Software Requirements Specification

for

edUFind

Version 1.4 approved

Prepared by Team UPS Elon

Nanyang Technological University

9 April 2022

Revision History

Name	Date	Reason For Changes	Version
All Members	22 March 2022	First draft of document	1.0
All Members	01 April 2022	Partial Documentation	1.1
All Members	07 April 2022	Completed Documentation and Formatting	1.2
All Members	08 April 2022	Refinement of document	1.3
All Members	09 April 2022	Refinement of document	1.4

1. Introduction

1.1 Purpose

This is the Software Requirements Specification document for a web application: edUFind. The purpose of edUFind is to take advantage of publicly available government data (data.gov.sg) to underpin creative innovation to further accelerate Singapore's development into a Smart Nation. edUFind intends to do this by assisting students in their decision making process in determining what is the next step in their academic journeys.

This document describes the scope, objectives, goals, functional and non-functional requirements, conceptual and dynamic models of the edUFind application.

1.2 Document Conventions

For this SRS document, we will be using the **Arial** font with font sizes 18, 14, 12 and 11 for the headings, subheadings 1, subheadings 2 and content, respectively. Headers are in bold.

1.3 Intended Audience and Reading Suggestions

The intended audience for this software requirement specification document are the Users, Developers and Testers. We recommend beginning reading the document with the overview sections and proceeding through the sections relevant to each reader type. For example, web application developers may find sections pertaining to website development more relevant.

Our Users are students who are interested in finding out more about the next stage in their education. The users can use this document to discover the application's main functions. The sections relevant to them will hence be the User Interface (UI) Mockups and the Functional Requirements.

A Tester can also use this document to understand the app's functionality and conduct BlackBox testing based on the described functions. The sections relevant to them will be Design and Implementation Constraints, System Features and Security Requirements.

The Developer can use the document to review the application's current capabilities and gain insight as to which section of the application requires attention and the implementation of new functions.

1.4 Product Scope

Our web application, edUFind, aims to provide users with a convenient platform with comprehensive information about the schools in Singapore, ranging from the Primary to Junior College level. In doing so, our users will have the necessary information they need to make an informed decision about which school is the right next step for them on their educational journey.

edUFind also provides tailored recommendations to each user based on their interests and academic preferences. Furthermore, edUFind allows users to maintain a shortlist of the schools they are interested in. Users can also write comments to explain what it was about a certain school that piqued their interest and made them add that school to their shortlist.

Ultimately, our goal is to aid students by simplifying the process of choosing a school that is best suited to their unique needs and interests.

1.5 References

1.5.1 Front End Development Tools

- We used the following tools to style our application's User Interface (UI) components:
    - React https://reactjs.org/docs/getting-started.html
    - Redux https://redux.js.org/
    - Material UI https://material-ui.com/
    - React Router https://reactrouter.com/

1.5.2 Backend Development Tools

- We used the following tools to develop the server side of the application:
    - NodeJS https://nodejs.org/en/docs/
    - ExpressJS https://expressjs.com/en/5x/api.html

- We used MongoDB as our Database Management System (DBMS) to store all user related information. Its documentation can be found at the following link: https://www.mongodb.com/docs/

1.5.3 Third Party APIs

- We obtained all the school related information displayed on our website from the “School Directory and Information” Data API from https://data.gov.sg/ . The documentation for the same can be found here: https://data.gov.sg/dataset/school-directory-and-information

1.5.4 Cloud Hosting Tools

- Our Website is hosted on Amazon Web Services (AWS), specifically on an AWS EC2 Instance. The documentation can be found here: https://docs.aws.amazon.com/ec2/?id=docs_gateway
- We used NGINX as our public facing web server to receive HTTP requests for both the front-end and back-end. The documentation can be found here: http://nginx.org/en/docs/
- We used PM2 as a process manager to run and manage our NodeJS Backend. The documentation can be found at the following link: https://pm2.keymetrics.io/docs/usage/quick-start/

1.5.5 Testing Frameworks

- We used the Python Selenium Library to conduct automated testing. The documentation can be found here: https://www.selenium.dev/

2. Overall Description

2.1 Product Perspective

Students may often find it challenging to make decisions with regards to the next step of their education due to the vast array of choices available as well as the difficulties involved in obtaining information about each school. edUFind addresses this by providing a one-stop platform which allows students to obtain all the relevant information they need about a school to make an informed decision about which school is the right fit for them as they go to the next stage of their academic journey.

edUFind is a standalone web application developed using the MERN Stack (MongoDB, Express, React, Node.js). Both the frontend and backend of the application are hosted on an Amazon Web Services EC2 instance. We used publicly available government data from https://data.gov.sg/ to obtain all school related information through an API call. The user specific data is stored using MongoDB.

2.2 Product Functions

edUFind has the following key functions:

- Searching for Schools
- Filtering of Schools based on specified fields
- Displaying of school details
- Registration of new user account
- User login and logout
- Shortlisting of schools
- Editing of user password
- Commenting on schools
- Recommendation of schools

Kindly refer to Appendix B's Conceptual Diagrams (Class Diagram) and Behavioral Diagrams (Dialog Map and Sequence Diagrams), which detail the various class interactions and states of the application.

2.3 User Classes and Characteristics

Unregistered Users: Unregistered Users are users without an edUFind account. They are able to search and view the details of different schools. However, they are unable to access other key functions such as shortlisting and customized recommendations. Unregistered Users are less likely to be frequent users of the application, using it in a sporadic manner.

Registered Users: Registered Users are users with an edUFind account. While they are able to access all functionality available to Unregistered Users, they are also able to use several other "Registered User only" features. Registered Users are expected to be more frequent visitors of the website as they are actively utilizing edUFind to try and find the ideal school for themselves, taking full advantage of the additional features that edUFind provides once a user logs in. Registered Users are our priority users.

Developers: The developers of edUFind should be able to view Registered Users' information (apart from their account password). Furthermore, they should be able to make alterations to the MongoDB database with the ability to add, edit, and delete, entries from the database.

2.4 Operating Environment

1. edUFind can be used on any browser that is connected to the internet
2. Database:
    a. MongoDB Atlas
    b. Government Data API
3. Backend Server
    a. Runtime Environment: Node.js
    b. Node.js Web Framework: Express
    c. Open-source web server and reverse proxy: NGINX
    d. Process Manager for Node.js applications: PM2
4. Hosted on an AWS EC2 Instance

2.5 Design and Implementation Constraints

1. Due to the limited amount of publicly available data, the amount of information that can be displayed on edUFind is limited. For example, cut-off point data of various schools is not publicly available. Therefore, we are not able to advise users as to which schools may be a good academic fit as per cut-off points.

2. Due to cost constraints, edUFind is running on the free tier of the AWS EC2 instance. As a result, we are bound by the limited runtime and storage capacity capability offered in the free tier.

2.6 User Documentation

1. A video walkthrough demonstrating the key functionality of edUFind was produced to help users understand the functionality and navigation of our application. The video also serves to guide users as to how they can leverage the full power of edUFind.

2. The Data Dictionary defines the key terminologies used (refer to Appendix A).

2.7 Assumptions and Dependencies

2.7.1 Project Schedule

- Given a timeline of 13 weeks for the project, the team must complete and satisfy all functional and non-functional requirements. In addition, the team must ensure that the application is error-free and is free of defects.

2.7.2 User hardware capability

- Users must have an electronic device that is capable of running this web application.

2.7.3 Connectivity

- Users require an internet connection for authentication and data management purposes. The server requires an internet connection to access the relevant third party APIs.

2.7.4 Authentication

- Users have the choice of authenticating themselves by providing an email address and a password during the creation of an account and when logging into the application using the aforementioned email address and password.

2.7.5 User Assumptions

- Users are assumed to understand the basics of navigating a web application.
- Users are assumed to have access to a reliable internet connection.

2.7.6 Dependencies

- Our team's web server is hosted on the AWS cloud. In the event of a connection issue with the AWS cloud service, our web application would cease to run. In light of this possibility, our team has prepared a failsafe by hosting the web application on another backup server as well.
- Since the school data used in our web application is pulled from multiple APIs obtained from data.gov.sg, our web application will only function well and remain current if the data is updated in a timely manner and if the API remains available for public access.

3. External Interface Requirements

3.1 User Interfaces

3.1.1 Home Page & Navigation bar

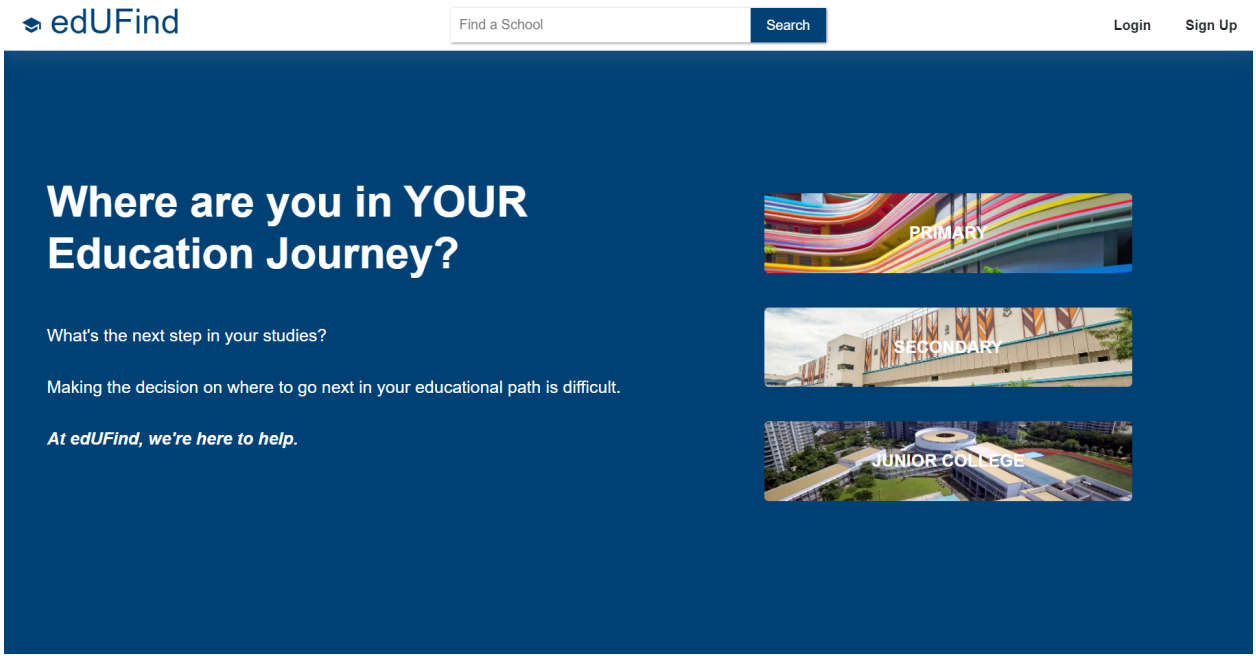


Fig 1. Homepage

Description:

- The home page briefly describes the purpose of edUFind.
- On the right-hand side of the screen, there are three containers that are ordered by education levels. Users can click on them to be redirected to a list of all schools that belong to the labeled education level.
- The navigation bar at the top of the screen is persistent and visible on every screen of edUFind.
    - Users can click on the edUFind logo from any screen to immediately return to the edUFind home page.
    - Users can search for schools using the search box.
    - Users that are not logged in can click on the ‘Login’ button to log in.
    - Users can also register for an account by clicking the ‘Sign Up’ button.

3.1.2 Registration

Registration requires users to run through 3 UI pages to create a personalized account.

