System-wide design

- Backend will still remain in use of the hexagonal architecture to ensure modularity and loose coupling between modules/packages.
- AWS Cognito will have to be incorporated into the system-wide design once we get AWS set up.

Current state of Backend

- TypeScript Express is fully set up with logging, dependency injections and testing.
    - Setting MySQL and connecting it to the backend service will be the next priority for the backend
- Java Spring mitigation plan is fully set up with logging, dependency injections and testing. This mitigation plan will allow us to switch backend technology without the downtime required for setting up if needed.

Prototyping TypeScript Express vs Java Spring

- Set up a mitigation plan to switch to Java Spring in case TypeScript Express proves to be challenging, difficult or too time-consuming to set up
- Prototyped REST endpoint calls through Express and Spring
    - Created diagnostic endpoints that involved making use of Dependency Injections and framework-related REST API setups
    - Both prototypes were successful
- Prototyped logging infrastructure in Express and Spring
    - Log4j and log4js are essentially the same logging framework and had successful usage in both Express (log4js) and Spring (log4j)

Table of Pros & Cons from prototyping with the two different technologies

Pros & Cons	Java Spring	TypeScript Express
Pros	● Has flyway migration to support database schema changes ● Has many easy out-of-the-box ready-to-use implementation of concepts such as Controllers, Dependency Injections, DB Driver integration. ● Fast set-up and easy configurations compared to express ● Strongly typed, object-oriented which is what everyone is used to	● Everyone on the team is familiar with Express ● Same language for Front-end and back-end ● Supports functional programming better than Javas ● Everyone on the team is familiar with JavaScript ● Simple to use and easy to learn ● Not much business logic, so Express is more suitable
Cons	● Not everyone on the team is familiar with Spring ● Steep learning curve for Spring	● Fickle configurations, pain to configure everything ● Not everyone is familiar with TypeScript ● Have to bootstrap many different dependencies together ● Longer set up time compared to Spring

Technologies adoption/discontinuation

- Currently decided on AWS Cognito for Auth, and University servers for hosting
- Switched from Flutter to React Native due to complications regarding Dockerizing Flutter
- Considering database migration technologies such as Flyway so we can make changes to schemas in the database
    - Db-migrate is a node package that does a similar task.

Knowledge Sharing

- Knowledge share of CI/CD pipeline and Docker with buildmaster

Peer Reviews

- Backend team peer reviewed set up and changes, noted down things that should be worked on

Dev tasks for next iteration

1. ~~Restructure the codebase for Express~~ (DONE)
    a. Currently too messy, need to re-organize everything properly. Ideally should look like the file structure in the java-backup branch codebase.
2. Fix logging code in the backend. Should only have 1 logger per class. Logger should be an implementation of a logging interface and instantiated by a factory.
    a. This future-proof any changes/switches to logging packages like if we need to switch to another logger or if we want to use multiple different logging solutions
3. Set up MySQL and connecting it to the backend server
4. Set up AWS Cognito and incorporate it into our system
5. Set up environment variables for local development vs production (partially done)
6. Design and implement REST APIs
7. Create and implement repository interfaces and create domain models
8. Reconfigure log4js to properly log what we want
9. Reconfigure jest
    a. Configure to run:
        i. All tests
        ii. Tests at each suite level (unit, integration, repo)
    b. Configure to output failed tests name and location (file, line #) if there are any
    c. Set up the scripts for each respective suite in package.json
10. Look into database migration packages, this allows us to change db schemas and have it reflected everywhere and to everyone