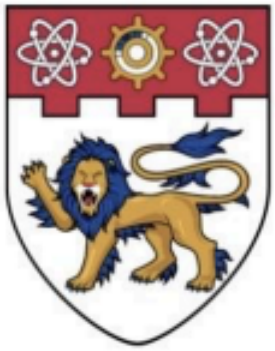


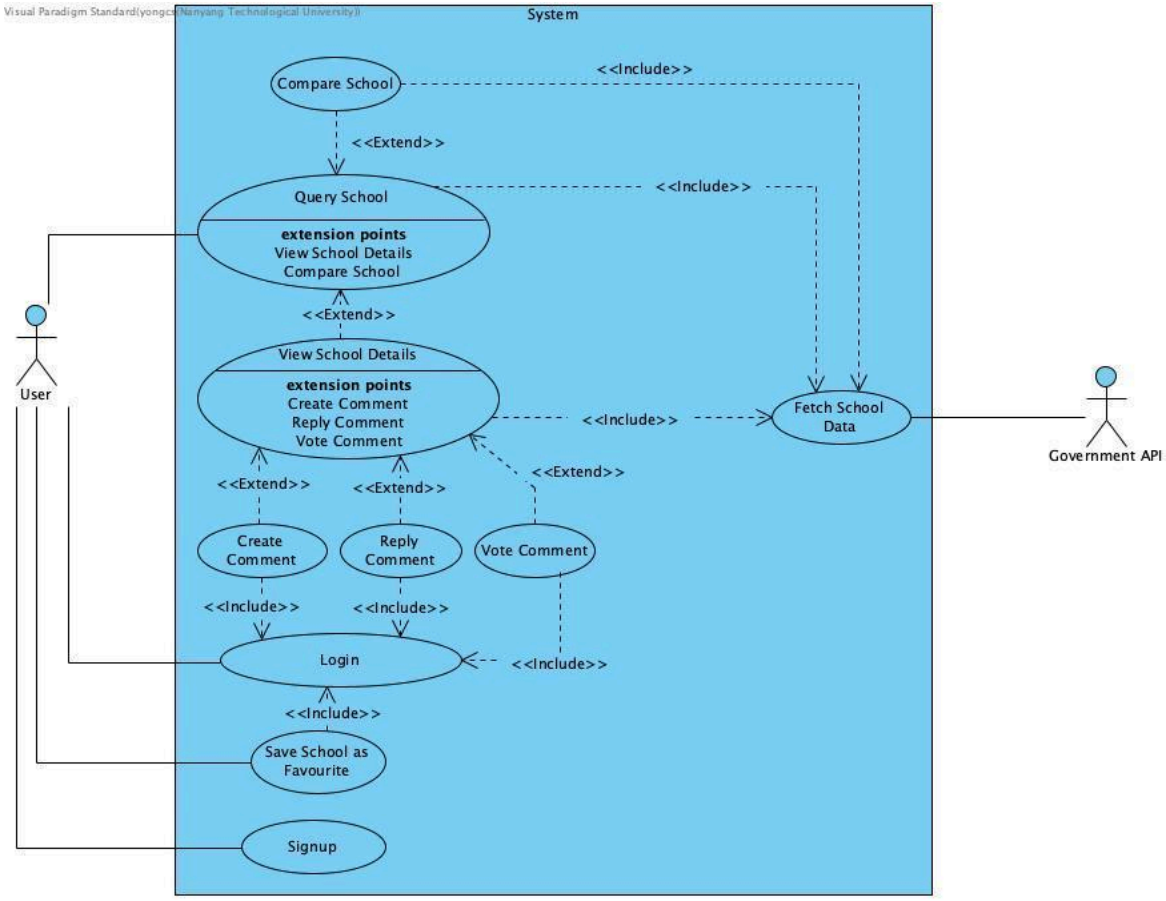
NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC2006-Software Engineering

Lab 3 Deliverables

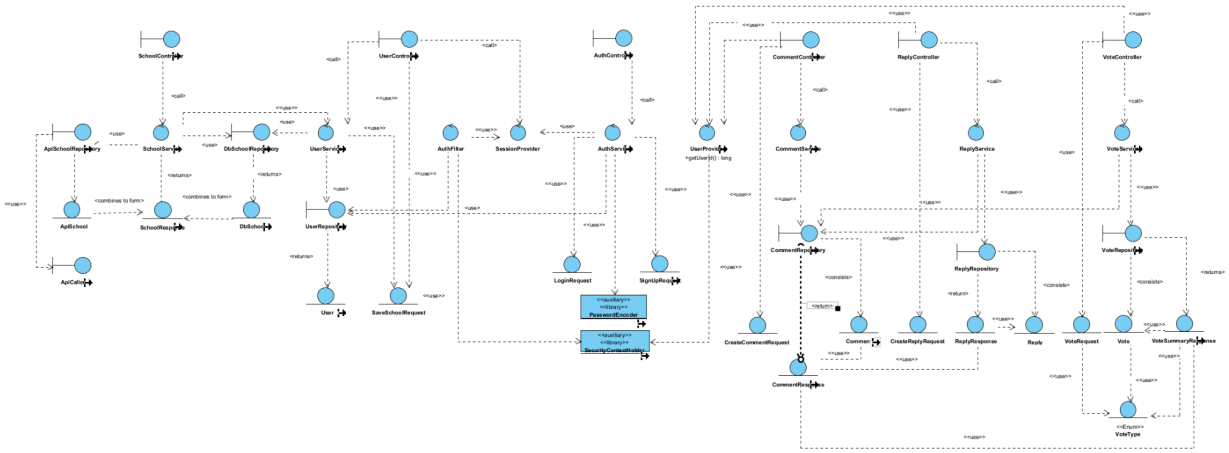
Group Member	**Matric Number**
Lim Kiat Yang Ryan	U2421937D
Yu Wenhao	U2421425F
Yong Chee Seng	U2420563K
Peng Sizhe	U2423895H
Tarun Ilangovan	U2422251A

