HLDs/LLD

1. Assessed Revamp (Subjective Test Assessment) FE HLD/LLD: https://docs.google.com/document/d/14qT_qVquR7Ip594PbTGBbjKBytBY7LhU3VSRe7ghE6s/edit#
2. Quiz Ops (Objective Assessment) FE HLD/LLD: https://docs.google.com/document/d/1eVnjDeSsb7CwlGvguWENSfNCy1AoNDGwNQDs5x6jM-g/edit **(file deleted)**

POCs

1. Subjective Questions in QUIZ-SDK: POCs: https://docs.google.com/document/d/1y1yM02FfL05ee7CZ4BsO_zeCf89DZyovNfEaBcSRPmQ/edit
2. Auto Print Failure Analysis**:** https://docs.google.com/document/d/1BZGxB-4RZLNvPdKxfrSZqmIO1w8p4U6N4EMuetsr-bI/edit
3. Objective Quiz Assessment - Integration: https://docs.google.com/document/d/1mnNQ6jM3BlW0UI1rgtWMqGR_VNDBAQSyTTkrOE75DcY/edit#heading=h.5jsnmk1xkyw0
4. Editor: https://docs.google.com/spreadsheets/d/1ZKxxM6YQgMHq8SV7cm-gG5rhJK7Md5X8XNqQUgfJMCs/edit#gid=0 **(file deleted)**
5. Applet: https://docs.google.com/document/d/1wNqbqfsMMCm3mDPbSEqaycMv3zzt9cf2WE4DzH4eWwg/edit?usp=sharing **(file deleted)**

Deliverables / Impact / Legacy system support / New system

Quiz

1. **Setup Quiz Ops repo from scratch** - next.js, react.js, redux, tinyMCE Editor, swagger
2. Quiz/Question List - Filters - Configurable UI
3. Quiz Creation - Multiple Tabs, multiple forms, store form data in redux and fetch data form redux
4. Quiz Configuration
5. Quiz Type
6. **tinyMCE Editor** for creating question
    a. **IMPACT:** did comparison between all available text editor for their features, pricing, support and finalized TinyMCE Editor free version, and added many custom plugins in it to support features which comes with the paid version e.g. music annotation, fill in the blanks, image prePaste control which helps company **save lots of money**
7. Consolidated Question Bank (CQB)

8. Fixed **Image copy paste converting into base64 string** issue in tinyMCE editor.
    a. **IMPACT: Teacher was very comfortable with copy & pasting image** from their computer or internet, but earlier when pasting image in the editor, image get stored as base64 string in the state because of which payload for the save api becomes too long, hence api use to fail, so I fixed it by uploading the pasted image on s3 and storing that uploaded image link in state, **so it makes it very easy & natural for teacher to create question supporting copy pasting image.**
9. **Refactor Quiz SDK Init** Code
    a. **IMPACT:** in the Quiz SDK, lots of things used to happen in a single file. **That file has become so complex and was not easily readable and understandable**. I was given a task to refactor it. The tech lead suggested that I just organize the code, comment on the logic and task of function and remove repeated code. I proposed a different solution, the solution was good but it can put the complete SDK at risk because it was the entry file of the project because I was going to divide the **single file in 5 files/layers, main, error, api, socket, view**. I implemented my solution and hence the entry code became very organized, readable, and easy to understand. **Therefore impacted many developers to do their task easily & quickly because of easy to understand & organized code.**

Assessed

 # Uploader:

1. Filter on Dashboard V1
    a. IMPACT: user is now able to search for the required/needed bundle easily. **It saves the user's time.**
2. Many other small issues

 # Image Server

1. Remarks & Error Type of Assessment and showing these details on Uploader (FE+BE)
2. Grouping from dropdown instead of drag-and-drop
3. Displaying **Offline Assessment Status Colour Coding** on Teacher Dashboard according to the status of the assessment on image server (FE+BE)
    a. **IMPACT**: Teacher is now able to see the status of the assessment from the dashboard itself instead of opening the assessment. **It saves the teacher's time.**
4. Auto Print Failure Analysis
5. Many other small issues

 # Super Admin Dashboard

1. Upload/Download Batches/Students/Teachers

 # School Admin Dashboard

1. Upload Students

 # Teacher Dashboard

1. **Dashboard Revamp** - Filter, Assessment Status Colour Coding, Segmentation Error Modal + Color Representation in Correction, etc (FE+BE)
    a. **IMPACT:** Teacher now directly lands on the correction assessments list and is now able to search for the desired one easily & also see the status of the correction for which page wise correction is needed on the assessment dashboard itself. **It saves the teacher's time & click.**
2. **Common Tutor Pool**
    a. **IMPACT:** Teacher can directly land on the correction screen without worrying about login & searching for the needed assessment. **It is more convenient for the teacher and also saves their time.**
3. **Subjective Test**
4. Auto Print in Constrained Paper Development Support to Somu
    a. IMPACT: Reduced number of nd students because students now do not need to write their details on the paper sheet as it is already written in their question paper.
5. Auto Space/Line in Constrained Paper Development Support to Indrajit
6. **Image Fit Issue** on Constrained Paper
    a. **IMPACT:** Teacher was not able to print the constrained sheet in which there was any question which contains any image that is not able to fit in the A4 paper. I fixed this issue by reducing the size of that image till it fits in the A4 paper. **Now Teacher is able to print the question paper.**
7. Constrained Paper Print issue on latest version of chrome
    a. **IMPACT:** Teacher was not able to print the constrained paper correctly in the latest version of chrome. I fixed the issue. **Teacher is now able to print it in all the version of chrome**
8. Many other small issues