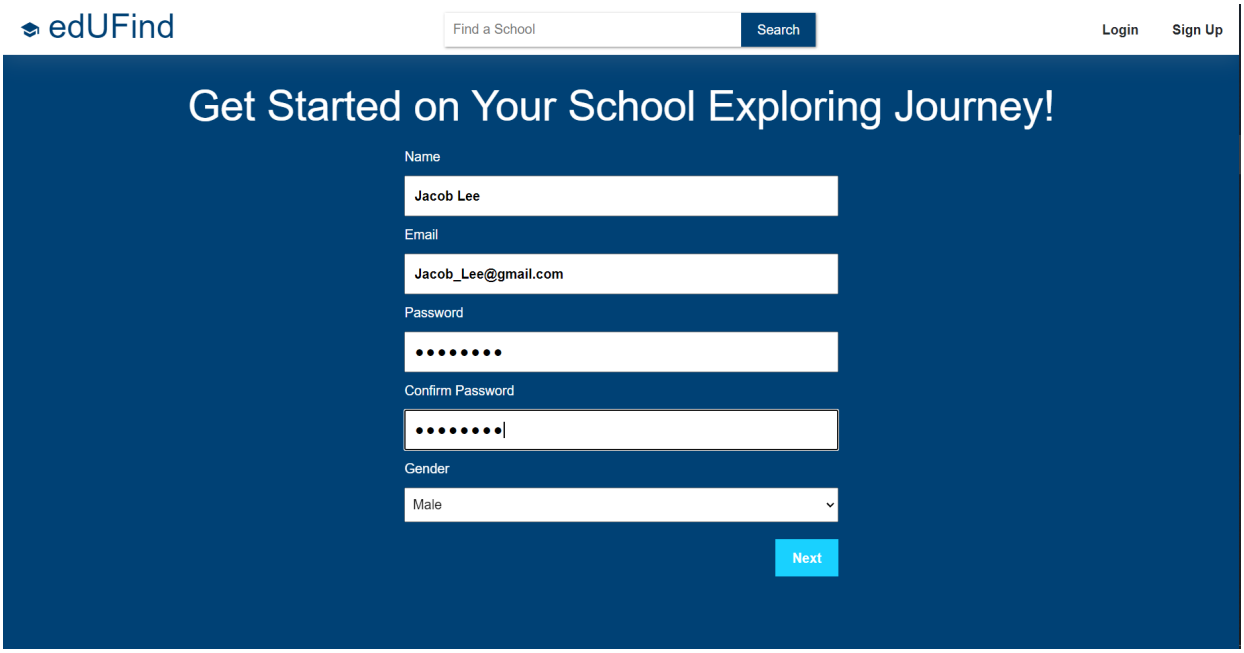


Fig 2a. Page 1 of User Account Registration

Description:

- When the user is not logged in and the user clicks on the “Login” Button on the Navbar, the user will be brought to Fig. 2a.
- In Fig. 2a, the user will input their personal information to create a user account. Starting from the top, the user will input their Name, Email Address, Password, Confirm Password (repeat password entry) and Gender in the respective input fields.
- Once the user is ready to proceed with registration, they will click on the “Next” button to proceed to the next page in the registration process.

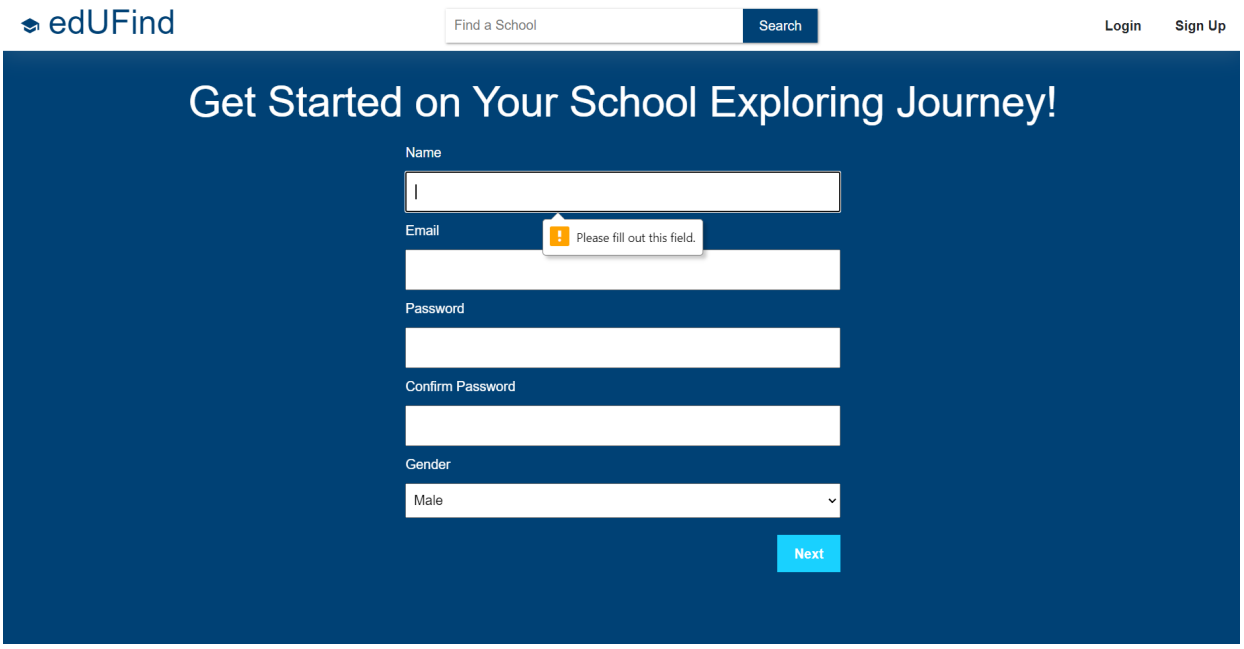


Fig 2b. Error during Empty Name Field

Description:

- For simplicity, Fig. 2b only shows the event when the first input box is left empty by the user.
- Continuing from Fig. 2a, after the user clicks the “Next” button, if they left any input field empty, an alert will prompt the user to enter their details in that field.
- The user is unable to proceed until all fields are filled with valid user inputs.

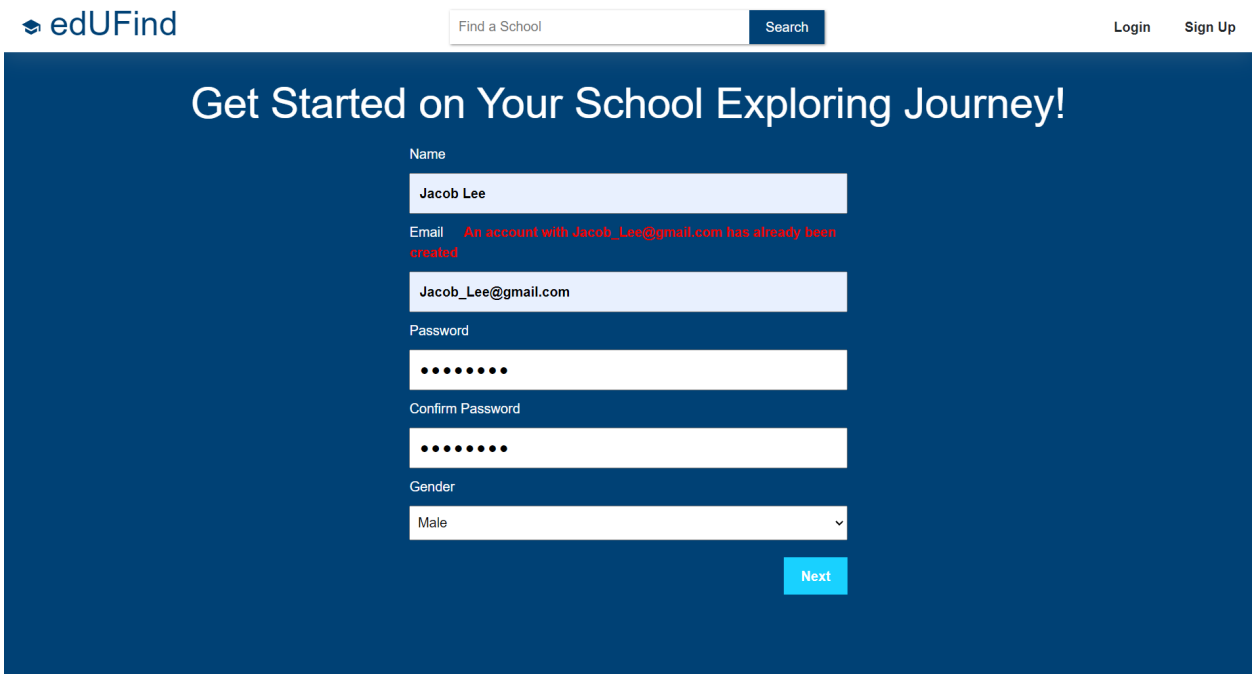


Fig 2c. Error during duplicated email address

Description:

- If the user inputs an email address that already exists in the database, an error message is presented to the user that states that the inputted email address is already associated with an edUFind account.
- The user will not be able to proceed to the next page for registration until they enter an email that is not associated with any existing account and therefore does not exist in the database.

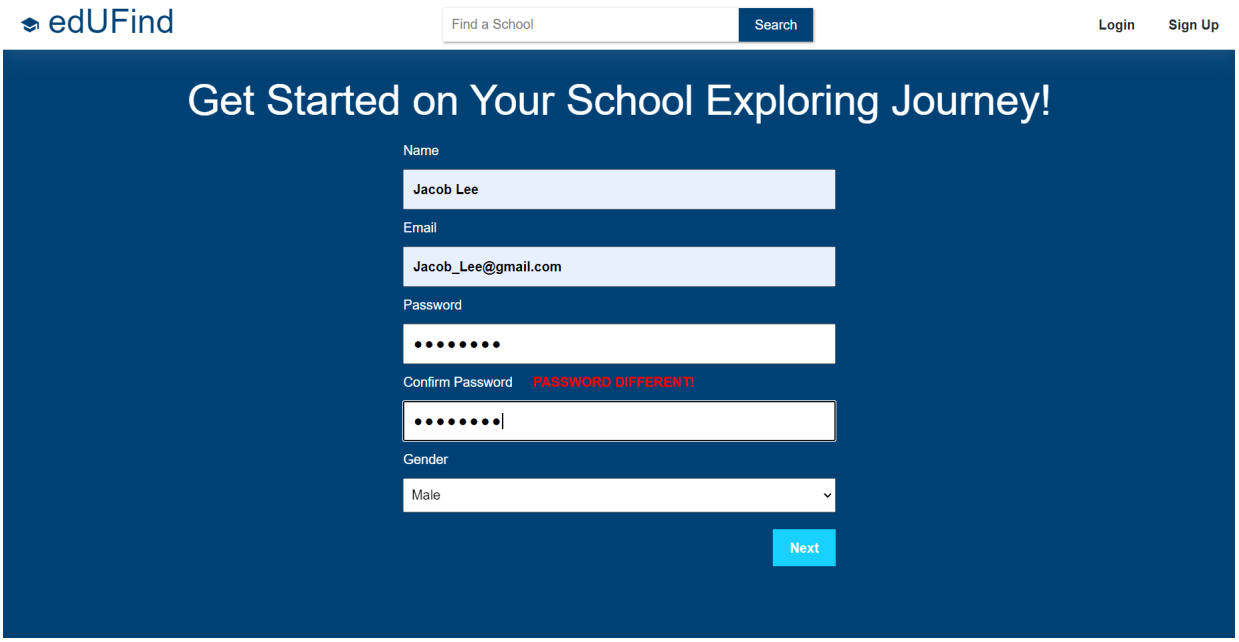


Fig 2d. Error during mismatch of password

Description:

- If the user does not enter matching values in the “Password” and “Confirm Password” fields, the system will notify the user of the mismatch.
- The user will not be able to proceed to the next page of the registration process until they rectify the mismatch.