A. Complete Use Case Diagram

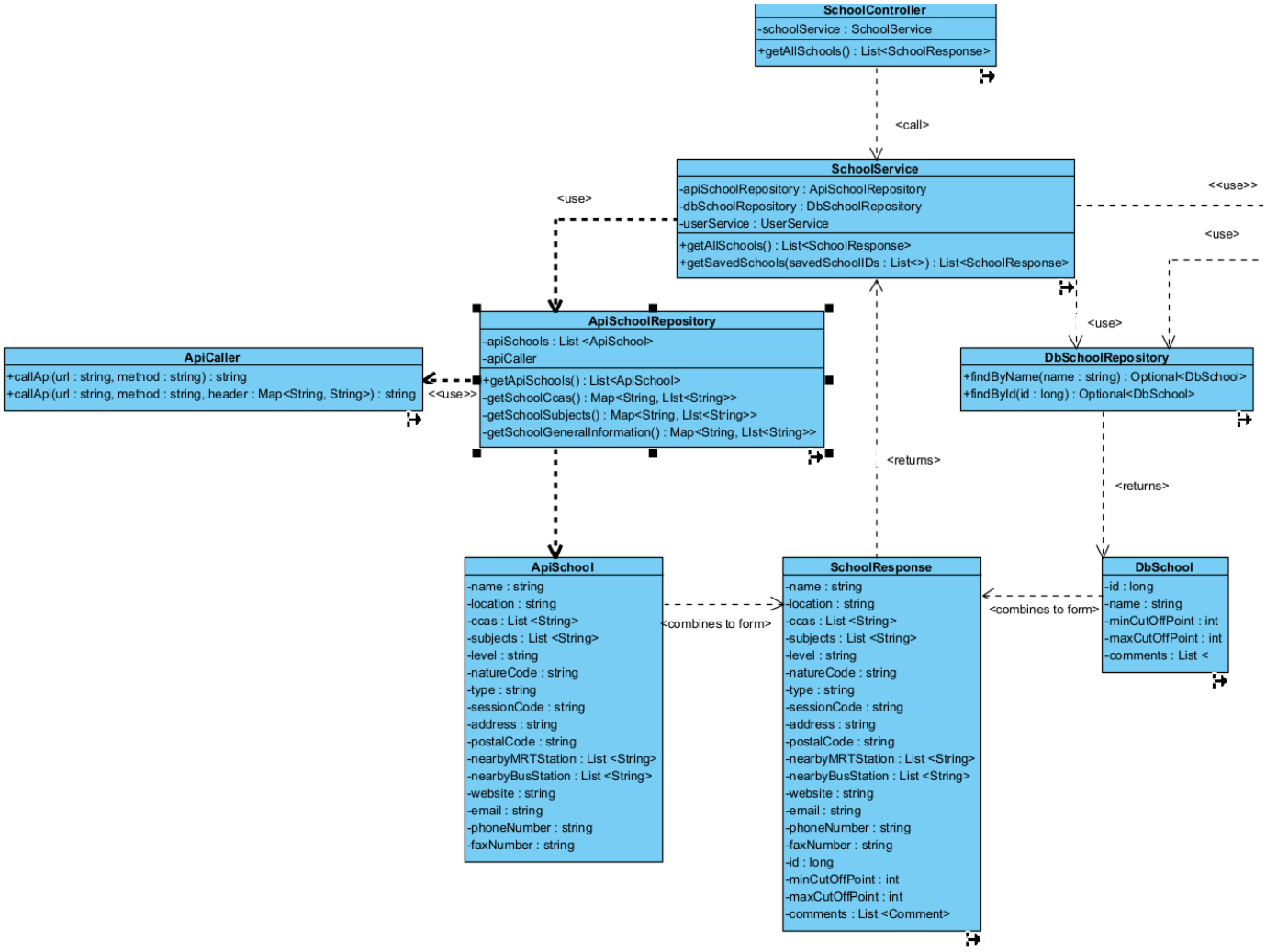


B. Design Model: Class Diagram

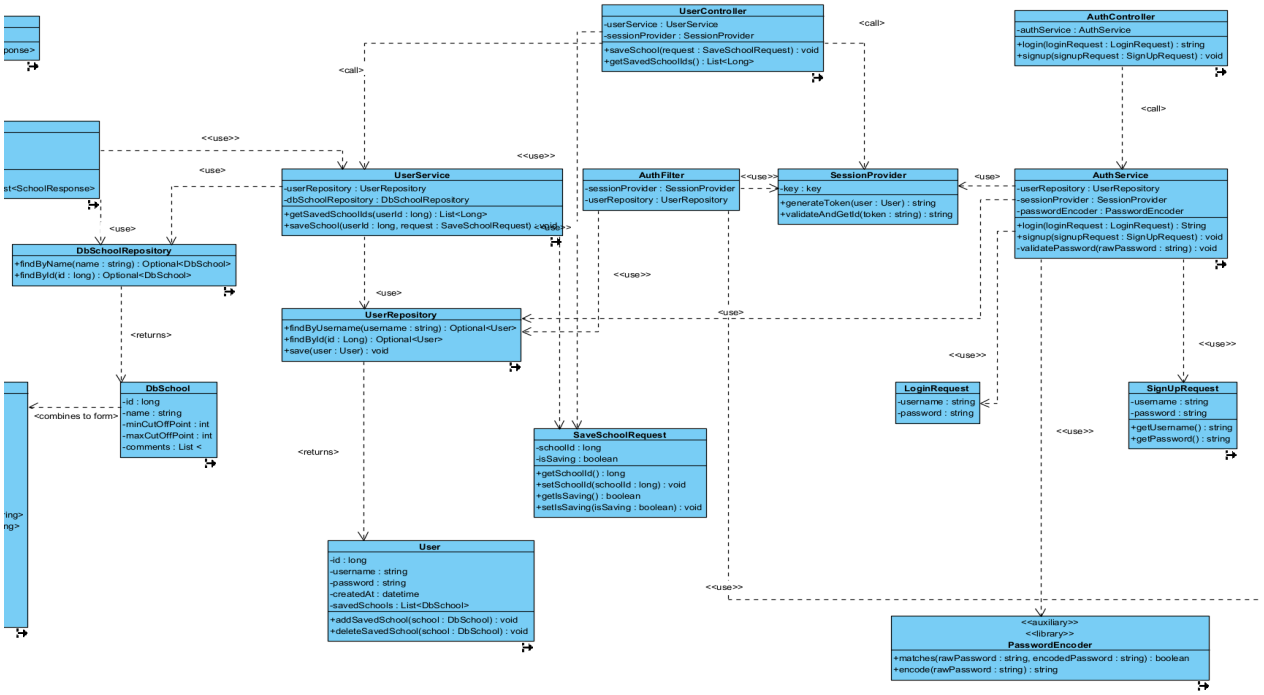
C.1 Stereotype Diagram



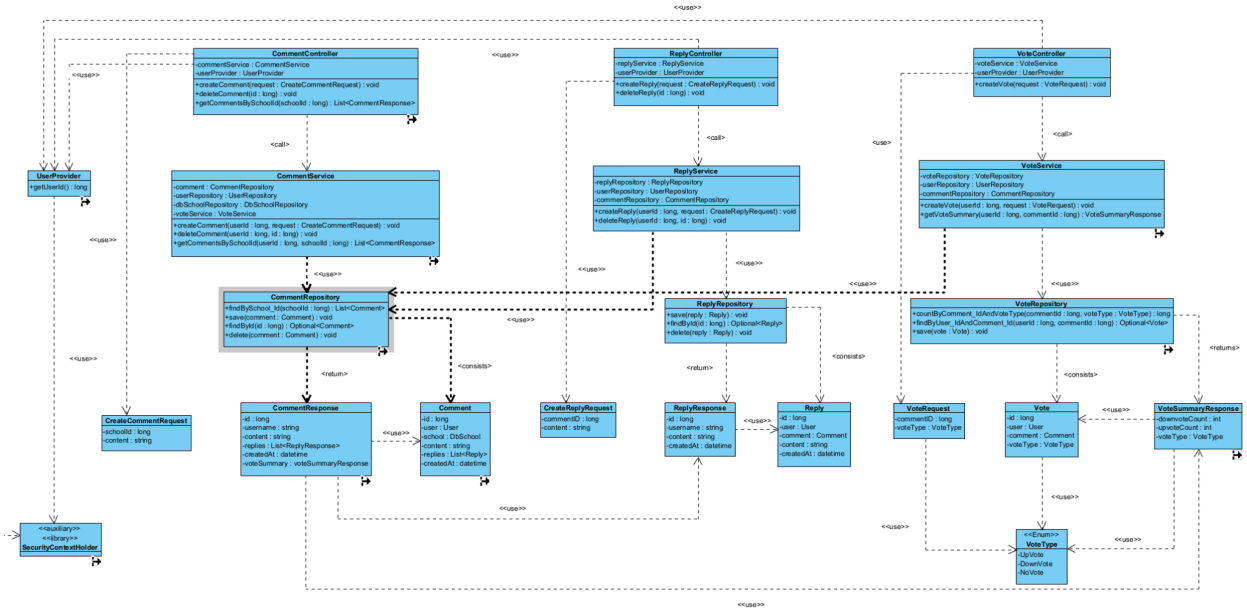
C.2 In-Depth Diagram Of SchoolController