 # Student Dashboard

1. Student Report Issues

Assessed Revamp

 # SDK, Demo & Admin Panel

1. Setup **subjective-assessment-sdk repo** from **scratch** - react, vite, tailwindcss, swagger
    a. **IMPACT:** Did bundle splitting for different sdks, deprioritized analytics/monitoring script. **Hence impacting user experience, network call and bundle size.**
2. Setup **subjective-assessment-demo repo** from **scratch** - react, cra
3. Setup **subjective-assessment-admin-ui repo** from **scratch** - next.js, react.js, tailwindcss, redux, swagger

4. Created **Custom Image Annotation Component** which does not requires any external react package/library
    a. **Legacy System Support:** It also supports old legacy system annotations
    b. **IMPACT:** We didn't use any external library for the image annotation feature. **It gives us full control & understanding of the code and will be easily scalable and extensible**.
5. **Setup code structure** to be used in all sdks.
    a. IMPACT: It makes code very organized, structured and readable thus impacting all developers.
6. Worked on **Correction SDK**
    a. Question Wise Correction
    b. Student Wise Correction
    c. **OMR Correction**
7. Guided **Tanmaya** in completing rest of the **Correction SDK**
8. Guided **Amritpal** in developing **Upload SDK**
9. Guided **Amritpal & Dhairya** in developing **Report SDK**
10. Guided **Dhairya** in developing Admin Panel

 # Admin Correction / Image Server (Legacy System)

1. Converted it from **monolith** (Frontend + Backend) to **Frontend** Project
    a. **Legacy System Support:** We are able to use old codebase
    b. IMPACT: It takes less infra cost, **hence helping organization reduce cost.**
2. Removed unused code/apis/routes/backend code/files
3. **Refactor** all files where any api is getting called and its response is used
4. Added **swagger-client** to use new apis
5. **Integrated new apis**
6. Fixed issues coming during integration & testing

 # Subjective Questions in Quiz SDK

1. Did POC for Image Capturing & Image Editing & image storage memory limit in browser
    a. **IMPACT:** Same sdk can be used to conduct subjective quizzes too. It reduces many resources like developers, infra, etc
2. Created **Image Capture** Component
3. Created **Image Editor** Component
4. Guided **Amritpal** in completing the task

 # Deployment

1. **Demo:** Did code modification to support the **same branch codebase** in all three environments (dev, stage, prod) & deployed it.
    a. **IMPACT: We don't need to maintain different branch/codebase for different environments and also single domain/url can be used for all environments.**

2. **Admin Panel:** Deployed in all the environments.

Post go-live learnings / Operational Excellence / Triaging / RCA driving

 # Post go-live learnings

1. Feature/design/behavior should be very natural / common / general for the user. It should be the same where/how it is everywhere else.
2. Behavior of the custom component should be very natural and easy to understand.
3. Code Review should always be very strict no matter how small the feature/code is.

 # RCA driving

1. Did RCA of Image Copy Paste issue in Tiny Editor
2. Did RCA of QUIZ-SDK socket issues and fixed it by refactoring the QuizInit file.
3. Did Analysis of Auto Print Failure Issue
4. Did RCA of image fit issue dialog in constrained sheet
5. DId RCA of Constrained Paper Print issue on latest chrome version

Stakeholders Management / Integration Support

Process Improvement

1. We have created guidelines to strictly follow while doing estimation and task breakup, solution & coding.
2. While coding I keep in mind
    a. write very organized, simple and **easy to understand and maintain code**.
    b. component/util **separation on concern** so that it can be used at many other places
    c. component/util **scalability & extensibility** so that it can be used to handle many other feature
    d. check for **responsiveness, keyboard accessibility** and add Html tag aria attributes to make to accessible to specially abled users
    e. For improving performance, I use **debounce, throttle** wherever I feel the need, compress images, and take care of **Perception (Loader/error)**. load Non-critical stuff later
    f. Error handling without breaking the code/experience.
    g. Follow the design system of the project
3. In code review I check for
    a. responsiveness
    b. loader/error handling
    c. null check

    d. localisation (react-intl)
    e. Impact on other components/routes/files/sdks
    f. Asset Optimization
    g. sendToAnalytics events
    h. logs
4. Before PR merge, I check for
    a. Build success in local
    b. errors/warnings/consoles on browser console
    c. Impact on others sdks

Innovation

1. The code structure we follow has been proven very readable/maintainable and easy to understand.
    a. Code looks very clean & organized.
    b. We are able to debug the issue/development problems very quickly
    c. We are able to locate the culprit file/component/function very easily

Grooming resources

1. We have created guidelines for other developers to follow while doing estimation and coding
2. Help them in solutioning and give them other approaches possible and help them decide on the best solution.
3. Help them in estimation and task breakup.
4. Help them with their issues/problems they face while development
5. Always Review their code and check the feature on local before merging.
6. **IMPACT:**
    a. Me & **Tanmaya** in developed the **Correction SDK**
    b. **Amritpal** has developed the **Upload SDK**
    c. **Amritpal & Dhairya** has developed **Report SDK**
    d. **Dhairya** has developed Admin Panel
    e. **Amritpal** has added Subjective Question in Quiz SDK Feature

Cross Org Contribution