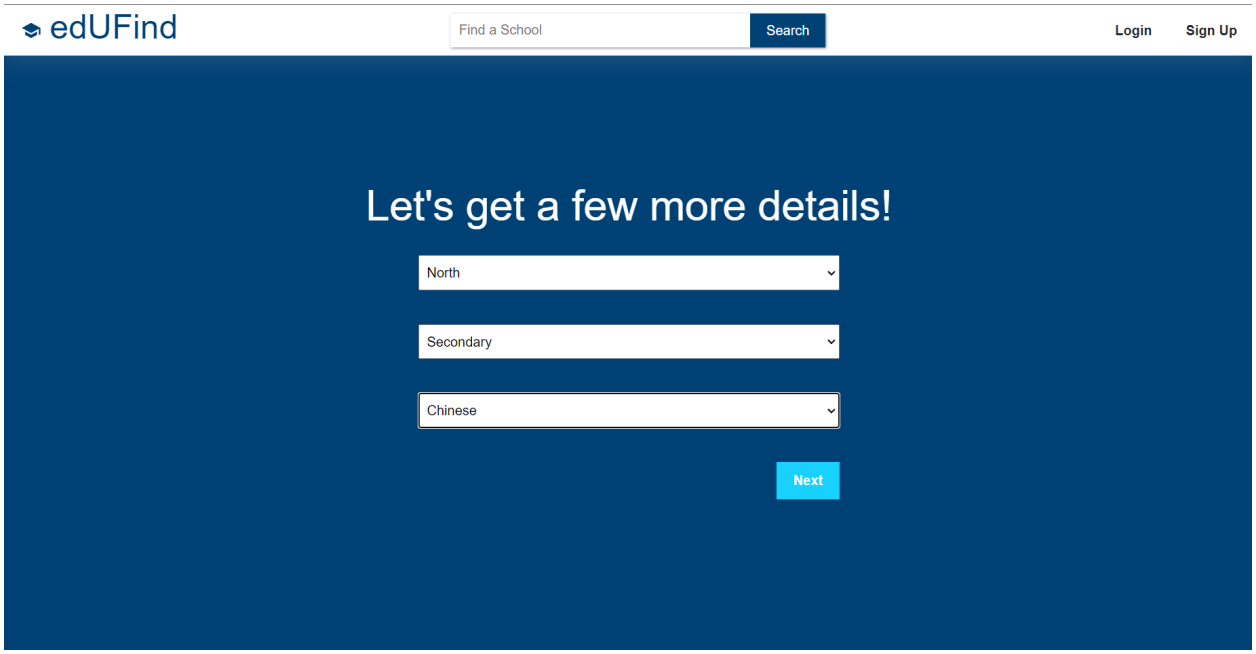


Fig 2e. Page 2 of User Account Registration

- After the user completes the first phase of the registration process, they will reach Fig. 2e.
- In Fig. 2e, the system will present dropdown fields to the user to collect input that will later be used by edUFind to provide personalized school recommendations to the user.
- The first dropdown is for the user to select a geographical region. The user is expected to select the region where they stay so that the system can recommend schools close to their area of residence.
- The second dropdown requires the user to select either their current educational level or the educational level that they are progressing to (in the instance that they're going from one level to the next). The system will use this information to recommend schools at the appropriate education level for the user.
- Lastly, the third dropdown requires the user to select their mother tongue language. Similar to the previous two options, the system will use this information to recommend schools to the user that offer the study of the user's selected mother tongue language
- Once the user is ready to proceed to the final phase of the registration process, they can click on the "Next" button.

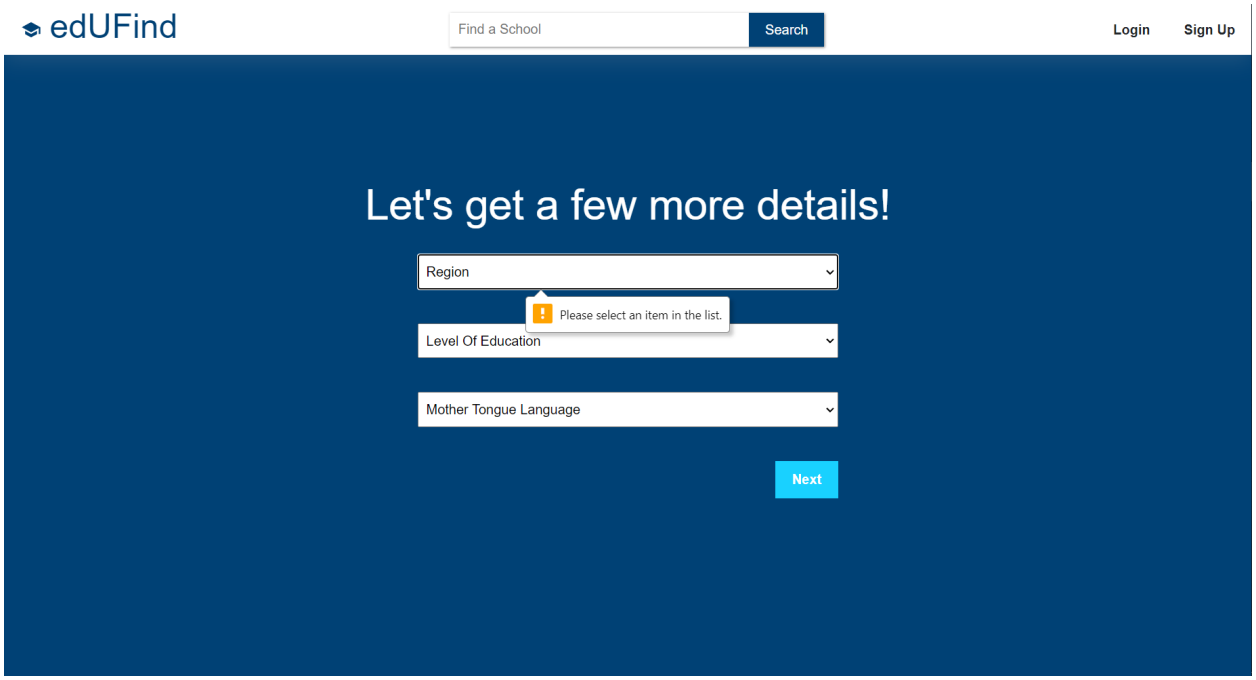


Fig 2f. Error when no option for Region is specified

Description:

- Continuing from Fig. 2e, if the user did not select an option for each of the three dropdowns, an error message, positioned below the corresponding dropdown box, will prompt the user that they must select an item for that particular field.
- The user will not be able to proceed to the final phase of the registration process if they do not select an option for each of the three dropdowns.

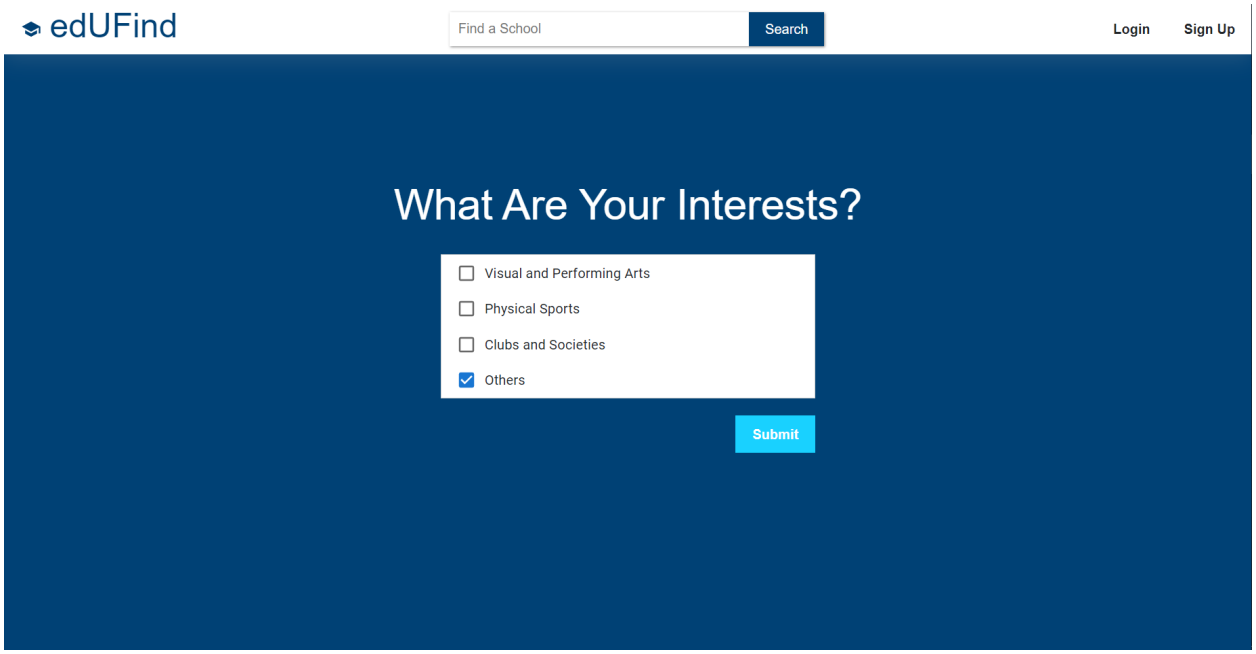


Fig 2g. Page 3 of User Account Registration

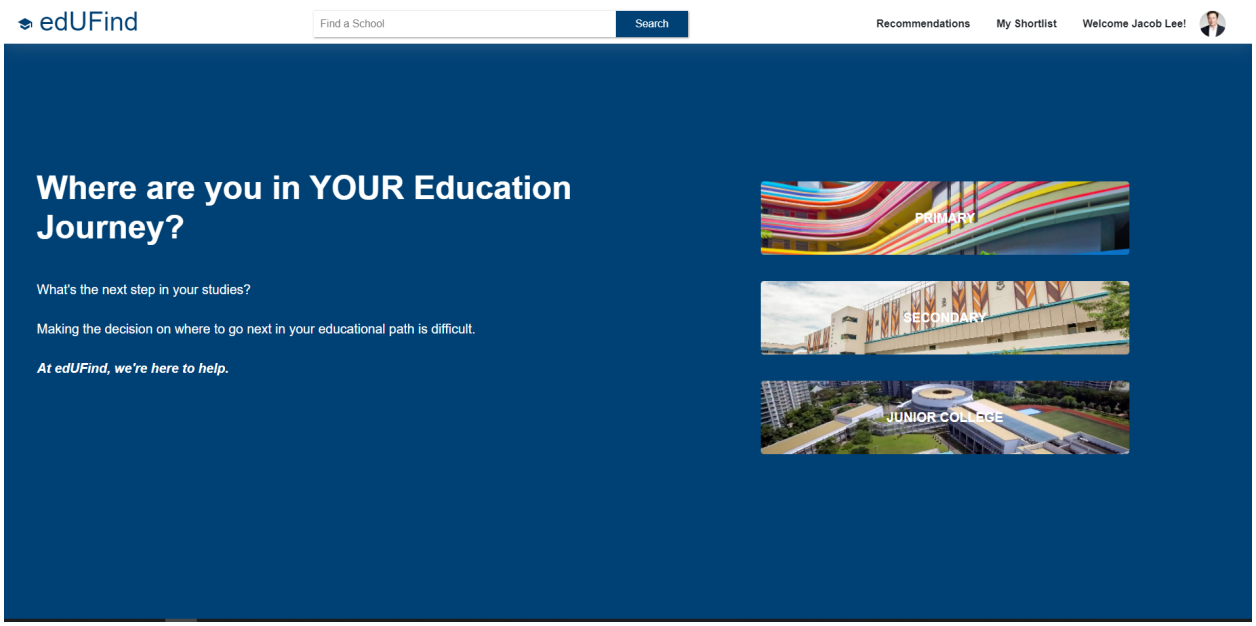


Fig 2h. After submission

Description:

- In the last phase of the registration process, the user is required to select the categories of CCAs in which they are interested.
- There are no limitations imposed on the number of categories of CCAs the user can indicate interest in. There is also no requirement to select at least one option. As such, the user is free to select as little as no option and as many as all options.
- Once the user is ready, they can click the “Submit” button to finalize registration of their edUFind account.
- Upon clicking the “Submit” button, the user will be routed back to the Home Page (Fig 2h.) and the Navigation bar will now show that the user is automatically logged in after creating an account.