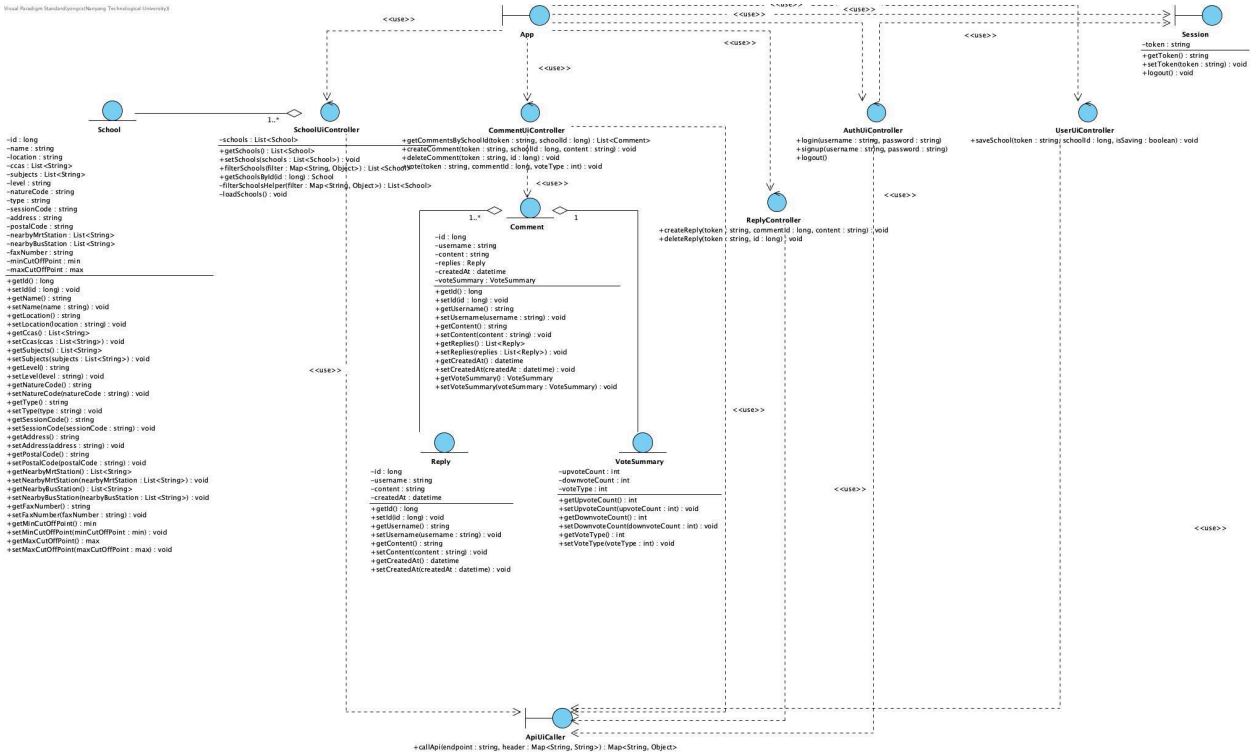
C.3 In-Depth Diagram Of UserController & Auth Controller



