Detailed Design

for

Design and Implementation of a Lightweight Education Data Bay Area (E-DBA)

Version 1.0 Approved

Prepared by:
Taian Yu (FULL)   Yolanda Yin (FULL)
Justin Zhang (FULL)   Aaron Liu (FULL)
Steven Luo (FULL)   Younger Yang (FULL)

Group C01

April 9, 2025

Contents

Revision History

Name	Date	Reason For Changes	Version
All Members	9 April	Create the document	1.0

Chapter 1

Overview

1.1 Project Description

This project focuses on designing the architecture for five core subsystems: User, Restructure, Service, Others, and Data. Each subsystem will have its own architecture, UI layers, and persistent data tables to ensure scalability and organization. The design aims to minimize interdependencies between subsystems, allowing for independent development and seamless integration.

Detailed design documentation will include use case diagrams, class diagrams, and explicit system component relationships to ensure autonomous operation and integration between subsystems.

1.2 References

- Software Requirements Specification (SRS)
- UI Design Document
- Architecture Design Document

1.3 Design Purpose

The design aims to establish a modular structure enhancing maintainability, scalability, and separation of concerns across subsystems. Dedicated subsystems for different architectural layers and data tables minimize associations across boundaries, ensuring efficient data flow and simplified future extensions.

Chapter 2

Overall description

2.1 Class diagram

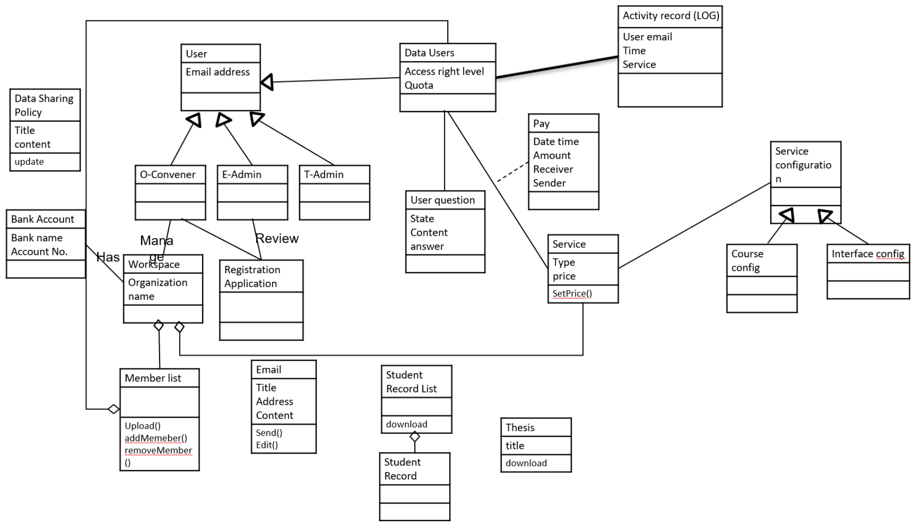


Figure 2.1: Class diagram

2.2 Refinements

Splitting overly complex classes, merging similarly functioning classes, and adjusting inheritance and association relationships between classes.

- **Identifying features:** Clarify the core functionality and properties of each class. By analyzing the requirements and functional modules, we identify the key characteristics of each class and translate them into specific properties and methods.
- **Specifying Visibility and Constraints:** We set the appropriate level of access (such as public, private, or protected) for the properties and methods of each class to ensure the encapsulation and security of the data.
- **Constraints between classes:** For example, polymorphism through interfaces, or inheritance of classes to restrict how certain behaviors can be implemented.