3.1.3 Login

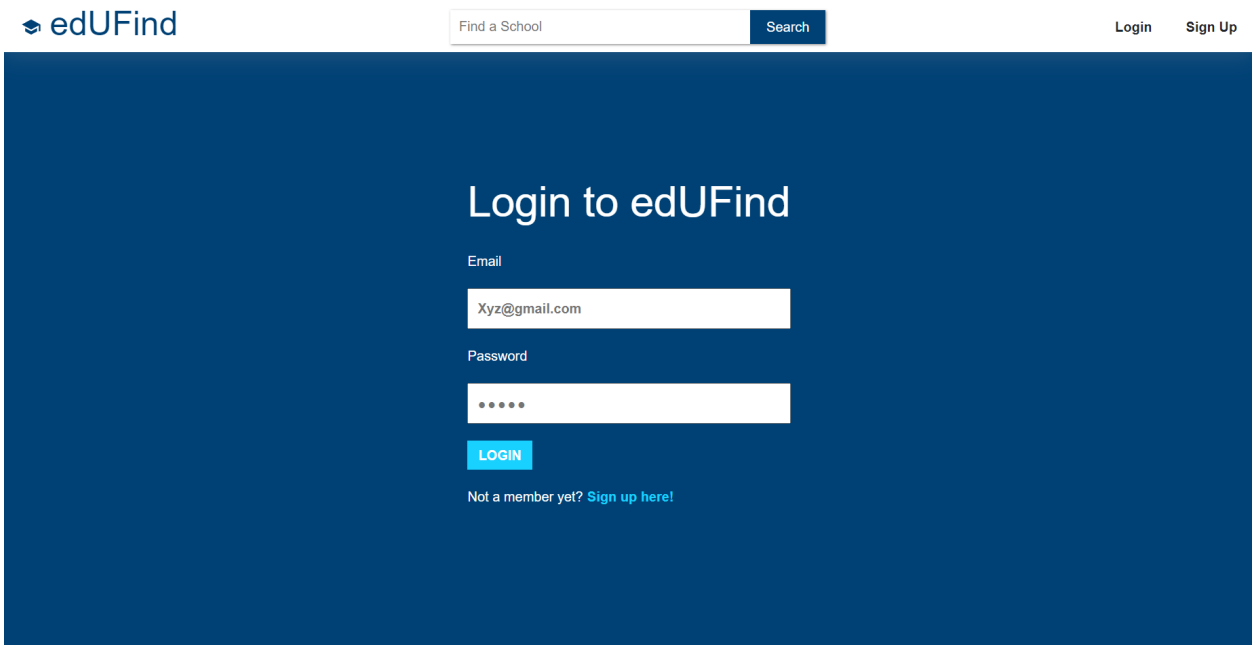


Fig 3a. Login Page

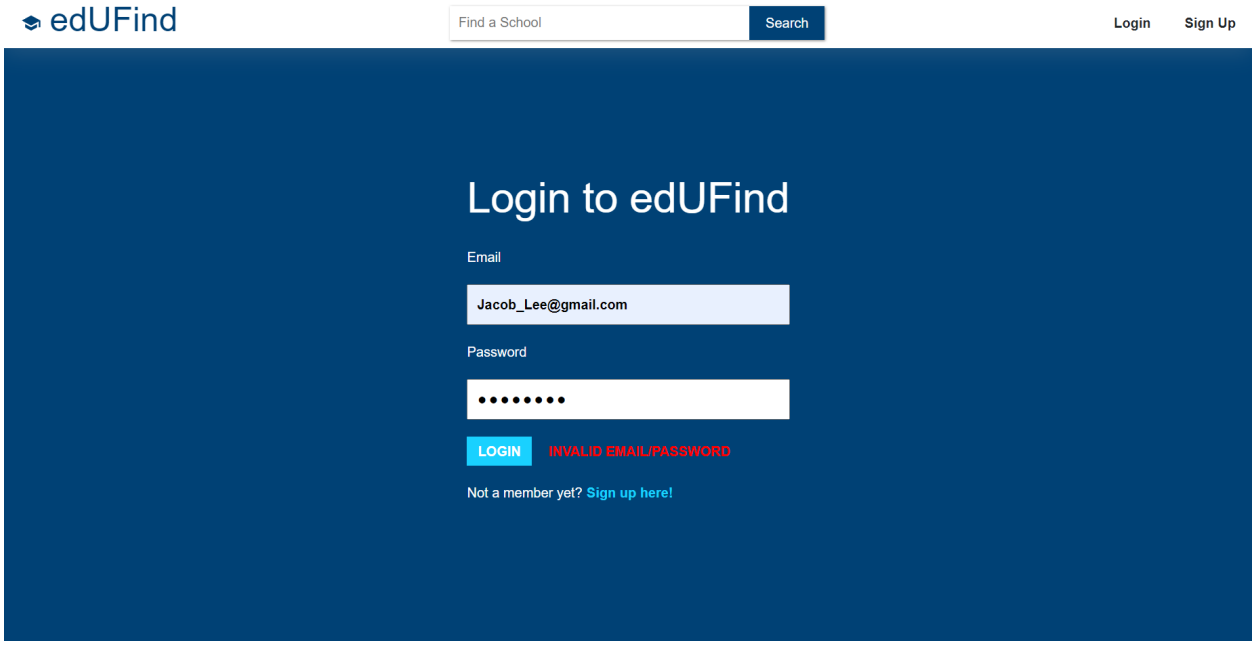


Fig 3b. Login Invalid input

Description:

- This page requires users to login by entering the email address and password combination associated with their edUFind account. Users then click the ‘LOGIN’ button to attempt to log in.
- If a user enters an invalid email address and password combination, edUFind will display an error message to notify them of the same.
- For Unregistered Users, they can click on the ‘Sign up here!’ link to navigate to the ‘Registration Page’ in Fig. 3a.

3.1.4 School List Page

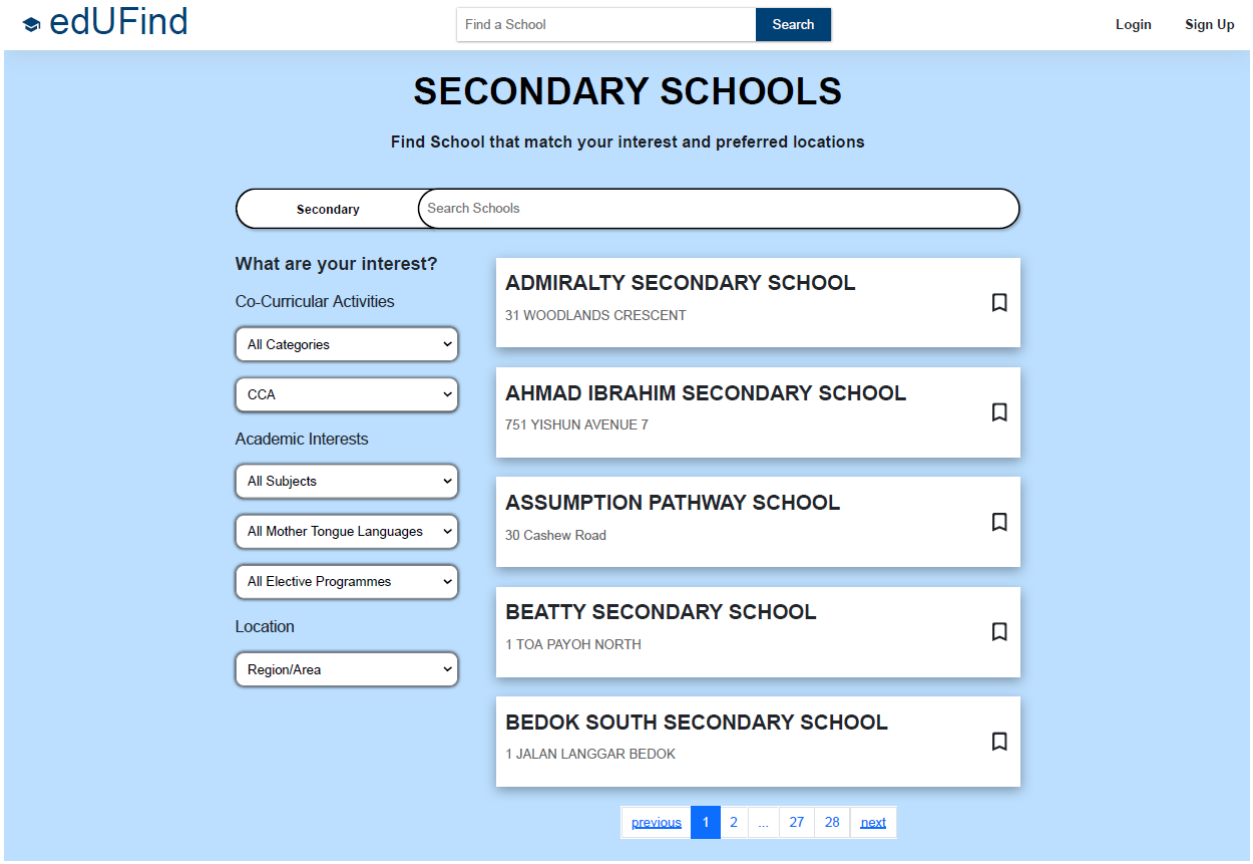


Fig 4a. School List Page

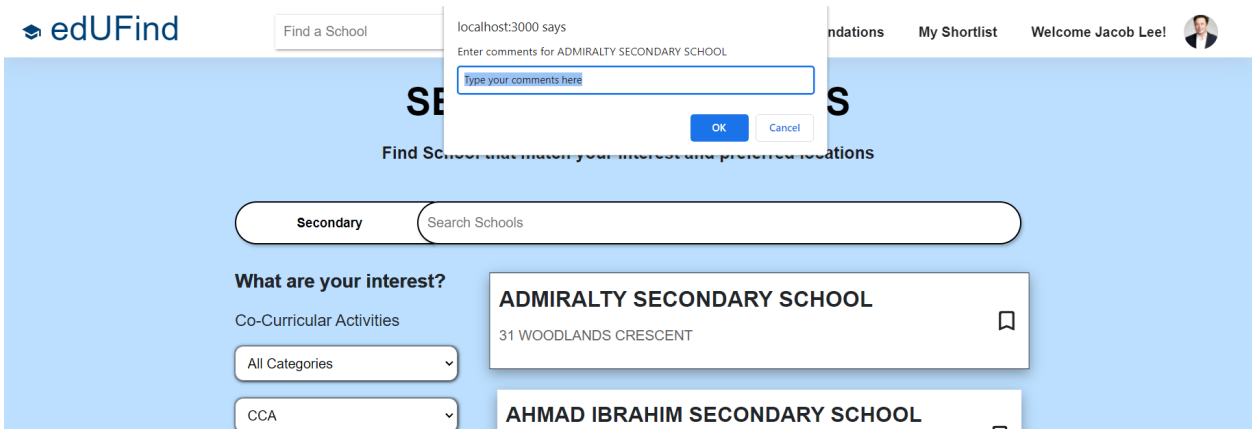


Fig 4b. School List Page Shortlisting

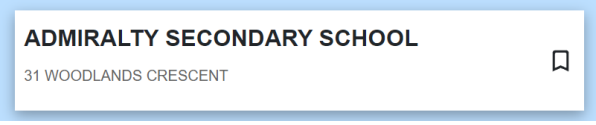


Fig 4c. Before Shortlisting

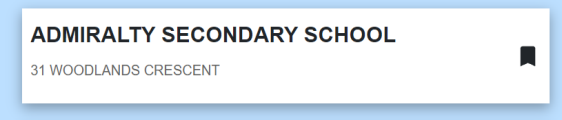


Fig 4d. After Shortlisting

Description:

- Upon clicking on one of the education level containers on the home page, users are directed to Fig. 4a, a page displaying the list of schools that cater to the respective education level. Each page contains 5 schools and users can click on the pagination component below to navigate to the other pages.
- Here, users can perform the following actions:
    - Users can search for schools using the search bar provided.
    - Users can sift through schools using the various filters on the left-hand side. Users can filter by:
        - CCA Category
        - Specific CCA
        - Subject
        - Mother Tongue Language
        - Elective Programme
        - Location
    - Registered Users can shortlist a school by clicking on the bookmark icon on the respective school card. Users will be provided the option to enter their own remarks to indicate the reason as to why they are bookmarking the particular school (Fig. 4b). Successful shortlisting results in the color of the bookmark changing from white to black (Fig. 4d).
    - Registered users can also click on the bookmark icon of an already shortlisted school to remove the school from their shortlist.