C.6 In-Depth Diagram Of CommentController, ReplyController, VoteController



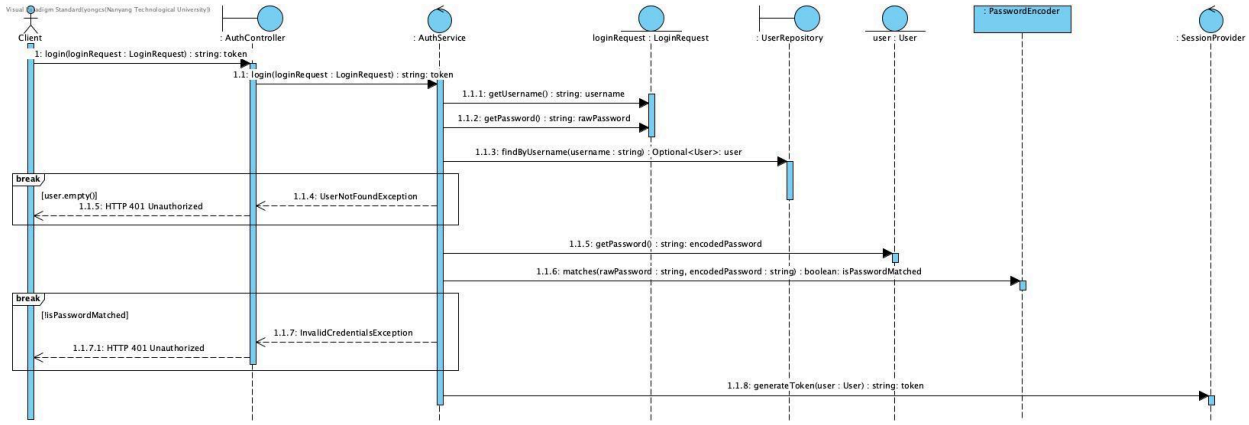
C.7 Class Diagram Of Frontend



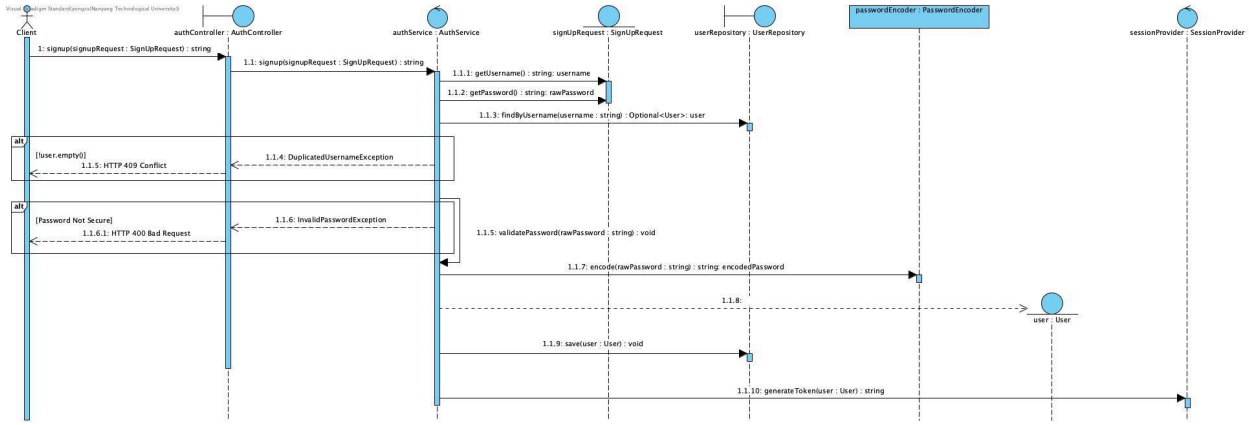
D. Design Model: Sequence Diagrams

Backend

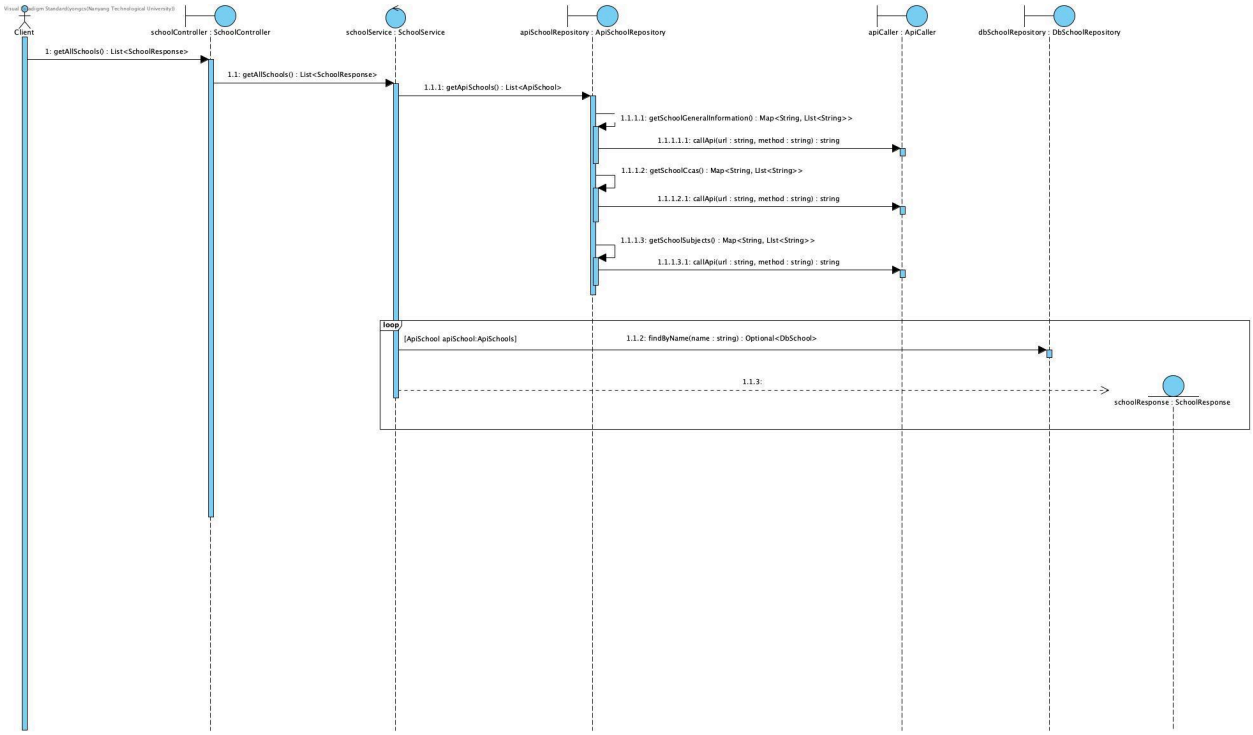
DB.1 Login



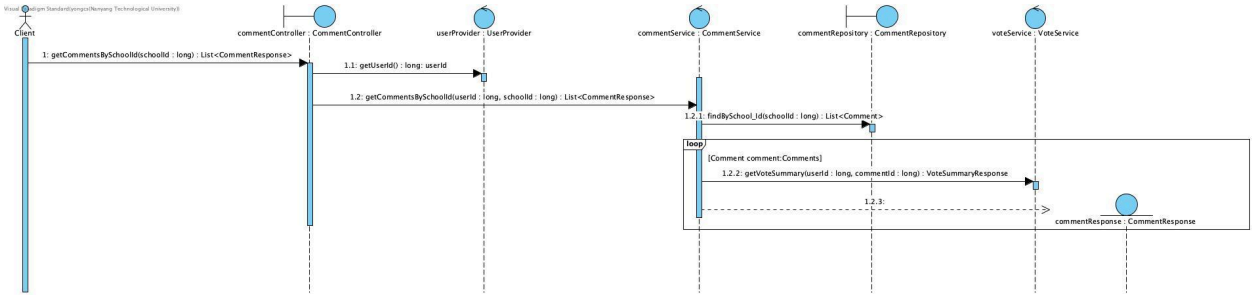
DB.2 Signup



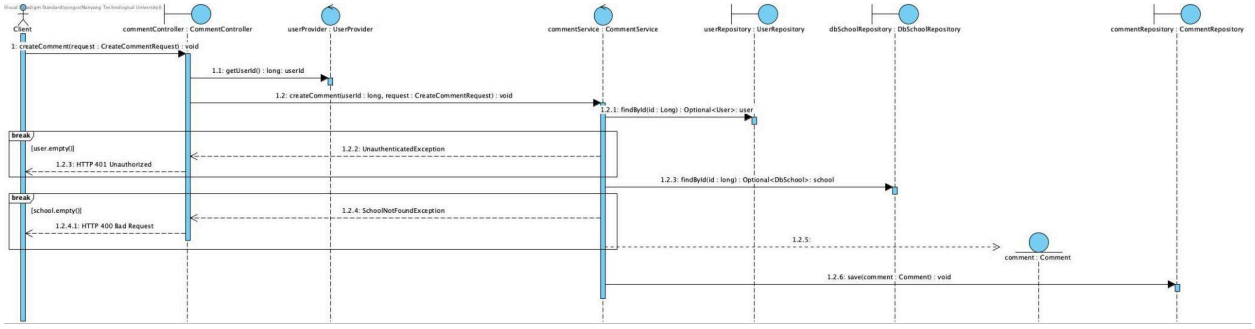
DB.3 GetSchools



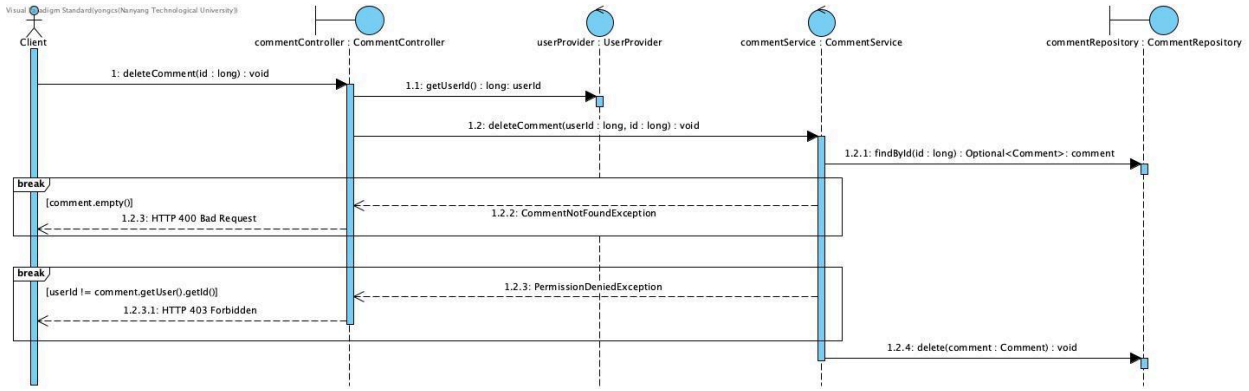