Chapter 3

Detailed design

3.1 Class diagram

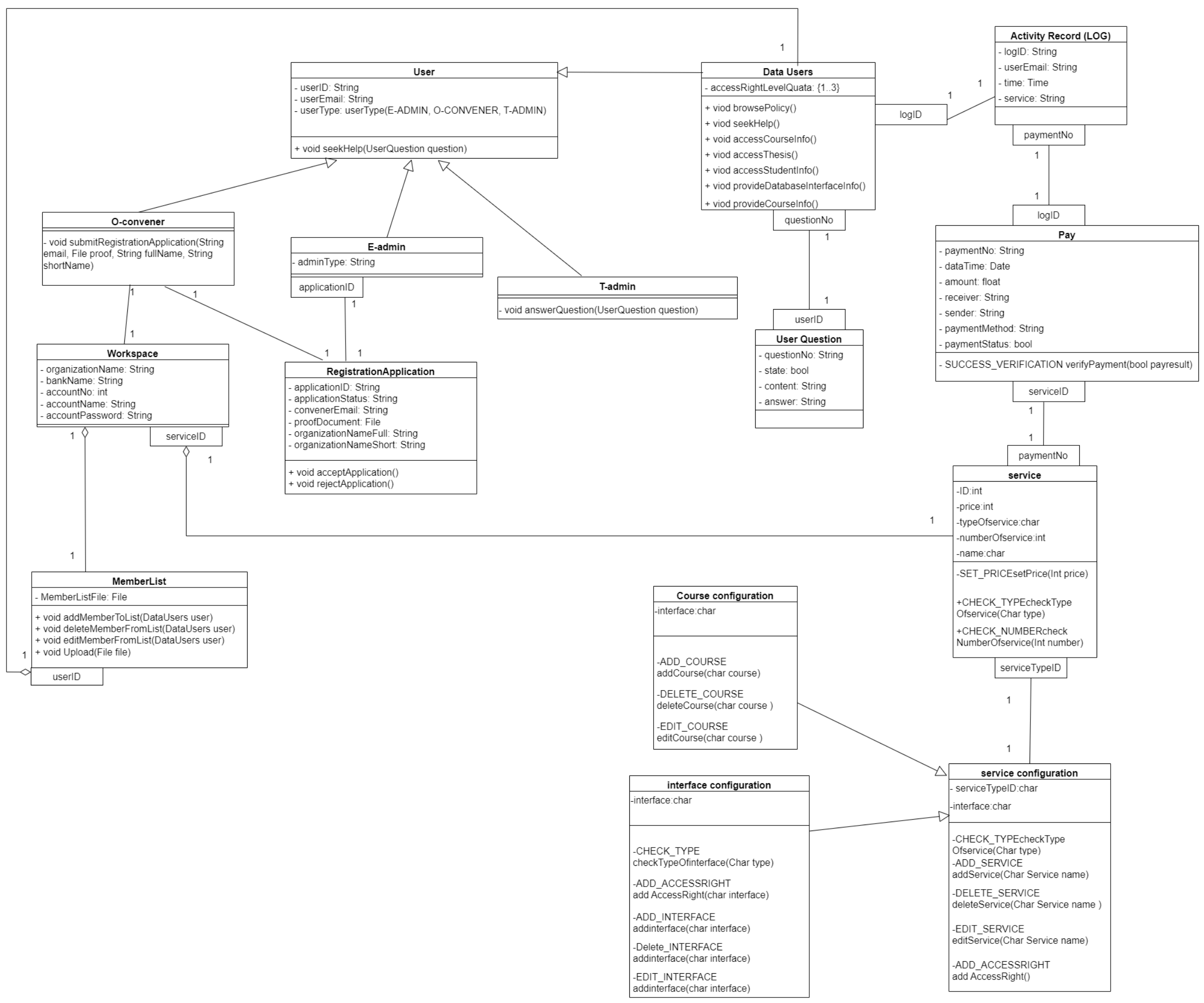

User
- userID: String
- userEmail: String
- userType: userType(E-ADMIN, O-CONVENER, T-ADMIN)
+ void seekHelp(UserQuestion question)
Data Users
- accessRightLevelQuata: {1..3}
+ viod browsePolicy()
+ viod seekHelp()
+ void accessCourseInfo()
+ void accessThesis()
+ void accessStudentInfo()
+ viod provideDatabaseInterfaceInfo()
+ viod provideCourseInfo()
Activity Record (LOG)
- logID: String
- userEmail: String
- time: Time
- service: String
logID
paymentNo
questionNo
userID
User Question
- questionNo: String
- state: bool
- content: String
- answer: String
Pay
- paymentNo: String
- dataTime: Date
- amount: float
- receiver: String
- sender: String
- paymentMethod: String
- paymentStatus: bool
- SUCCESS_VERIFICATION verifyPayment(bool payresult)
serviceID
O-convener
- void submitRegistrationApplication(String email, File proof, String fullName, String shortName)
E-admin
- adminType: String
applicationID
T-admin
- void answerQuestion(UserQuestion question)
Workspace
- organizationName: String
- bankName: String
- accountNo: int
- accountName: String
- accountPassword: String
RegistrationApplication
- applicationID: String
- applicationStatus: String
- convenerEmail: String
- proofDocument: File
- organizationNameFull: String
- organizationNameShort: String
+ void acceptApplication()
+ void rejectApplication()
service
-ID:int
-price:int
-typeOfservice:char
-numberOfservice:int
-name:char
-SET_PRICEsetPrice(Int price)
+CHECK_TYPEcheckType Ofservice(Char type)
+CHECK_NUMBERcheck NumberOfservice(Int number)
serviceTypeID
MemberList
- MemberListFile: File
+ void addMemberToList(DataUsers user)
+ void deleteMemberFromList(DataUsers user)
+ void editMemberFromList(DataUsers user)
+ void Upload(File file)
Course configuration
-interface:char
-ADD_COURSE addCourse(char course)
-DELETE_COURSE deleteCourse(char course)
-EDIT_COURSE editCourse(char course)
interface configuration
-interface:char
-CHECK_TYPE checkTypeOfinterface(Char type)
-ADD_ACCESSRIGHT add AccessRight(char interface)
-ADD_INTERFACE addinterface(char interface)
-Delete_INTERFACE addinterface(char interface)
-EDIT_INTERFACE addinterface(char interface)
service configuration
- serviceTypeID:char
-interface:char
-CHECK_TYPEcheckType Ofservice(Char type)
-ADD_SERVICE addService(Char Service name)
-DELETE_SERVICE deleteService(Char Service name)
-EDIT_SERVICE editService(Char Service name)
-ADD_ACCESSRIGHT add AccessRight()
1