3.1.5 Search Results Page

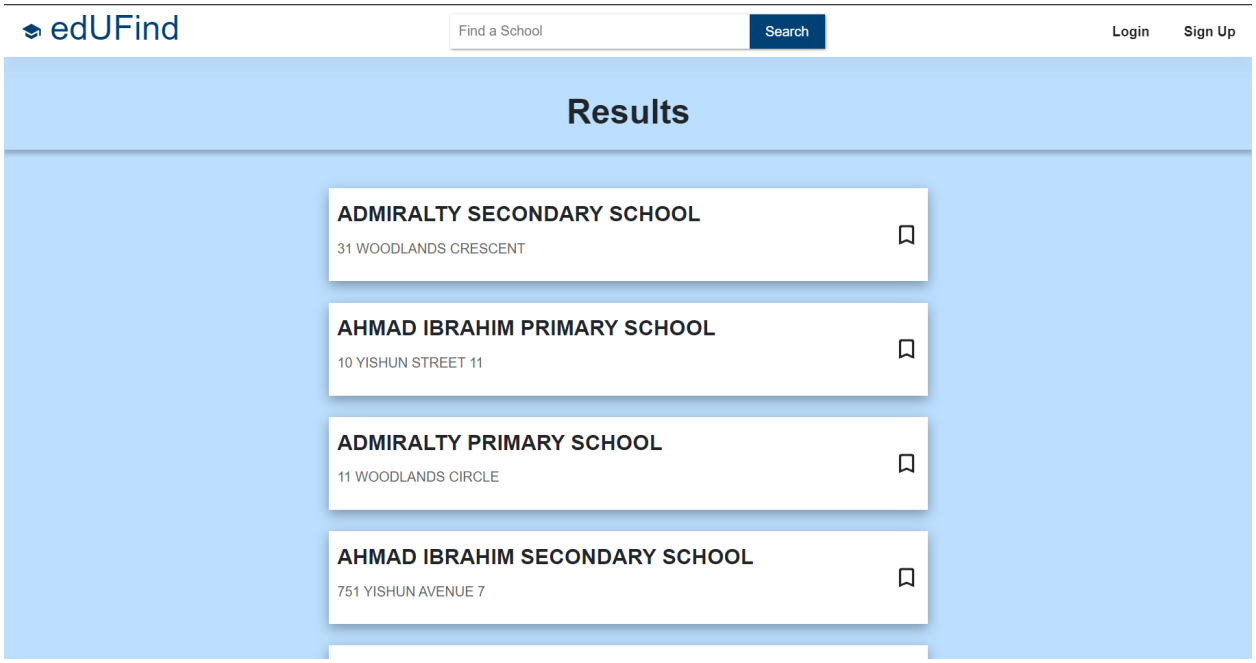


Fig 5. Search Results Page

Description:

- Upon searching for a school using the search functionality built into the navigation bar, users will be directed to a search result page similar to Fig. 5.
- Each page of the search results displays 10 schools and users can change pages by either clicking on a page number to directly navigate to that specific page or on the arrows at the bottom to go from one page to the prior page or the next page in an incremental manner.

3.1.6 School Profile Page

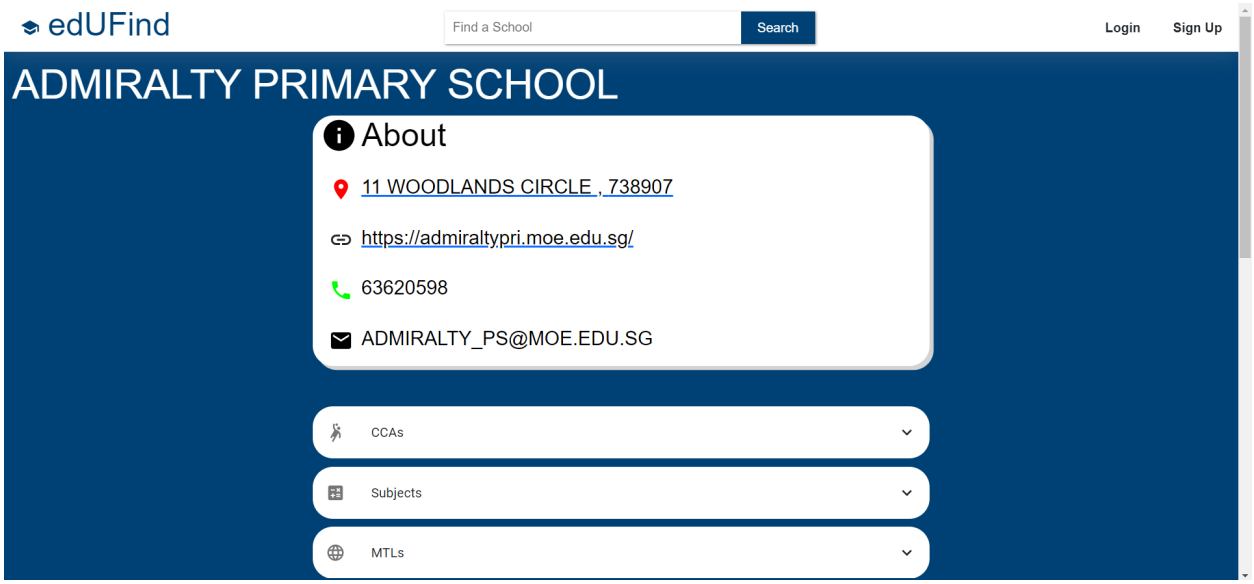


Fig 6a. School Profile Page (Details)

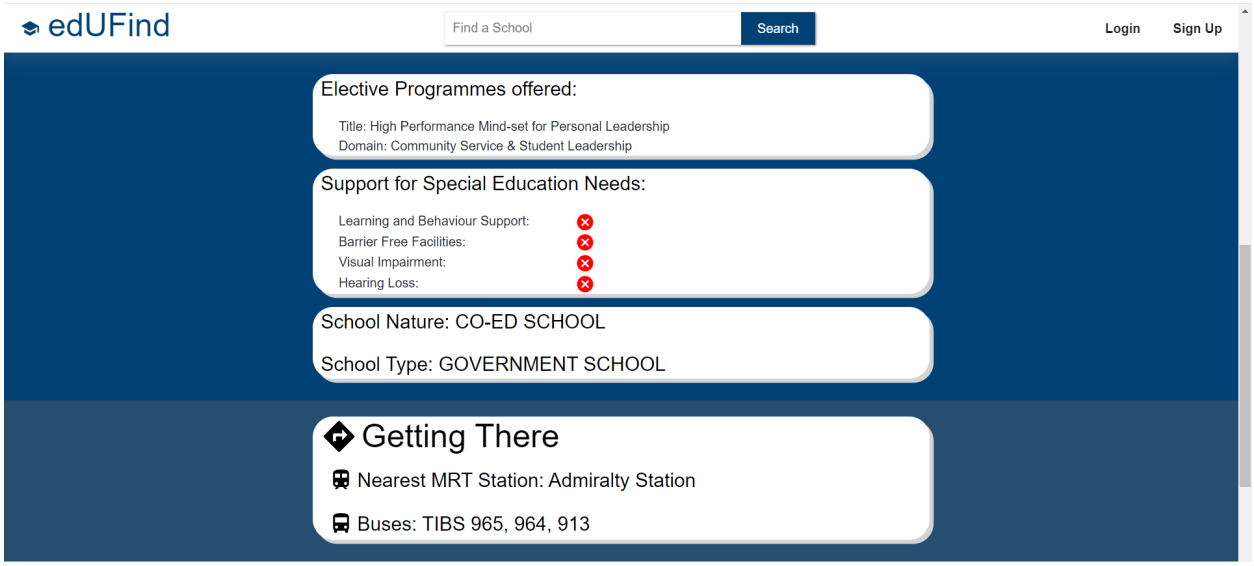


Fig 6b. School Profile Page (More Details)

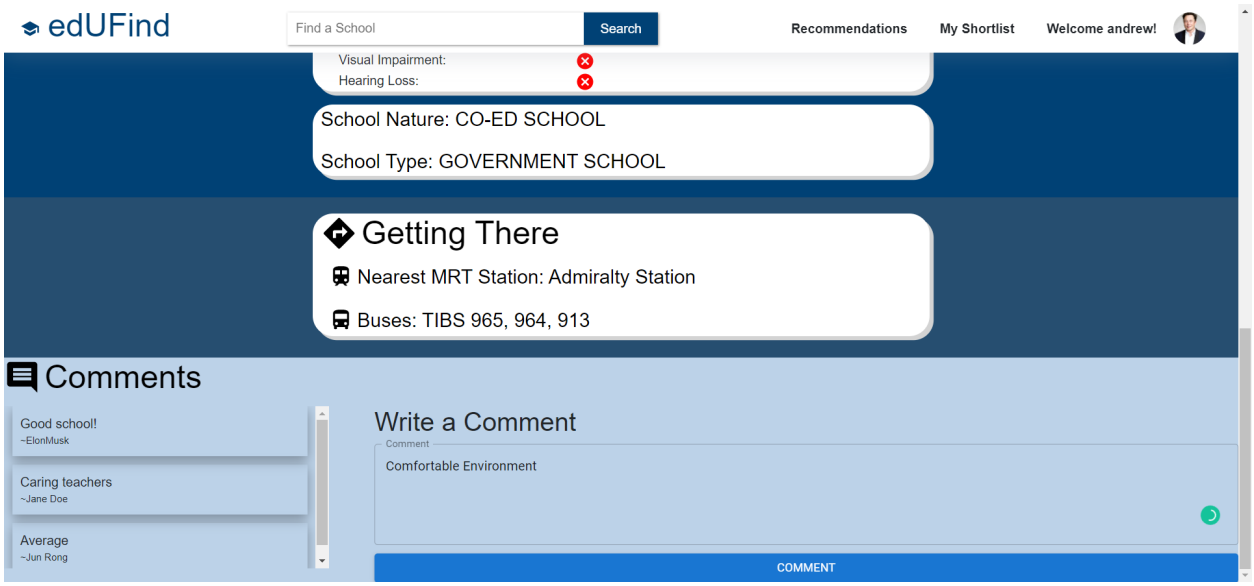


Fig 6c. School Profile Page (Comments Section)

