];  
 }  
 return code.join("");  
}  
function getHash(bytes) {  
 return crypto.createHash("sha256").update(bytes).digest("base64");  
}  
const signWithPlaceholder = async (  
 filePath,  
 targetPath,  
 placeholder = null,  
 signer = null,  
 visibleSignatures = []  
) => {  
 console.log(filePath);  
 var pdfBuffer = fs.readFileSync(filePath);  
 let pdfDoc = await PDFDocument.load(pdfBuffer);  
 pdfDoc.registerFontkit(fontkit);  
 const fontBytes = await new Promise((resolve) =>  
 fs.readFile("./public/fonts/Microsoft JhengHei Bold.ttf", (err, data) => {  
 if (err) {  
 console.log("Font not loaded", err);  
 resolve(null);  
 } else resolve(data.buffer);  
 })  
 );  
 let customFont;  
 if (fontBytes) {  
 pdfDoc.registerFontkit(fontkit);  
 await pdfDoc.embedFont(fontBytes);  
 customFont = await pdfDoc.embedFont(fontBytes);  
 }  
 for (let visibleSignature of visibleSignatures) {  
 const { x, y, pageNo, fontSize, displayText, maxWidth } = visibleSignature;  
 const pages = pdfDoc.getPages();  
 const firstPage = pages[pageNo];  
 firstPage.drawText(displayText, {  
 x: x,  
 y: y,  
 size: fontSize,  
 font: customFont || (await pdfDoc.embedFont(StandardFonts.Helvetica)),  
 color: rgb(0, 0, 0),  
 maxWidth: maxWidth,  
 });  
 }  
 if (placeholder) {  
 pdflibAddPlaceholder({  
 pdfDoc: pdfDoc,  
 reason: "The user is declaring consent through JavaScript.",  
 contactInfo: "signpdf@example.com",  
 name: "John Doe",  
 location: "Free Text Str., Free World",  
 team: "Signority",  
 byteRangePlaceholder: "\*\*\*\*\*\*\*\*\*\*",  
 ...placeholder,  
 });  
 }  
 if (placeholder || visibleSignature) {  
 pdfBuffer = await pdfDoc.save();  
 }  
 if (signer) {  
 pdfBuffer = await signpdf.sign(pdfBuffer, signer);  
 }  
 if (targetPath) {  
 fs.writeFileSync(targetPath, pdfBuffer);  
 }  
 return pdfBuffer;  
};  
const externalSign = async (pdfBuffer, raw, signingTime = undefined) => {  
 let pdf = (0, \_utils.removeTrailingNewLine)(  
 (0, \_utils.convertBuffer)(pdfBuffer, "PDF")  
 );  
 const { byteRangePlaceholder, byteRangePlaceholderPosition } = (0,  
 \_utils.findByteRange)(pdf);  
 if (!byteRangePlaceholder) {  
 throw new \_utils.SignPdfError(  
 "No ByteRangeStrings found within PDF buffer.",  
 \_utils.SignPdfError.TYPE\_PARSE  
 );  
 }  
 const byteRangeEnd =  
 byteRangePlaceholderPosition + byteRangePlaceholder.length;  
 const contentsTagPos = pdf.indexOf("/Contents ", byteRangeEnd);  
 const placeholderPos = pdf.indexOf("<", contentsTagPos);  
 const placeholderEnd = pdf.indexOf(">", placeholderPos);  
 const placeholderLengthWithBrackets = placeholderEnd + 1 - placeholderPos;  
 const placeholderLength = placeholderLengthWithBrackets - 2;  
 const byteRange = [0, 0, 0, 0];  
 byteRange[1] = placeholderPos;  
 byteRange[2] = byteRange[1] + placeholderLengthWithBrackets;  
 byteRange[3] = pdf.length - byteRange[2];  
 let actualByteRange = `/ByteRange [${byteRange.join(" ")}]`;  
 actualByteRange += " ".repeat(  
 byteRangePlaceholder.length - actualByteRange.length  
 );  
 pdf = Buffer.concat([  
 pdf.slice(0, byteRangePlaceholderPosition),  
 Buffer.from(actualByteRange),  
 pdf.slice(byteRangeEnd),  
 ]);  
 pdf = Buffer.concat([  
 pdf.slice(0, byteRange[1]),  
 pdf.slice(byteRange[2], byteRange[2] + byteRange[3]),  
 ]);  
 if (raw.length \* 2 > placeholderLength) {  
 throw new \_utils.SignPdfError(  
 `Signature exceeds placeholder length: ${  
 raw.length \* 2  
 } > ${placeholderLength}`,  
 \_utils.SignPdfError.TYPE\_INPUT  
 );  
 }  
 let signature = Buffer.from(raw, "binary").toString("hex");  
 this.lastSignature = signature;  
 signature += Buffer.from(  
 String.fromCharCode(0).repeat(placeholderLength / 2 - raw.length)  
 ).toString("hex");  
 pdf = Buffer.concat([  
 pdf.slice(0, byteRange[1]),  
 Buffer.from(`<${signature}>`),  
 pdf.slice(byteRange[1]),  
 ]);  
 return pdf;  
};  
const prepareDocDigest = async (pdfBuffer) => {  
 let pdf = (0, \_utils.removeTrailingNewLine)(  
 (0, \_utils.convertBuffer)(pdfBuffer, "PDF")  
 );  
 const { byteRangePlaceholder, byteRangePlaceholderPosition } = (0,  
 \_utils.findByteRange)(pdf);  
 if (!byteRangePlaceholder) {  
 throw new \_utils.SignPdfError(  
 "No ByteRangeStrings found within PDF buffer.",  
 \_utils.SignPdfError.TYPE\_PARSE  
 );  
 }  
 const byteRangeEnd =  
 byteRangePlaceholderPosition + byteRangePlaceholder.length;  
 const contentsTagPos = pdf.indexOf("/Contents ", byteRangeEnd);  
 const placeholderPos = pdf.indexOf("<", contentsTagPos);  
 const placeholderEnd = pdf.indexOf(">", placeholderPos);  
 const placeholderLengthWithBrackets = placeholderEnd + 1 - placeholderPos;  
 const placeholderLength = placeholderLengthWithBrackets - 2;  
 const byteRange = [0, 0, 0, 0];  
 byteRange[1] = placeholderPos;  
 byteRange[2] = byteRange[1] + placeholderLengthWithBrackets;  
 byteRange[3] = pdf.length - byteRange[2];  
 let actualByteRange = `/ByteRange [${byteRange.join(" ")}]`;  
 actualByteRange += " ".repeat(  
 byteRangePlaceholder.length - actualByteRange.length  
 );  
 pdf = Buffer.concat([  
 pdf.slice(0, byteRangePlaceholderPosition),  
 Buffer.from(actualByteRange),  
 pdf.slice(byteRangeEnd),  
 ]);  
 pdf = Buffer.concat([  
 pdf.slice(0, byteRange[1]),  
 pdf.slice(byteRange[2], byteRange[2] + byteRange[3]),  
 ]);  
 return getHash(pdf);  
};  
module.exports = {  
 signWithPlaceholder,  
 externalSign,  
 prepareDocDigest,  
 getSignCode,  
};