Figure 3.1: Restructure Class

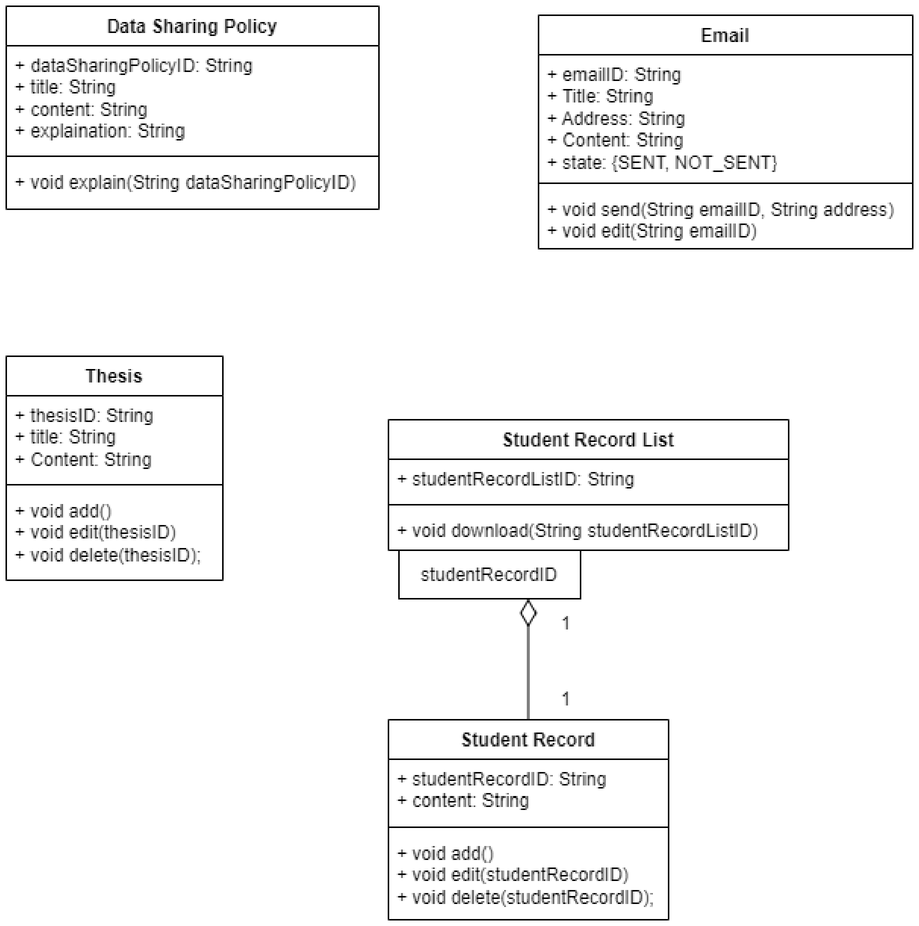


Figure 3.2: Restructure Class

Compared to the previous class diagram, the following relationships have been added first:

- O-convener and Workspace are a one-to-one relationship.
- O-convener and Registration Application are a one-to-one relationship.
- E-Admin and Registration Application are a one-to-many relationship.
- Workspace and Member List are a one-to-one relationship.
- Workspace and Service are a one-to-many relationship.
- Member List and Data Users are a one-to-many relationship.
- Data Users and User Questions are a many-to-many relationship.
- Data Users and Activity Records are a one-to-many relationship.
- Service and Service Configuration are a one-to-many relationship.
- Implement an association Class Pay as a class. Among them, Data Users and Pay are a many-to-many relationship, while Pay and Service are a many-to-many relationship.

After reconstruction, the relationship changes as follows:

- Collapse the BankAccount class into attributes of Workspace.
- The following relationships use Use Qualifier to reduce multiplicity:
  - E-Admin and Registration Application
  - Member List and Data Users
  - Workspace and Service
  - Data Users and User Questions
  - Data Users and Activity Records
  - Service and Service Configuration
  - Data Users and Pay
  - Pay and Service

3.2 User

3.2.1 User

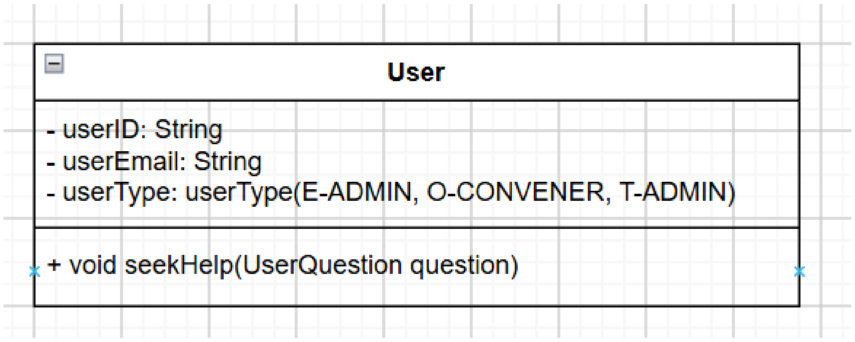

User
- userID: String
- userEmail: String
- userType: userType(E-ADMIN, O-CONVENER, T-ADMIN)
+ void seekHelp(UserQuestion question)

Figure 3.3: User

Explanations

- The additional attribute `userType: char` determines the role of the user:
  - `E-admin` (Event Administrator)
  - `O-convener` (Organizing Convener)
  - `T-admin` (Technical Administrator)
- The operation `seekHelp()` allows users to post technical help requests to `T-admins`.