DB.4 GetCommentsBySchoolId



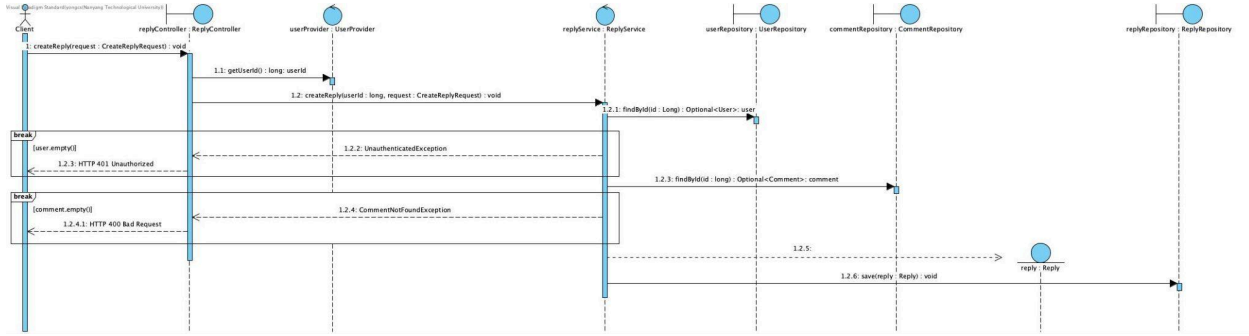
DB.5 CreateComment



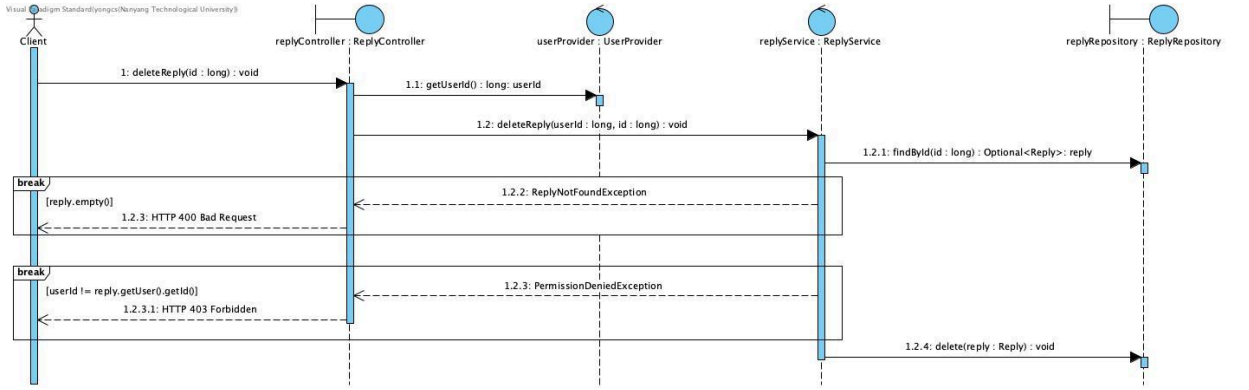
DB.6 DeleteComment



DB.7 CreateReply

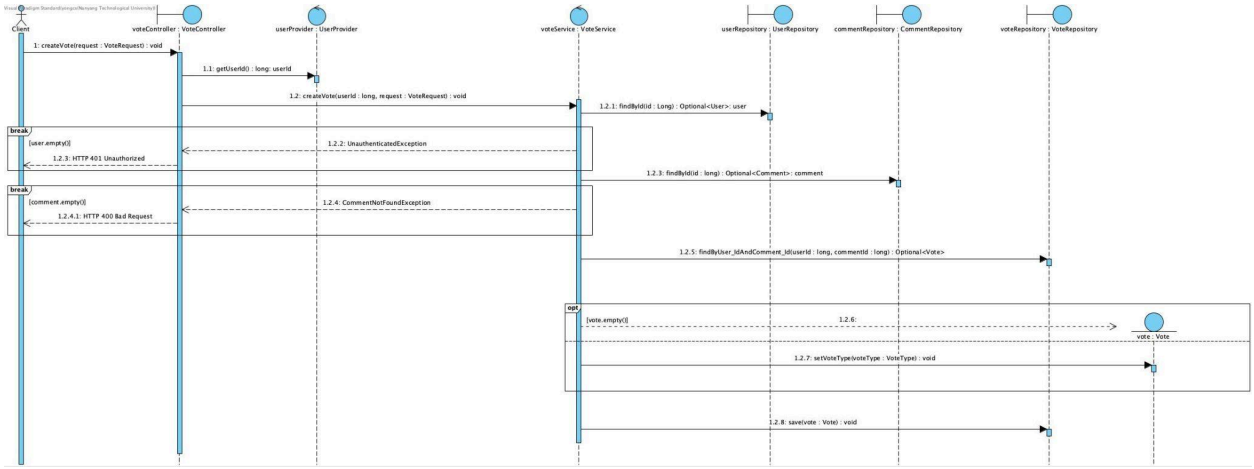


DB.8 DeleteReply

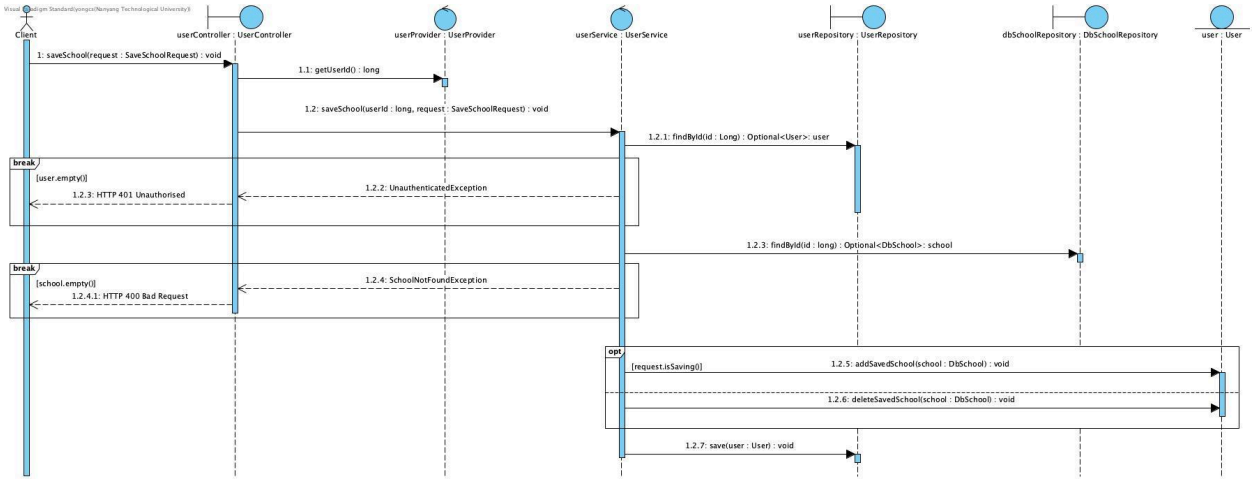


DB.9 GetVoteSummary

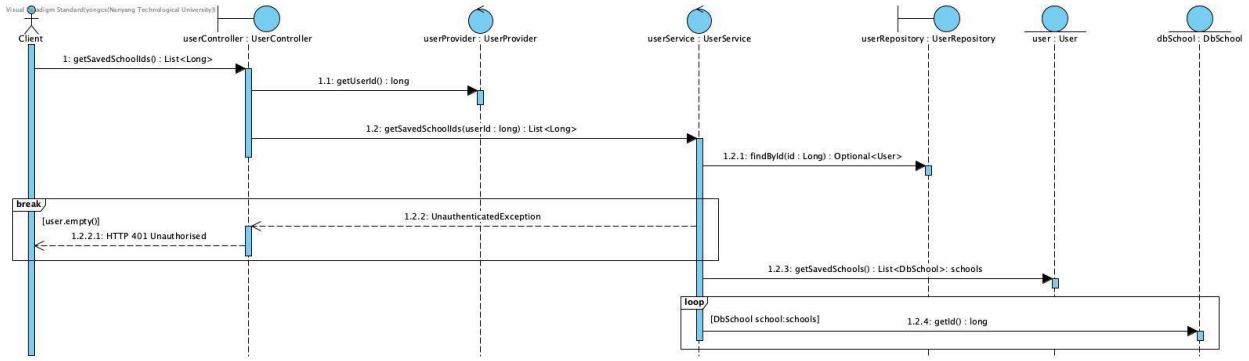
DB.10 CreateVote



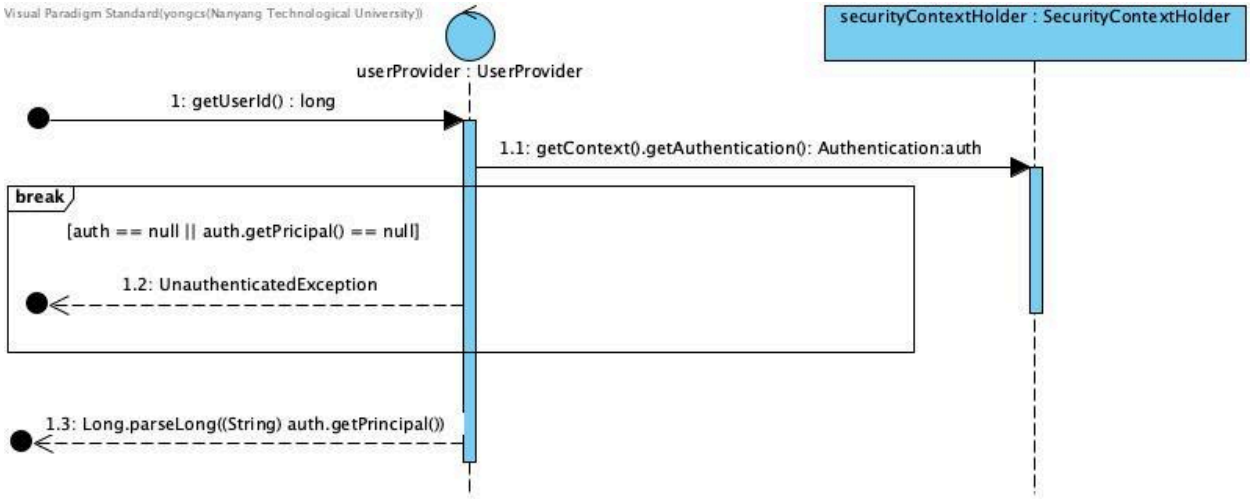
DB.11 SaveSchool