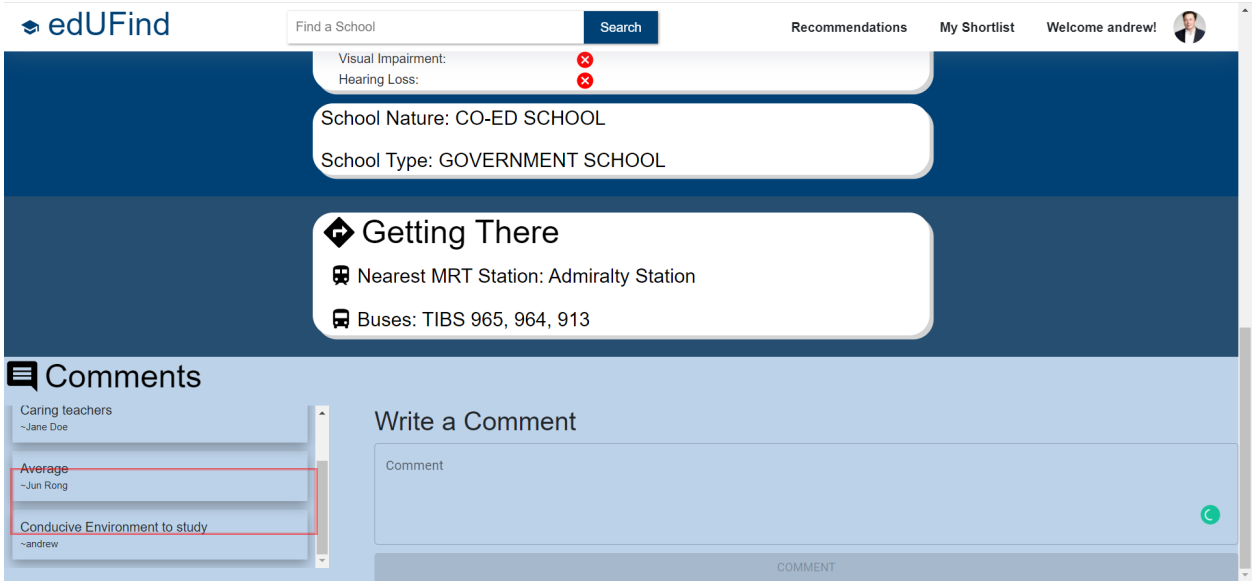


Fig 6d. School Profile Page (Comments Posted)

Description:

- Upon clicking on one of the school cards, the user will be routed to the school details page for that particular school. Fig. 6 shows the school details page, the page that the user is routed to when they click on a school card which is the card for “Ahmad Ibrahim Primary School” in this example.
- On this page, information about the school is displayed to the user. Information such as the various CCAs available at the school, the different elective programmes that the school offers, and more, are available on this page. More details are shown in Fig 6a and Fig. 6b above.
- At the bottom of the page, as referenced in Fig. 6c, there is a comment section for users to type and post comments about the school in question. Users must be logged in to edUFind in order to be able to write a comment for a school, however that requirement does not apply in order to view comments that have already been posted.
- As seen in Fig. 6d, once the user clicks on the “COMMENT” button, the comment will be posted on the comment forum, at the bottom left of the page.

3.1.7 User Account Page

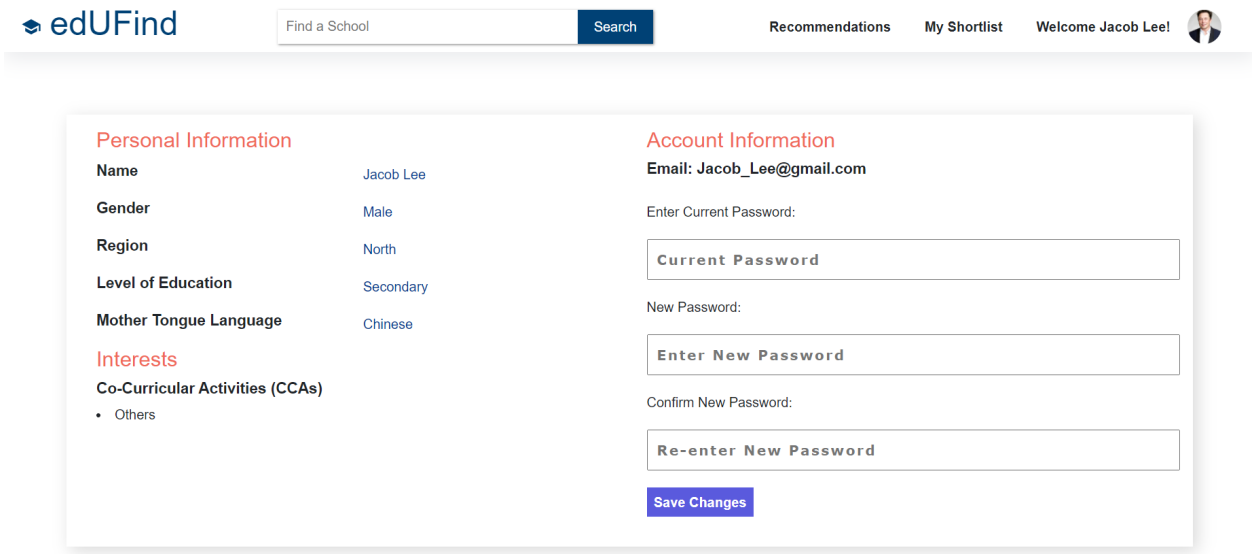


Fig 7a. User Account Page

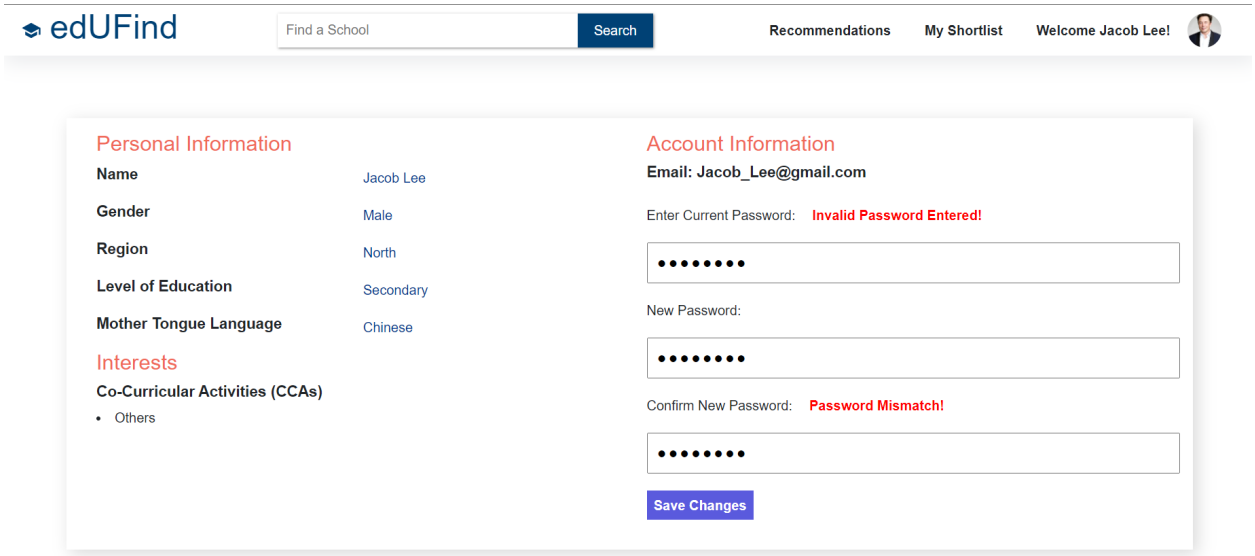


Fig 7b. User Account Page Error Indication

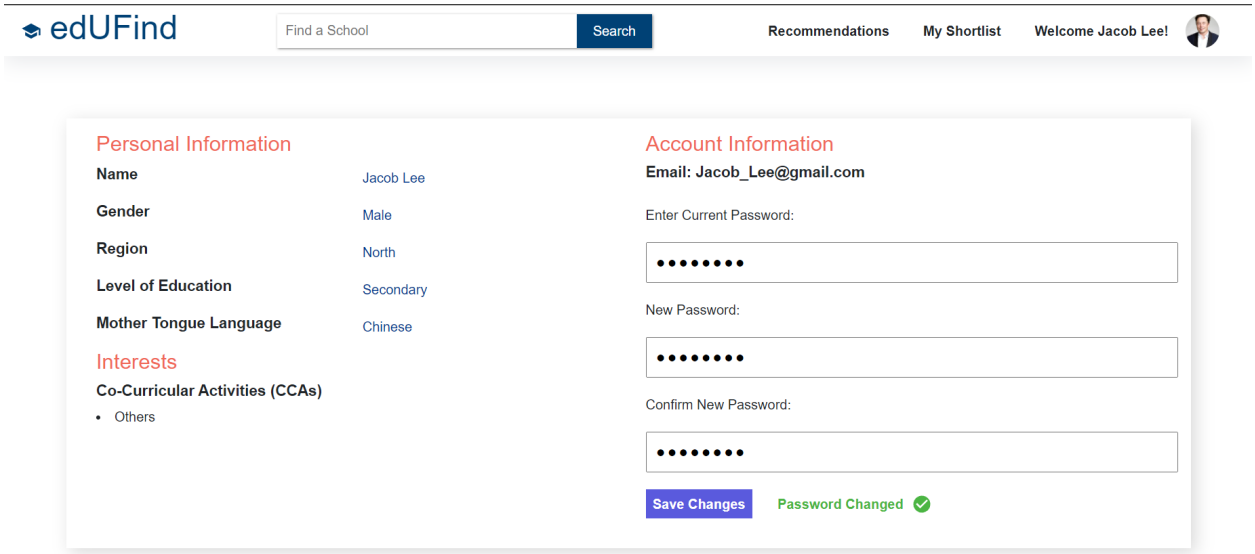


Fig 7c. User Account Page change success

Description:

- Registered users (that are currently logged into their accounts) can view their account information.
- Registered users can change their account password by entering their current password, desired new password, and their desired new password in the text fields on the right-hand side of the page.
    - Checks will be performed in the background to ensure that the user has entered the correct old password and that the input entered into the “New Password” and “Confirm New Password” text fields are identical.
    - If the changing of the account password was successful, edUFind indicates that to the user by displaying a success message.