Constraints
TBD

3.2.2 T-Admin

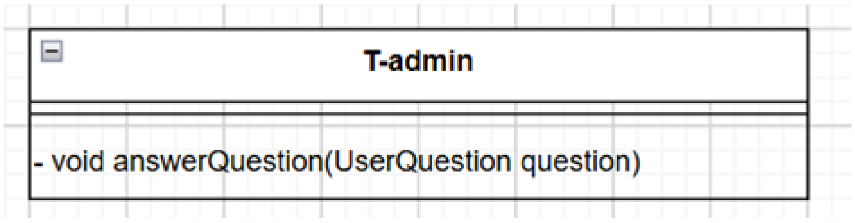


Figure 3.4: T-Admin

Explanations

- The operation `answerQuestion()` allows the `T-admin` to answer help requests.

Constraints
TBD

3.2.3 E-Admin

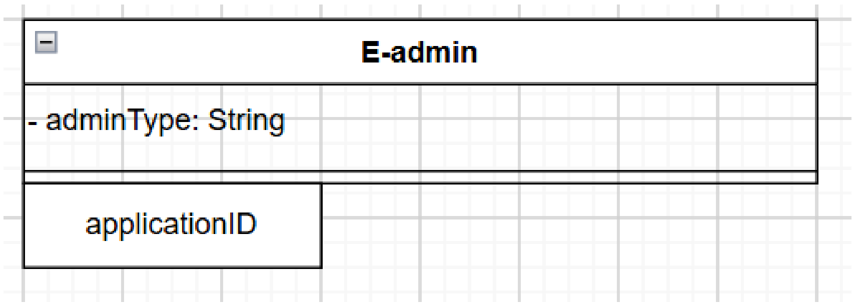


Figure 3.5: E-Admin

Explanations

- The attribute `adminType: String` indicates whether the user is a(n) `E-admin` or `Senior E-admin`.

Constraints
TBD

3.2.4 O-Convener

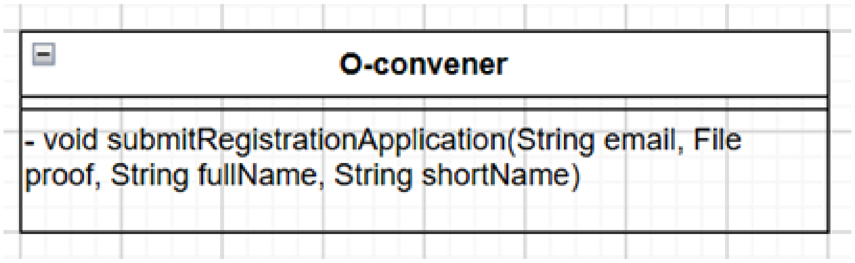

O-convener
- void submitRegistrationApplication(String email, File proof, String fullName, String shortName)

Figure 3.6: E-Admin

Explanations

- For the `submitRegistrationApplication` method:
  - **Pre-condition:**
    - email, proof, fullName, and shortName cannot be null
    - email must be in a valid email format
    - proof must be in a valid PDF format
  - **Post-condition:**
    - If email is already registered, return `EMAIL˙ALREADY˙REGISTERED`
    - Otherwise, a new registration application is created and sent to the E-admin for review, return `SUBMISSION˙SUCCESSFUL`

Constraints
TBD

3.2.5 Workspace

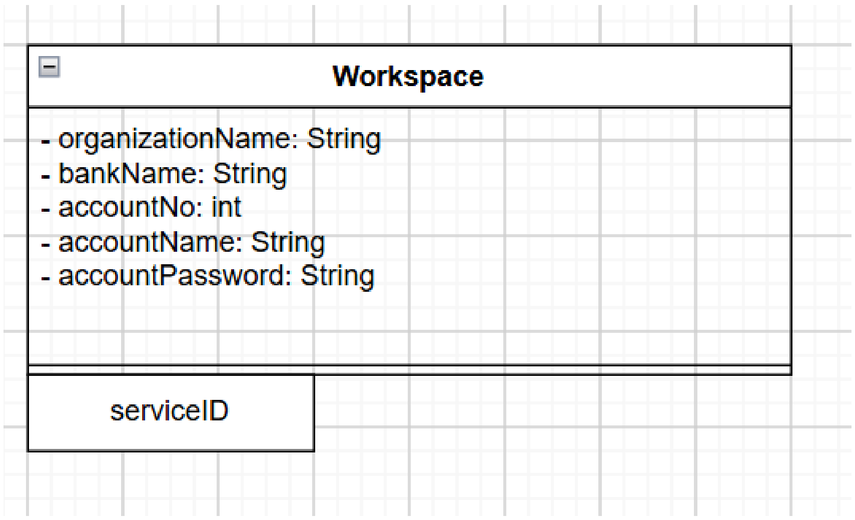


Figure 3.7: E-Admin

Explanations

- The attribute bankName indicates the name from where the account was opened.
- The attribute `accountNo` indicates the account ID.
- The attribute `accountName` indicates the name the account belongs to.