DB.12 GetSavedSchools

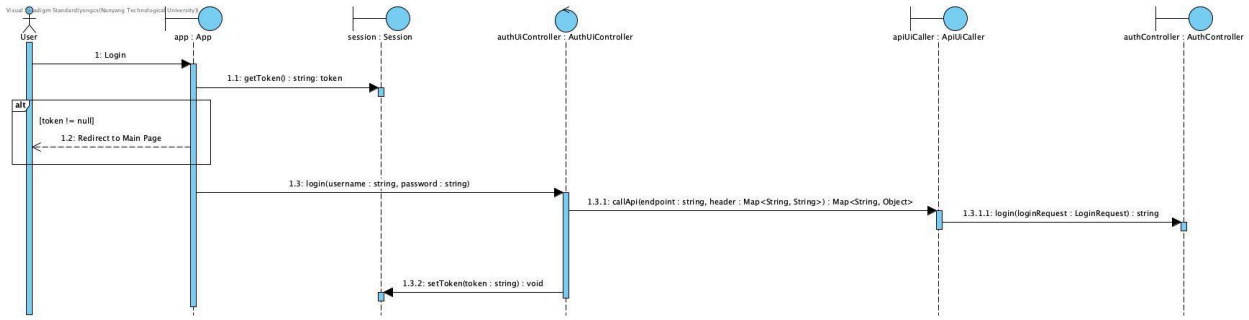


DB.13 GetUserContext

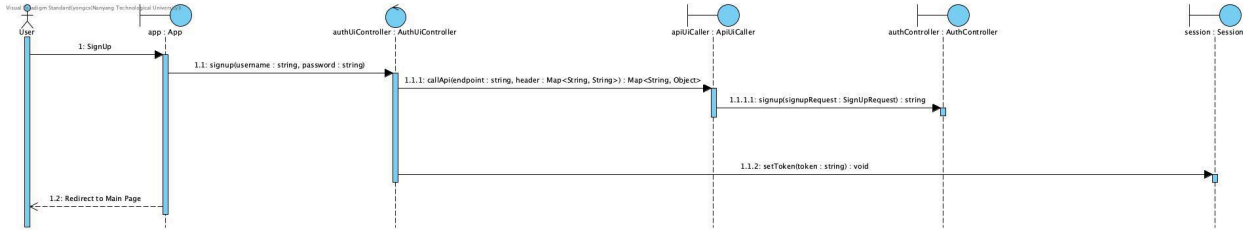


Frontend

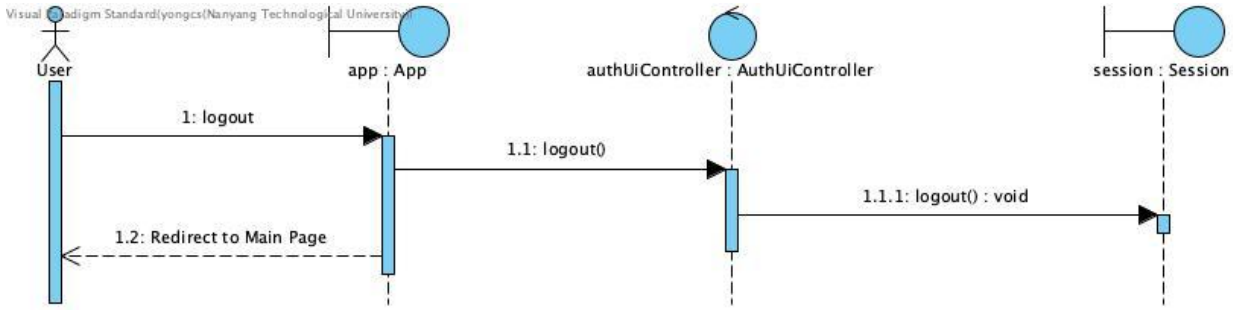
DF.1 Login



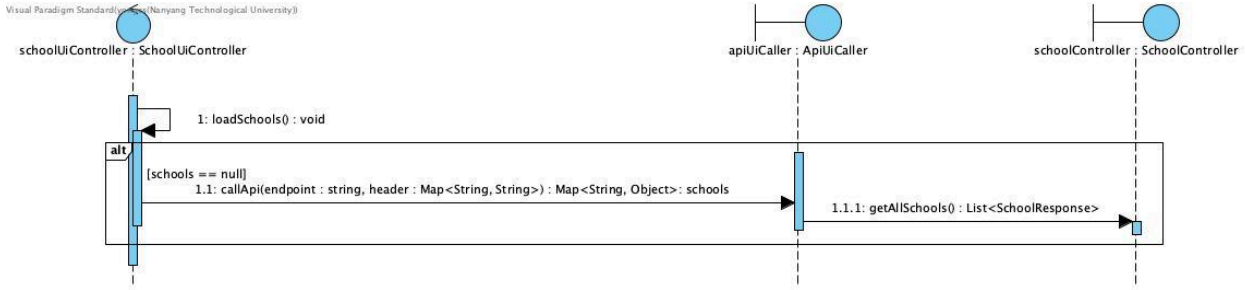
DF.2 Signup



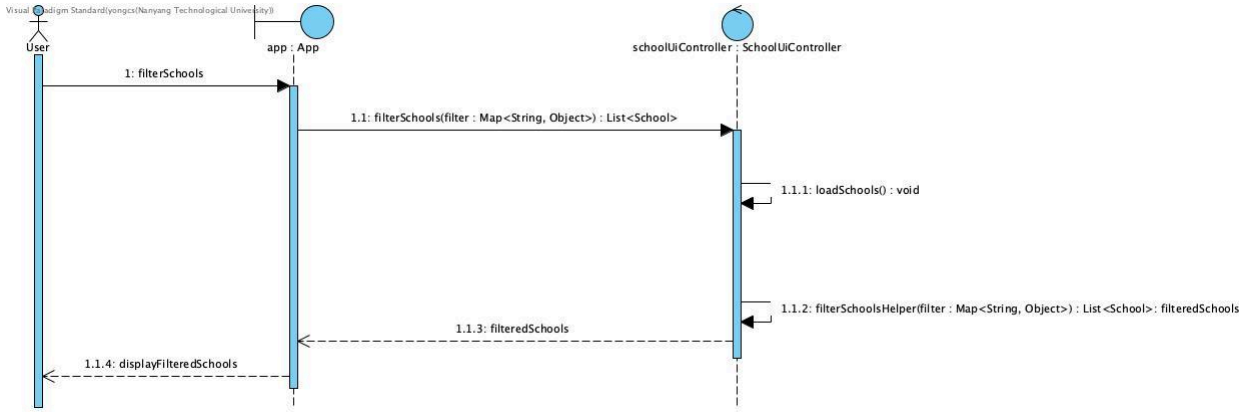
DF.3 Logout



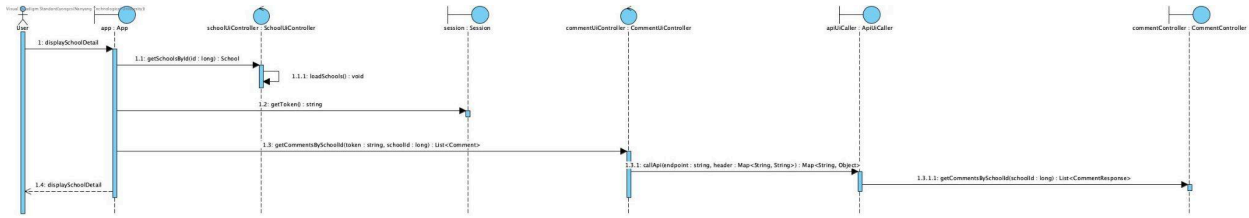
DF.4 LoadSchools



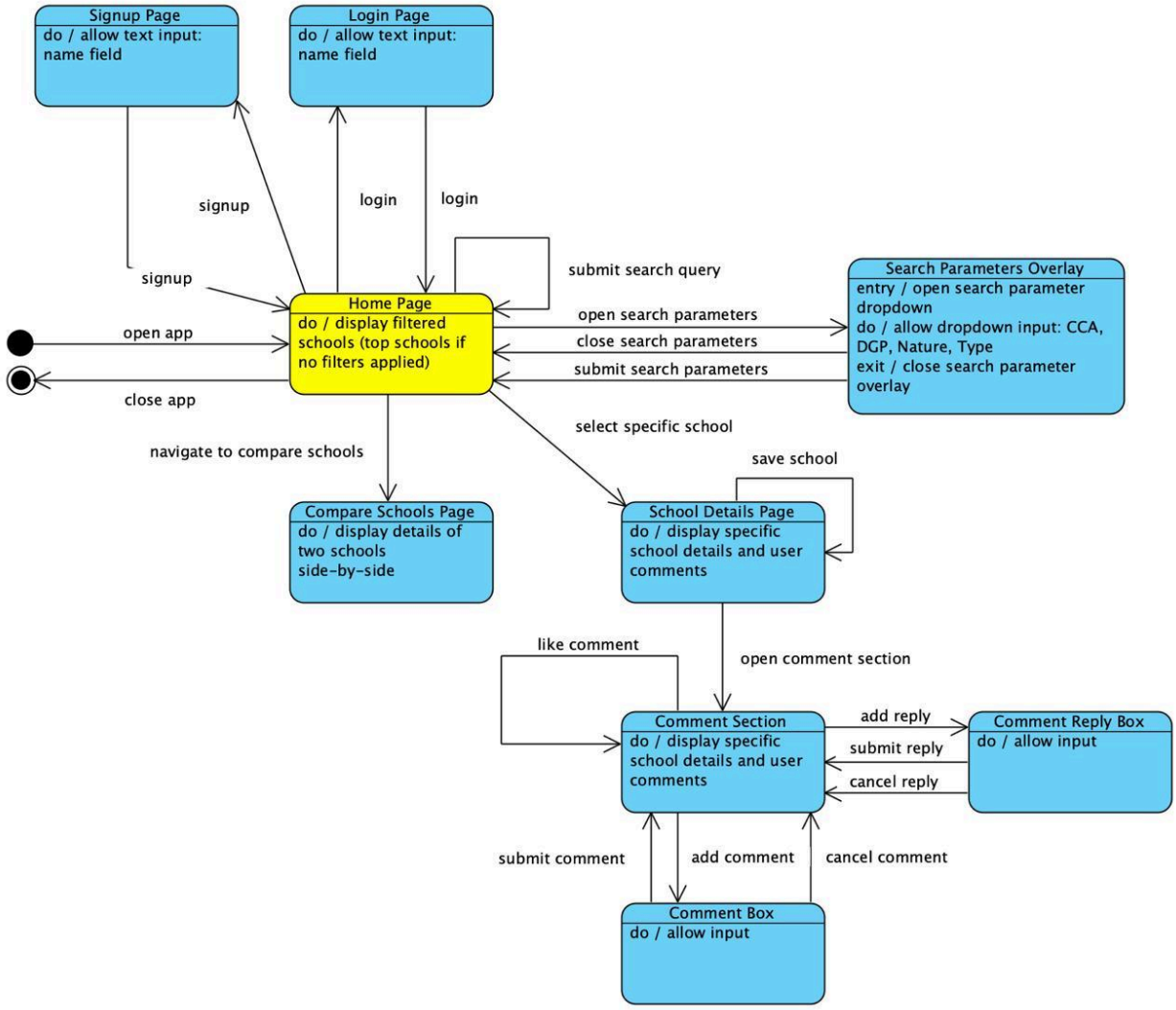
DF.5 FilterSchools