## File: bd-scs-nodejs-frontend/src/app.js

const express = require("express");  
const app = express();  
var createError = require("http-errors");  
var path = require("path");  
var cors = require("cors");  
require("dotenv").config();  
const port = process.env.PORT || 7001;  
const sequelize = require("./config/database");  
sequelize.authenticate().then(() => {  
 sequelize.sync({}).then(() => {  
 console.log("Tables synced");  
 });  
 console.log("Connection has been established successfully.");  
});  
app.use(  
 cors({  
 origin: true,  
 })  
);  
app.use(express.json());  
app.use(express.urlencoded({ extended: false }));  
app.use((process.env.ROUTER\_ROOT ?? "") + "/application", require("./routes/ApplicationController"));  
app.use((process.env.ROUTER\_ROOT ?? "") + "/auth", require("./routes/AuthController"));  
app.use((process.env.ROUTER\_ROOT ?? "") + "/esign", require("./routes/ESignController"));  
app.get(process.env.ROUTER\_ROOT ?? "/", (req, res) => {  
 res.send("Hello World!");  
});  
app.listen(port, () => {  
 console.log(`Example app listening at http://localhost:${port}`);  
});

## File: bd-scs-nodejs-frontend/package.json

{  
 "name": "scs-nodejs-frontend",  
 "version": "1.0.0",  
 "description": "",  
 "main": "index.js",  
 "scripts": {  
 "test": "echo \"Error: no test specified\" && exit 1",  
 "dev": "node src/app.js"  
 },  
 "author": "",  
 "license": "ISC",  
 "dependencies": {  
 "@aws-sdk/client-ses": "^3.714.0",  
 "@pdf-lib/fontkit": "^1.1.1",  
 "@signpdf/placeholder-pdf-lib": "^3.2.4",  
 "@signpdf/signer-p12": "^3.2.4",  
 "@signpdf/signpdf": "^3.2.4",  
 "axios": "^1.7.5",  
 "cors": "^2.8.5",  
 "csv-parser": "^3.1.0",  
 "dotenv": "^16.4.5",  
 "express": "^4.19.2",  
 "lodash": "^4.17.21",  
 "mariadb": "^3.4.0",  
 "moment": "^2.30.1",  
 "multer": "^1.4.5-lts.1",  
 "node-forge": "^1.3.1",  
 "node-rsa": "^1.1.1",  
 "node-zip": "^1.1.1",  
 "nodemailer": "^6.9.16",  
 "pdf-lib": "^1.17.1",  
 "pdf-signatures": "^0.1.0",  
 "pdfjs-dist": "^4.10.38",  
 "pg": "^8.13.3",  
 "pg-hstore": "^2.3.4",  
 "redis": "^4.7.0",  
 "sequelize": "^6.37.3",  
 "sequelize-cli": "^6.6.2",  
 "tedious": "^18.6.2",  
 "uuid": "^10.0.0"  
 },  
 "devDependencies": {  
 "nodemon": "^3.1.4"  
 }  
}