Constraints
TBD

3.2.6 Registration Application

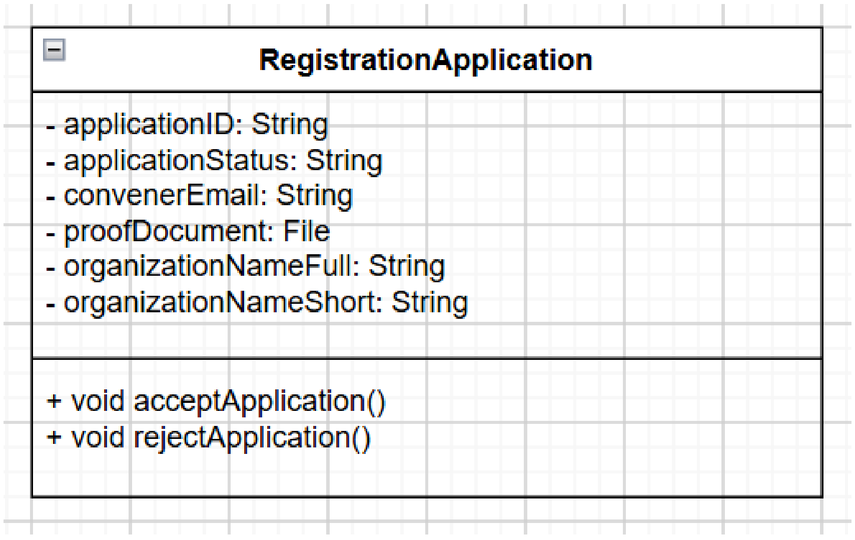

RegistrationApplication
- applicationID: String
- applicationStatus: String
- convenerEmail: String
- proofDocument: File
- organizationNameFull: String
- organizationNameShort: String
+ void acceptApplication()
+ void rejectApplication()

Figure 3.8: E-Admin

Explanations

- The additional attribute `applicationStatus: String` indicates whether the application passes the review of both the E-admin and Senior-admin.
- The attribute `convenerEmail` is required by E-DBA for contacting and notifying purposes.
- The attribute `proofDocument` is a PDF file as proofing documentation.
- The operation `acceptApplication()` changes `applicationStatus`.
- The operation `rejectApplication()` changes `applicationStatus`.

Constraints
TBD

3.2.7 Member List

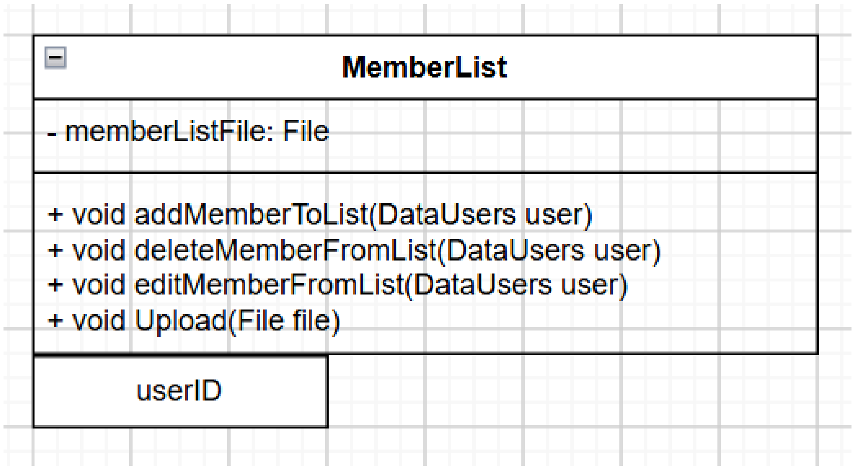

MemberList
- memberListFile: File
+ void addMemberToList(DataUsers user)
+ void deleteMemberFromList(DataUsers user)
+ void editMemberFromList(DataUsers user)
+ void Upload(File file)
userID

Figure 3.9: E-Admin

Explanations

- The additional attribute `memberListFile: File` is an Excel format file that includes the list of members in the organization, including information about name, email, access rights, and quota limits.