3.1.8 Shortlisted Schools Page

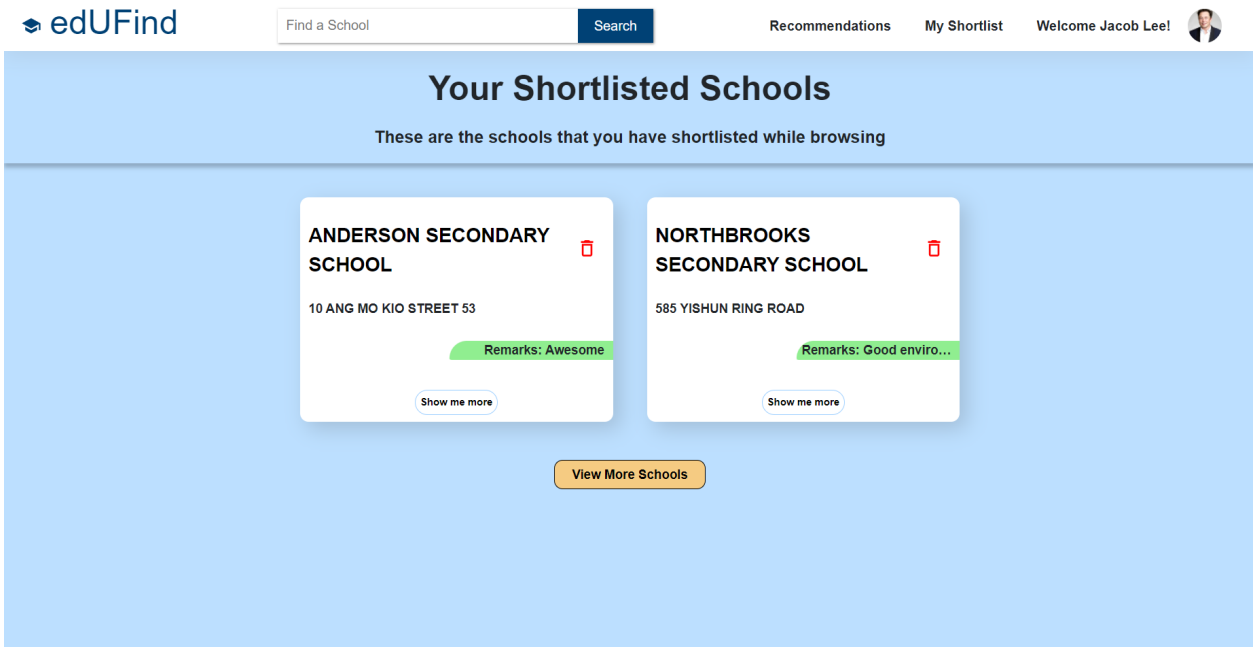


Fig 8a. Shortlist Page

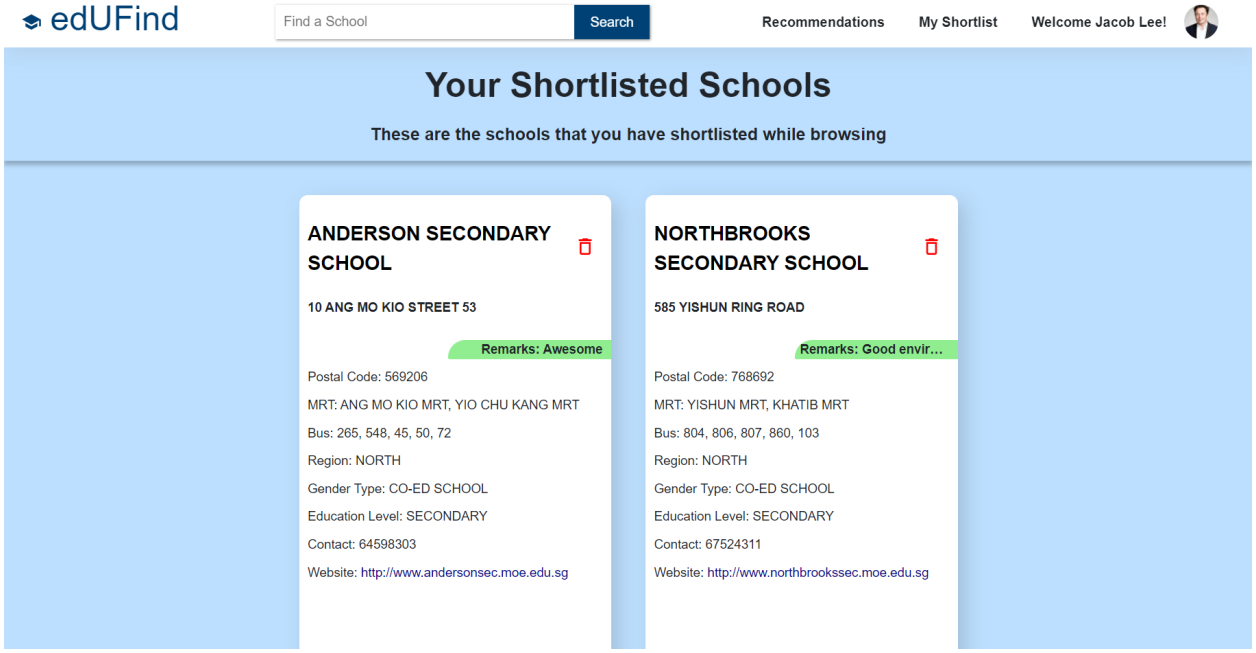


Fig 8b. Shortlist Page Expanded

Description:

- Registered Users can view their shortlisted schools and their corresponding remarks as to why they shortlisted a particular school.
- The cards can be expanded to view more details about the school at a glance. This makes it straightforward for users to perform a side-by-side comparison of a few of their shortlisted schools.

3.1.9 Recommendations Page

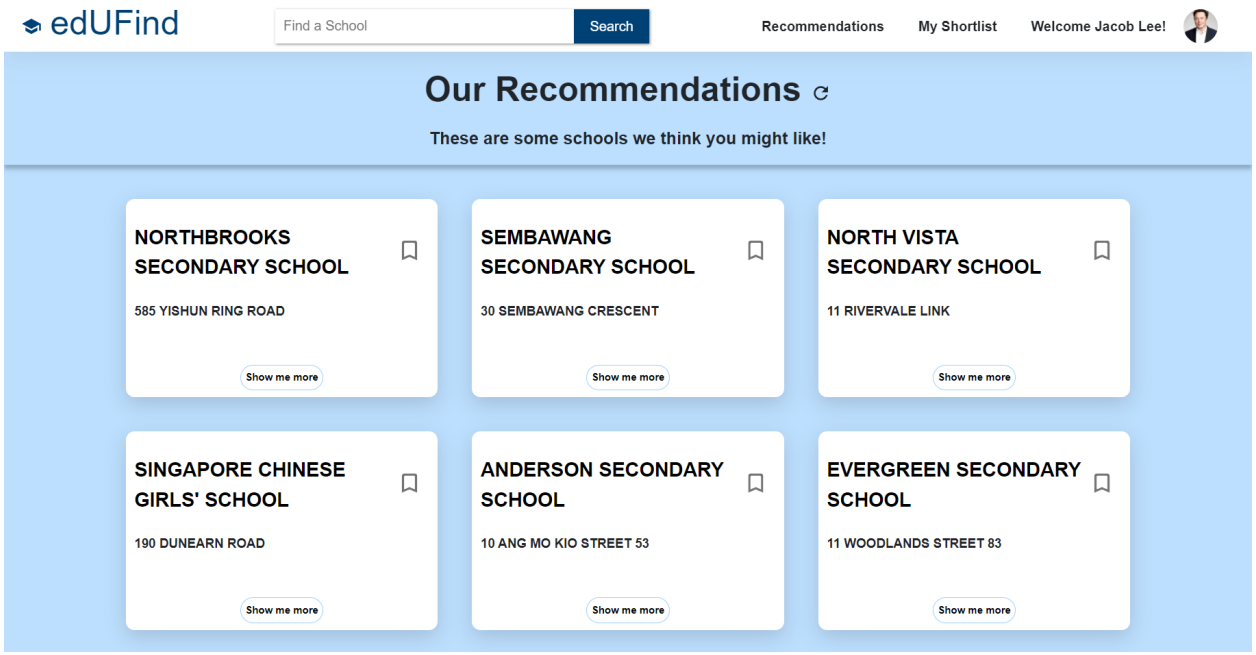


Fig 9. Recommendations Page

Description:

- Based on the user's interests, mother tongue language, residential region, and gender, edUFind will recommend nine schools to them.
- Users can click on the bookmark icon on a school's card to shortlist that particular school.
- Users can click on the refresh button next to 'Our Recommendations' title that refreshes edUFind's recommendation list, thereby presenting the user with a new assortment of recommended schools.

3.2 Hardware Interfaces

1. Desktop Computer: Presentation Layer
    a. Interaction with edUFind by using the keyboard to enter textual input and the mouse to navigate across the various pages and user interface artifacts of the web application.
2. AWS: Application Logical Layer
    a. AWS hosts the backend server, allowing communication between edUFind and backend databases.
    b. Frontend is also hosted in AWS.
3. MongoDB Database: Persistent Data Layer

    a. Storage of edUFind's permanent user data.

3.3 Software Interfaces

1. Data.gov.sg: Fetching of relevant school information.

2. Axios: API calls made by the frontend to interact with the backend server to retrieve, update, insert, or delete data.

3.4 Communications Interfaces

1. HTTP requests are used for communication between the frontend client and the backend server.
2. NGINX is used as a reverse proxy server to route HTTP GET, POST AND PUT requests from the client requests to the intended port on the backend side.

3. PM2, a Daemon Process Manager for Node.js applications, is used to keep our web application online.

4. System Features

4.1 Registration

4.1.1 Description and Priority

Executes the registration procedure to create a user account in the database. Users are prompted to enter their personal details to create an account.

High priority.

4.1.2 Stimulus/Response Sequences

User action: User enters account details which includes username, email, password and gender. Once done filling up the fields, the user clicks the “Next” Button.

System Response: Validate syntax of email address. Validate whether input for “password” and “confirm password” fields match. Once validated, personal information is entered into the database.

User action: User selects the region, education level and the mother tongue language they intend to take from 3 dropdown menus. Once done, the user clicks the “Next” button.

System Response: Saves personal information entered into the database.

User action: User chooses which categories from 4 different possible categories of CCAs. User clicks “Submit”.

System Response: Saves personal information entered into the database. The user is now routed to the homepage.

User action: From the navigation bar, the User clicks the “Recommendations” tab.

System Response: Moves over to the “Recommendation” page. Retrieves and displays a list of schools from the database that have details that align with the user’s personal information.

4.1.3 Functional Requirements

REQ-1: The application must be able to detect an invalid email address by checking if the input contains ‘@’ or ‘.com’.

REQ-2: The application must be able to detect an invalid email address by checking if the input length exceeds 300 characters.

REQ-3: The application must be able to search the database for email addresses associated with existing accounts within 5 seconds. This is to validate whether or not the email address entered by the user during the registration process is already used with another pre-existing edUFind account.

REQ-4: The application must be able to identify a mismatch in the “password” and “confirm password” fields while the user is populating these fields.

REQ-5: The application must be able to identify if the password entered by the user meets the minimum length requirement of 6 characters and does not exceed the maximum length limit of 50 characters.

4.2 Login

4.2.1 Description and Priority

Execute the login procedure to establish a connection between the account and the server. Retrieve the account information associated with the user such as personal details and shortlisted schools.

High priority.

4.2.2 Stimulus/Response Sequences

User action: Enters login information.

System Response: Validates login information. Produces an error if the user has entered invalid login information. Alternatively, route the user to the Home Page if the login information is valid and login is successful.

User action: From the navigation bar, User clicks the “My Shortlist” tab

System Response: Routes to “My Shortlist” page. Retrieves the list of schools previously shortlisted by the user from the database and displays it.

4.2.3 Functional Requirements

REQ-1: When the user tries to login, the application must be able to respond with either a successful login or produce a notification of a login error within 5 seconds.

4.3 Search and Filters

4.3.1 Description and Priority

Search for schools based on the user's search query and filter based on the user’s filter settings. School information can be accessed by clicking on the school.

High Priority.

4.3.2 Stimulus/Response Sequences

User action: Enters a search query.

System Response: Upon detecting a change in the query input, schools whose name is a superset of the query are displayed to the user.

User action: Selects one or more filters.

System Response: When one or more filters are selected, schools that fit the filters are displayed to the user.

User action: Enter a search query and select one or more filters.

System Response: Schools, which fit the filters selected and concurrently have their name as a superset of the search query, are displayed to the users.

User action: Clicks on a specific school card.

System Response: Routes the user to the specific school page that contains details about the school whose UI card was clicked on.

4.3.3 Functional Requirements

REQ-1: After a user enters a valid search query or selects a filter, schools that fit the search and/or filtration criteria should be displayed in no more than 10 seconds.

REQ-2: While the system attempts to retrieve the school data, a loading icon is used in order to keep the user informed about the loading progress.

REQ-3: If the search query and/or filter will return no corresponding results, then the user should be notified in less than 5 seconds.

REQ-4: When the user clicks a school for more details, they must be routed to the correct school details page within 5 seconds.

4.4 Shortlisting

4.4.1 Description and Priority

Registered users can shortlist schools they are interested in by clicking on the ‘bookmark’ button. This can help them keep track of which schools they are most interested in and may potentially enroll at.

High priority.

4.4.2 Stimulus/Response Sequences

User action: Clicks on the ‘bookmark’ button of a school card.

System Response: On the click of the ‘bookmark’ button, the system prompts the user to insert a comment. Following that, the system will then save the selected school into the shortlist database for that registered user

---

User action: Selects ‘Shortlist’ tab on the top navigation bar

System Response: User is directed to the ‘shortlisted schools’ page which displays concise information of previously shortlisted schools. Comments are also displayed

---

User action: Clicks on a ‘bin’ icon of a school on the ‘shortlisted schools’ page

System Response: Selected shortlisted school is deleted and removed from the database of that user

---

User action: Clicks on a ‘show me more’ button of a school on the ‘shortlisted schools’ page

System Response: Expands information of the selected school to show more information

---

User action: Clicks on a specific school on the ‘shortlisted schools’ page

System Response: Routes to the specific school page that contains details about the school

4.4.3 Functional Requirements

REQ-1: After a user clicks on the ‘bookmark’ button, the system must prompt the user for a comment within 1 seconds.