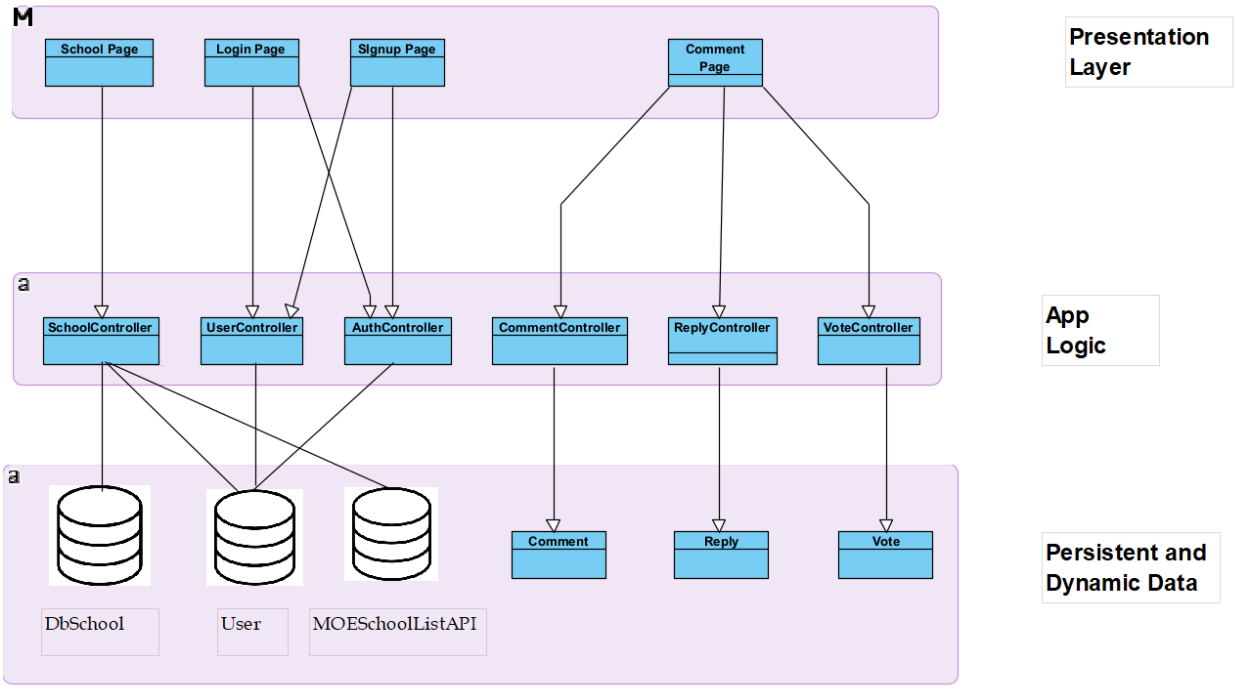
DF.6 ViewSchoolDetails



F. Design Model: Dialog Map



G. System Architecture



We will mainly be adopting a closed-layered architecture with the above 3 layers:

- Presentation Layer: This consists of the User Interface(UI), the frontend. The main interface will be the search interface and a menu to search for schools. There will also be webpages for users to login, and comment.