Constraints
TBD

3.3 Service

3.3.1 Service

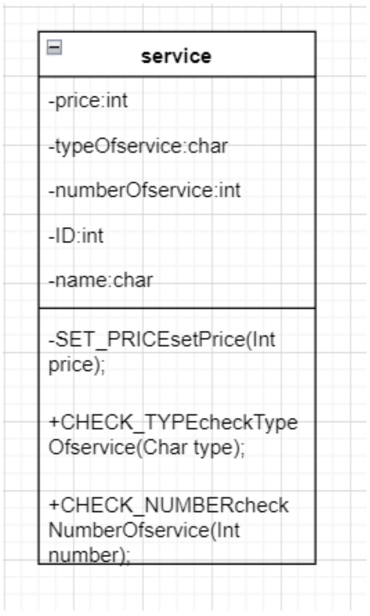

service
-price:int
-typeOfservice:char
-numberOfservice:int
-ID:int
-name:char
-SET_PRICEsetPrice(Int price);
+CHECK_TYPEcheckTypeOfservice(Char type);
+CHECK_NUMBERcheckNumberOfservice(Int number);

Figure 3.10: Service

Explanations

- `- price:` `int` is the price of the service
- `- typeOfservice:` `char` is the type of the service
- `- numberOfservice:` `int` is the number of the service
- `- ID:` `int` is the ID of the service
- `- name:` `char` is the name of the service
- `+ SET˙PRICEsetPrice(int price)` can let the admin set the price of the service
- `+ CHECK˙TYPEcheckTypeOfservice(char type)` can let the admin and user see the different types of service
- `+ CHECK˙NUMBERcheckNumberOfservice(int number)` can let the admin and user see the number of services

Constraints

- `- SET˙PRICEsetPrice(int price)` is the pre-condition. The price must be more than 0.

3.3.2 Service Configuration

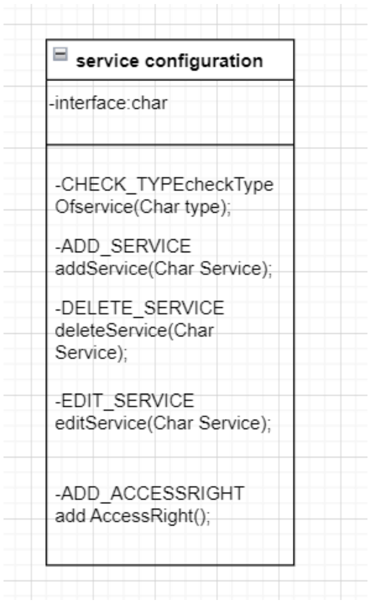

service configuration
-interface:char
-CHECK_TYPEcheckTypeOfservice(Char type);
-ADD_SERVICE addService(Char Service);
-DELETE_SERVICE deleteService(Char Service);
-EDIT_SERVICE editService(Char Service);
-ADD_ACCESSRIGHT add AccessRight();

Figure 3.11: Service Configuration

Explanations

- `- interface:` `char` is the interface of service configuration
- `- CHECK˙TYPEcheckTypeOfservice(char type)` is to check the type of service that needs to be configured

- `- ADD˙SERVICEaddService(char Service name)` is to add a new service that needs to be configured
- `- DELETE˙SERVICEdeleteService(char Service name)` is to delete the service
- `- EDIT˙SERVICEeditService(char Service name)` is to edit the service that needs to be configured
- `- ADD˙ACCESSRIGHTaddAccessRight()` is to add the right for users to access the service

Constraints
TBD

3.3.3 Course Configuration

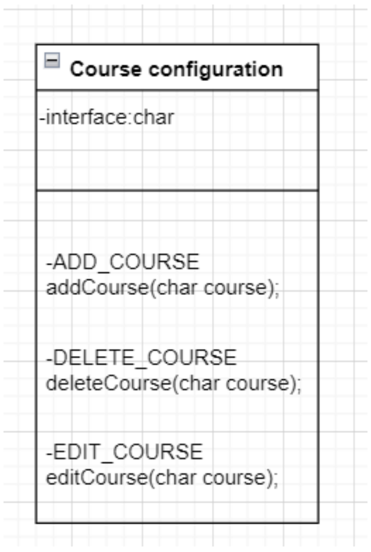


Figure 3.12: Course Configuration

Explanations

- `- interface:` char is the interface of course configuration
- `- ADD˙COURSEaddCourse(char Course)` is to add a new course that needs to be configured
- `- DELETE˙COURSEdeleteCourse(char Course)` is to delete the course
- `- EDIT˙COURSEeditCourse(char Course)` is to edit the course that needs to be configured

Constraints
TBD

3.3.4 Interface Configuration

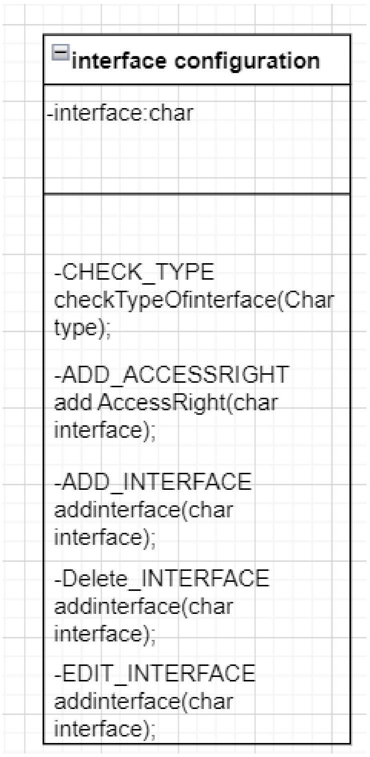

interface configuration
-interface:char
-CHECK_TYPE checkTypeOfinterface(Char type);
-ADD_ACCESSRIGHT add AccessRight(char interface);
-ADD_INTERFACE addinterface(char interface);
-Delete_INTERFACE addinterface(char interface);
-EDIT_INTERFACE addinterface(char interface);

Figure 3.13: Interface Configuration

Explanations

- `- interface: char` is the interface of interface configuration
- `- CHECK˙TYPEcheckTypeOfinterface(char type)` is to check the type of the interface
- `- ADD˙INTERFACEaddInterface(char interface)` is to add a new interface that needs to be configured
- `- DELETE˙INTERFACEdeleteInterface(char interface)` is to delete the interface
- `- EDIT˙INTERFACEeditInterface(char interface)` is to edit the interface that needs to be configured
- `- ADD˙ACCESSRIGHTaddAccessRight()` is to add the right for users to access the interface

Constraints
TBD

3.4 Data

3.4.1 Data User

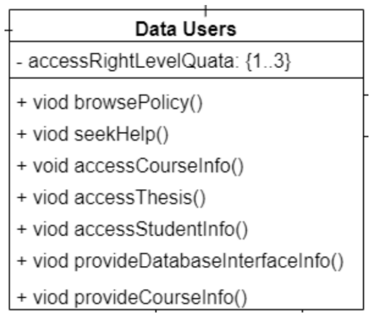

Data Users
- accessRightLevelQuata: {1..3}
+ viod browsePolicy()
+ viod seekHelp()
+ void accessCourseInfo()
+ viod accessThesis()
+ viod accessStudentInfo()
+ viod provideDatabaseInterfaceInfo()
+ viod provideCourseInfo()

Figure 3.14: Data User

Explanations

- Access Right Level Quota has 3 levels:
  - Public data access (access right level: 1)
  - Private data consumption (access right level: 2)
  - Private data provision (access right level: 3)
- `browsePolicy()`, `seekHelp()`, `accessCourseInfo()`, `accessThesis()`, `accessStudentInfo()`, `provideDatabaseInterfaceInfo()`, `provideCourseInfo()` are the functions for data users.

Constraints

- For the `provideDatabaseInterfaceInfo` method:
  - **Pre-condition:** The user should be the data user.
  - **Post-condition:** Open Database Info Interface.

3.4.2 User Question

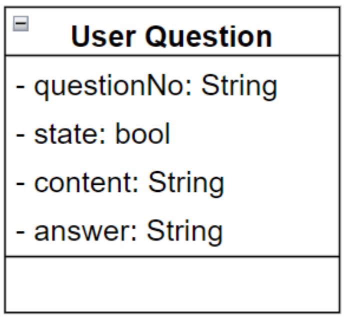


Figure 3.15: User Question

Explanations
QuestionNo is the ID of the question for data user to check.

Constraints
None

3.4.3 Activity Record (LOG)

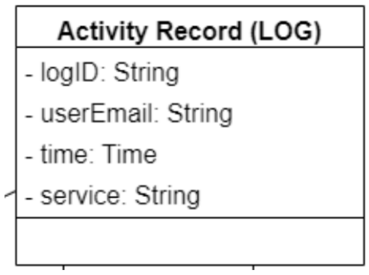


Figure 3.16: Activity Record (LOG)

Explanations
logID is the ID of the activity record for data user to check.

Constraints
None

3.5 Others

3.5.1 Email class

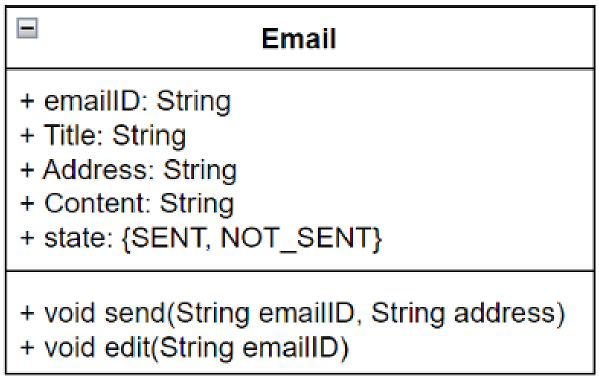

Email
+ emailID: String
+ Title: String
+ Address: String
+ Content: String
+ state: {SENT, NOT_SENT}
+ void send(String emailID, String address)
+ void edit(String emailID)

Figure 3.17: Email class

Explanations

- An email contains its **emailID**, title, address and content, while the state attribute of Email class represents whether the email is sent or not.
- The **send()** operation would allow users to send the email with corresponding **emailID** to the corresponding address, while the **edit()** operation would

allow users to edit the email with corresponding **emailID**.