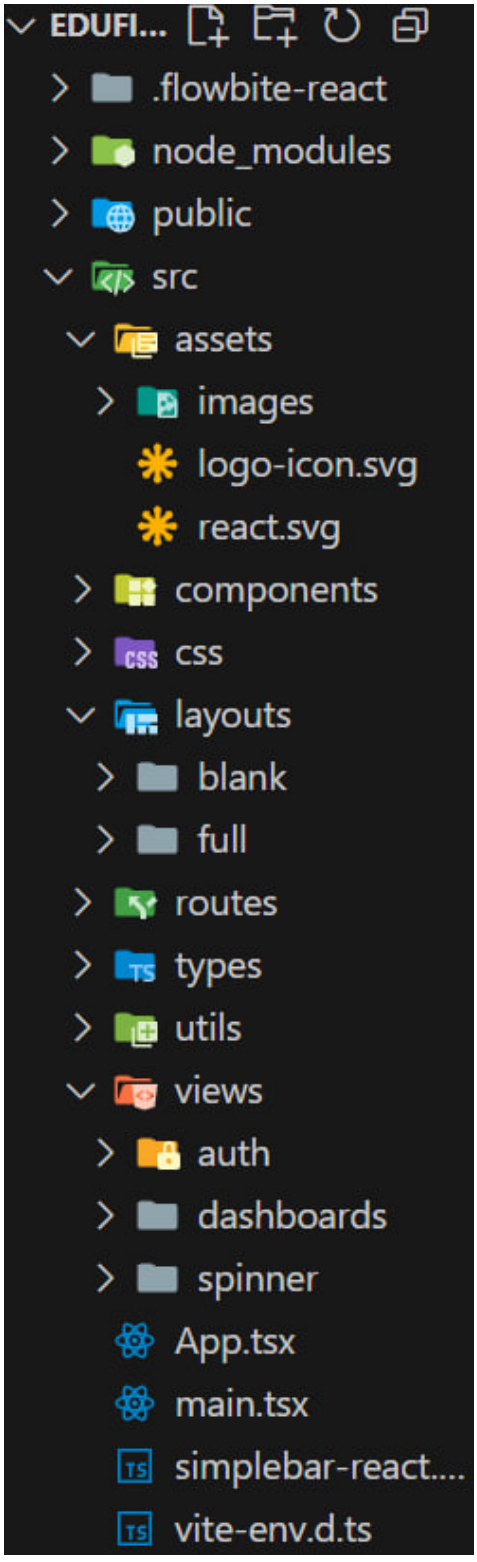
- Logic Layer: Implements the business logic which processes school data, location data, user data, and settings.

- Database Layer: This stores and manages all data storage i.e. Schools, User, etc.

We chose this style of closed-layer architecture due to numerous benefits. Presentation layer shows what the user can see in the browser. This architecture boosts security, as users cannot tap into the app logic and database and meddle with them from the presentation layer. There are clear dependencies between layers, resulting in simple maintenance as changes in one layer will not affect other layers. Thirdly, it promotes high scalability, as we can keep on adding classes that will not interfere with other layers. It is open to different adaptations too. It is also easily enhanced. Testing becomes elementary as each layer can be tested in isolation, with mock interfaces for adjacent layers and clear testing limits.

H. Application Skeleton

Frontend (React+Vite w/ TypeScript, Flowbite & Tailwind CSS)



Components:
This folder contains all our reusable components, like our filter menu and sidebars that appear on each page.

CSS: This folder contains our CSS styling.

Layouts:
We have two different layouts, a full layout for our main pages and a blank layout for our login/register page

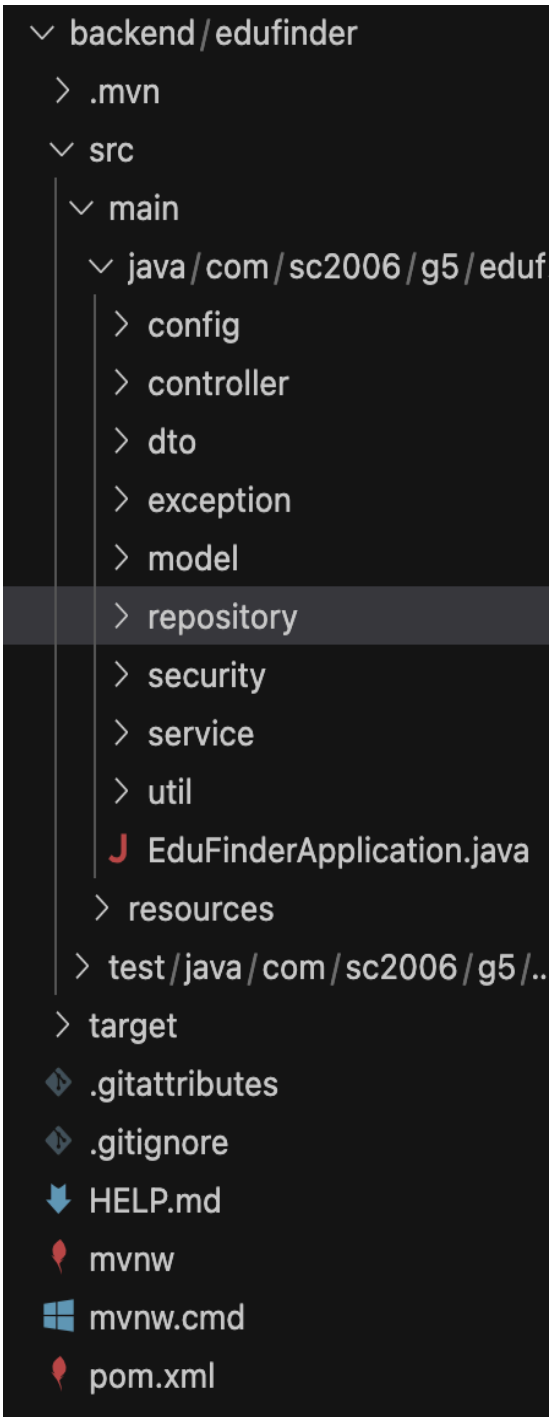
Pages:
This contains all the different pages that users will be able to interact with, for example, the home page and individual pages for schools.

Routes: This folder contains our routing for our pages.

Types: This folder contains the types used in Typescript.

App.jsx:
This is the entry point for our application, which routes our user to the correct page.

Backend (Java Spring Boot)



/src/main/:
Directory for the main application

/src/main/…/config/:
Directory for the config classes such as security and cors.

/src/main/…/controller/:
Directory for the REST Controllers that map API request

/src/main/…/dto/:
Directory for DTO objects for request and response.

/src/main/…/exception/:
Directory for custom exceptions.

/src/main/…/model/:
Directory for models (entity).

/src/main/…/repository/:
Directory for JPA repositories and custom repositories to store and handle data.

/src/main/…/security/:
Directory for security related classes such as SessionProvider and AuthFilter

/src/main/…/service/:
Directory for services that handle business logic.

/src/main/…/util/:
Directory for util classes.

/src/test/:
Directory for tests, have the same structure as main.