Constraints

- For the send method:
  - **Pre-condition:**
    - state must be NOT˙SENT
    - address and content must not be null
    - address must be in a valid format
    - the length of title must not be longer than the maximum length allowed
  - **Post-condition:**
    - If the state is SENT, return EMAIL˙ALREADY˙SENT
    - If the content is null, return CONTENT˙CANNOT˙BE˙NULL
    - If the address does not exist, return ADDRESS˙DOES˙NOT˙EXIST
    - If the length of title is longer than the maximum length allowed, return TITLE˙SHOULD˙NOT˙LONGER˙THAN˙MAXIMUM
    - Otherwise, a new email is sent and return SEND˙SUCCESSFUL

3.5.2 Data Sharing Policy class

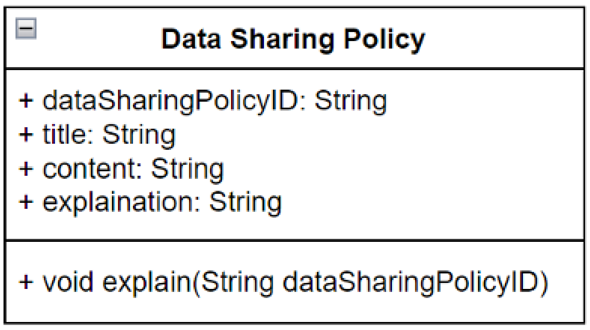


Figure 3.18: Data Sharing Policy Class

Explanations

An data sharing policy contains **dataSharingPolicyID**, the title and content of the data sharing policy, while the **explanation** attribute of **DataSharingPolicy** class represents how E-Admin or Senior E-Admin explain the corresponding data sharing policy.

The **explain()** operation would let E-Admins and Senior E-Admins explain the data sharing policy with the corresponding **dataSharingPolicyID**.

Constraints
TBD

3.5.3 Thesis class

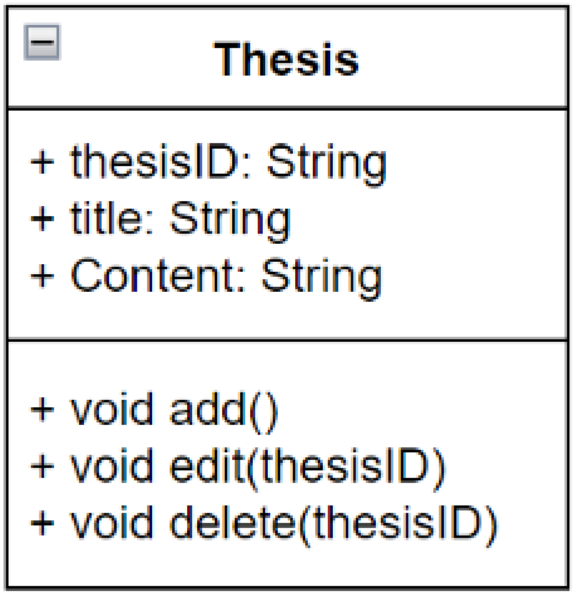


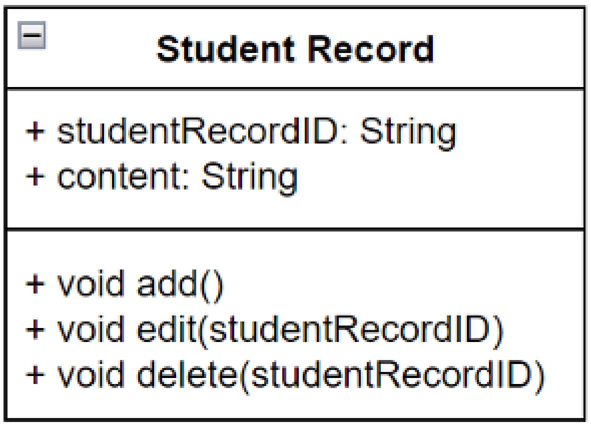
Figure 3.19: Thesis class

Explanations
The `Thesis` class contains three attributes: the **thesisID**, the **title**, and the `content` of the thesis.

The **add()** operation would add a new thesis to the system, while the **edit()** and **delete()** operations could edit and delete the thesis with the corresponding **thesisID**, respectively.

Constraints
TBD

3.5.4 Student Record class



Student Record
+ studentRecordID: String
+ content: String
+ void add()
+ void edit(studentRecordID)
+ void delete(studentRecordID)

Figure 3.20: Student Record class

Explanations
The `Student Record` class contains two attributes: the **studentRecordID** and the **content** of the student record.

The **add()** operation would add a new student record to the system, while the **edit()** and **delete()** operations could edit and delete the record with the corresponding **studentRecordID**, respectively.

Constraints
TBD

3.5.5 Student Record List class

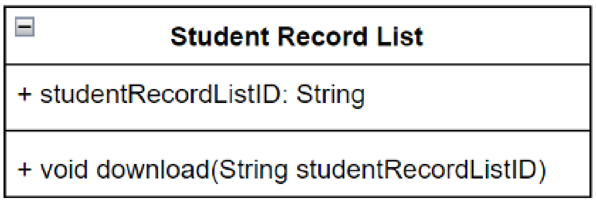

Student Record List
+ studentRecordListID: String
+ void download(String studentRecordListID)

Figure 3.21: Student Record List class

Explanations
The `StudentRecordList` class only has one attribute: the **studentRecordListID**, while one student record list contains one or more student records.

The **download()** operation would allow users to download the student record list from the system directly.

Constraints
TBD

Chapter 4

Alternative detailed design (Optional)

None

Chapter 5

More considerations

None