REQ-2: After a user inputs a comment and clicks ‘add’, the selected shortlisted school must be saved into the database within 5 seconds.

REQ-3: On the ‘shortlisted schools’ page, deletion of a shortlisted school must be updated within 3 seconds.

REQ-4: On the ‘shortlisted schools’ page, the list of user’s shortlisted schools must be loaded within 3 seconds.

REQ-5: When the user clicks a school for more details, they must be routed to the correct school page within 5 seconds

4.5 Recommendations

4.5.1 Description and Priority

Registered users can view a list of 9 recommended schools by the system at a time. This list of schools is recommended based on the inputted fields of the user during registration. High priority.

4.5.2 Stimulus/Response Sequences

User action: Selects ‘Recommendations’ tab on the top navigation bar

System Response: User is directed to the ‘recommended schools’ page which displays concise information of recommended schools by the system.

User action: Clicks on a refresh arrow next to the header of the ‘recommended schools’ page

System Response: System will refresh the list of recommended schools to display other recommended schools

---

User action: Clicks on a ‘bookmark’ button of a school on the recommended schools’ page

System Response: Steps of shortlisting the school is carried out (3.1.4)

User action: Clicks on a ‘show me more’ button of a school on the ‘shortlisted schools’ page

System Response: Expands information of the selected school to show more information

---

User action: Clicks on a specific school on the ‘recommended schools’ page

System Response: Routes to the specific school page that contains details about the school

4.5.3 Functional Requirements

REQ-1: After a user clicks on the refresh button, the system must suggest more recommended schools within 3 seconds.

REQ-2: After a user inputs a comment and clicks ‘add’, the selected shortlisted school must be saved into the database within 5 seconds.

REQ-3: On the ‘recommended schools’ page, the list of user’s recommended schools must be loaded within 3 seconds.

REQ-4: When the user clicks a school for more details, they must be routed to the correct school page within 5 seconds

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- The web application system database must be able to store a user's shortlisted schools for at least a year.
    - Enables users to track their shortlisted schools and their comments on them.
- The web application system database must be able to store 1000000 user accounts' personal information.
    - Supports multiple users using the web application in its lifetime.
- The web application must be able to support 5000 concurrent active queries and filters by users and still respond in at most 5 seconds.
    - Ensures that users do not experience a time lag between their input and their result; better user experience.

5.2 Safety Requirements

- edUFind must not display any advertisements that could potentially pose harm to children aged 12 to 16.
- edUFind must store minimal user's information. Those that are deemed as necessary. All other user information should be optional or otherwise not required.

5.3 Security Requirements

- Account
    - Account must be tagged to the user
    - Each user must be identified by MongoDB ObjectID which is unique and unretractable
    - Users should only shortlist their preferred schools only when they have an account
- Account Information
    - Users can access their personal details and shortlisted schools only if they are logged into their own account.
- Account Password
    - Minimum length for a valid password is 6, and a maximum length of 50
    - Users must able to edit their password if they wish to do so
- Password Encryption
    - Administrator and malicious intruder must not be able see nor unhash the password.
    - A breach of security must not allow the intruder to trivially learn all user passwords.
    - Hashed passwords must be computationally expensive to crack it.
- Token for authentication
    - A middleware must be used to authenticate the user, whenever they do an user-only operation.
    - The token must be unique and should expire in a set duration of time to prevent any misuse of stolen JWT token.
    - Upon the expiration of the JWT token, the system must prompt the user to re-login.
    - The token must be stored safely and not exposed in the browser.

5.4 Software Quality Attributes

- Usability
    - The web application has a intuitive design and is easy to navigate
    - Users are notified of errors due to their actions, e.g. Password mismatch, Loading Schools, No Schools Found
- Reliability
    - The web application must be able to operate normally after a reset
    - User must not be logged out of his account during page change or page refresh
    - The web application must not crashed at any point in time
- Maintainability
    - The web application should not require maintenance more than once a month
    - Web application functions must be modularized for easy maintenance and upgrading
    - Web application software should be designed according to industrial standards and properly documented for easy handover if necessary

5.5 Business Rules

- Any information related to the user's privacy should only be accessible by the owner.
- As user accounts are identified by MongoDB ObjectID, and the password is encrypted, the administrator will not be able to access nor identify the user's account information.
- Users of our web application need not be required to create an account to use the web application.

6. Other Requirements

6.1 Target Audience

Our product only targets Singaporeans seeking local education, and hence schools shown are those in Singapore. Current language of the application is only English, Singapore's language of communication.

6.2 Design Patterns

6.2.1 MVC Pattern

Web applications in general are event intensive applications. These applications if not designed appropriately can result in tight coupling. As such, there is a need to classify the code into 3 main categories. They are named as model, view, and controller. Implementing the MVC pattern in edUFind helps separate the logic from the user interface. In doing so, low coupling can be achieved among system components.

In addition, separating them into the 3 main categories allows for high cohesion. All related logic and their associated views can be structured together as one subsystem. With low coupling and high cohesion, the failure or change of one subsystem of the application would unlikely to affect another subsystem.

An example of a subsystem in edUFind is shown below. It illustrates MVC pattern when searching and filtering a list of schools under the SchoolsUI page.

Model:

Entity classes that hold information: Schools. This entity contains a database of all MOE schools in Singapore. This includes Primary, Secondary and Junior College levels.

Inputs selected by the users are processed by functions on the front end to sort the array of schools. Only schools containing values in which the user selected will be returned, thus affecting the information being displayed on screen for the user.

Views:

Presents all the school related data to the users in the following pages:

- School Details Page
- List of Schools Page

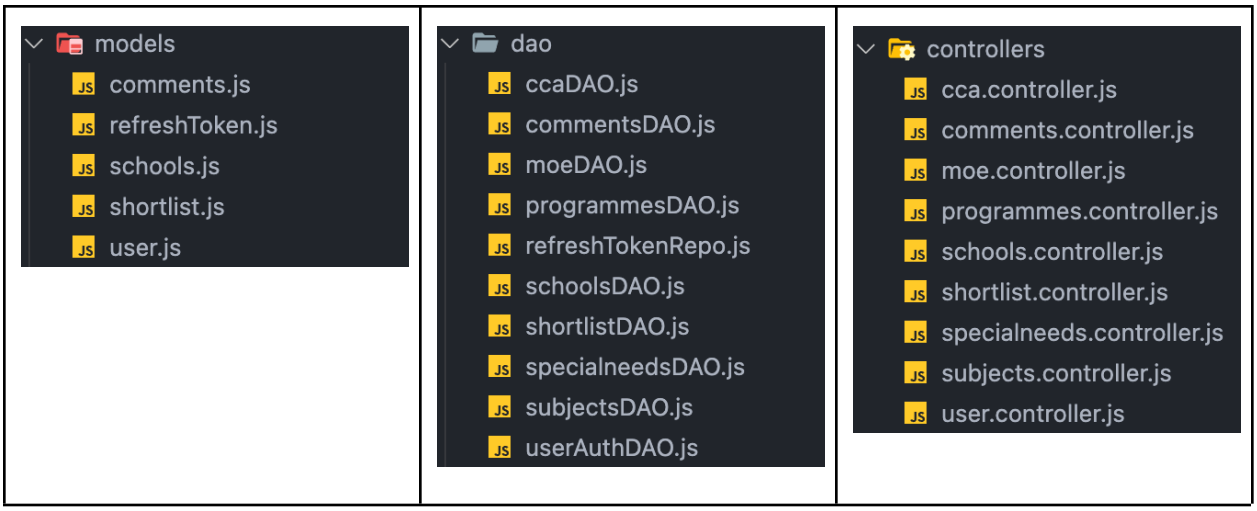
Data displayed on the screen dynamically changes based on the users' inputs.

Controllers:

Displays the interfaces for users to interact with:

- Search field: A input field filled by user to filter a list of schools based on a query
- Filter options: A select field consisting of options for users to filter a list of schools based on characteristics of a school such as:
    - CCA Category
    - CCA
    - Mother Tongue Language
    - Region
    - Elective Programmes

6.2.2 Data Access Object Pattern



We separated the client interface from the data access mechanism in our implementation.

Take a school dataset, for instance, a school object acts as a Model or Value Object. It serves as an interface that defines the standard operation on a model object defined using MongoDB modeling.

schoolDAO.js is the concrete class implementing Data Access Object Interface. It implements the above interface, schools.js. It will fetch data from MongoDB.

Schools.controller.js will use schoolDAO.js to demonstrate the use of Data Access Object pattern. It contains the get/post methods to store and retrieve data using DAO class.

Data Access Object Pattern allows the data access mechanism to change independently of the data's code. It separates and decouples persistent mechanisms from the rest of the application, advocating the Single responsibility principle.

6.2.3 Composite Pattern

Our UI Pages are broken up into reusable components such as:

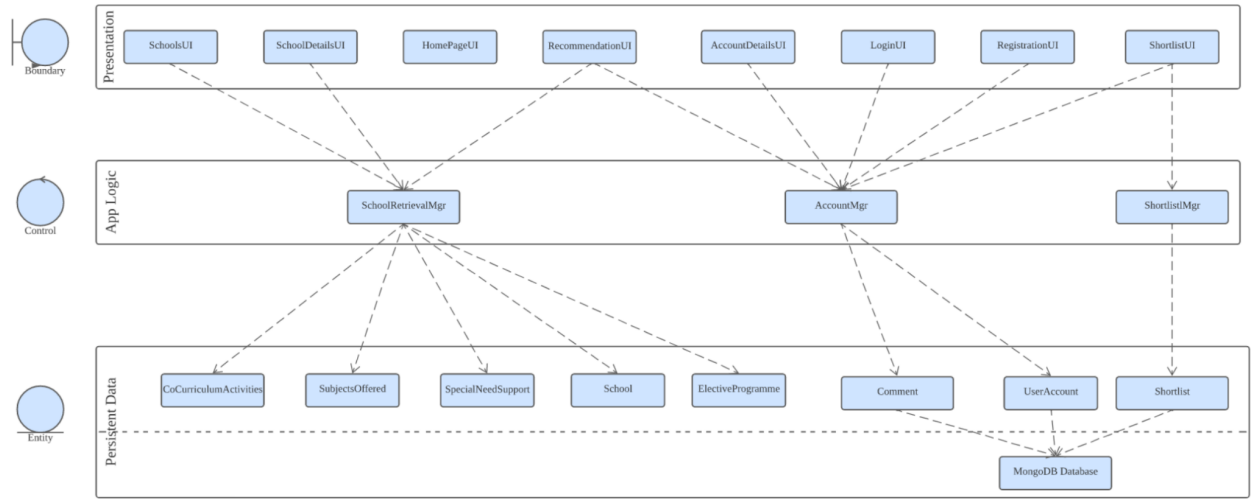
- Schools
- School
- Navbar
- And more

Since Navbar is a component, it can be applied to any of our UI pages simply by calling it, thus rendering it on the page itself. Similarly, our ‘School’ component is called upon by ‘School’s, allowing us to reuse the components in both the ‘Results Page’ and ‘School List Page’.

7. Appendix

Appendix A: Glossary

System Architecture



Data Dictionary

Term	Definition
User	A user is a person who is using the web application to find schools based on their interest and location in Singapore.
System	A system refers to the web application.
Search	Search is a feature that allows the user to find schools based on school's name or certain keywords.
Filter	The filter is a feature that the user can use to sort schools based on the school's characteristics (E.g., CCA, Subjects offered, etc.).
Shortlist	Shortlist is a feature that enables registered users to mark/save their preferred school. Users are also able to delete schools from their list or retrieve their list of shortlisted schools.
Registration	Registration is a feature for users to create a new account (provided

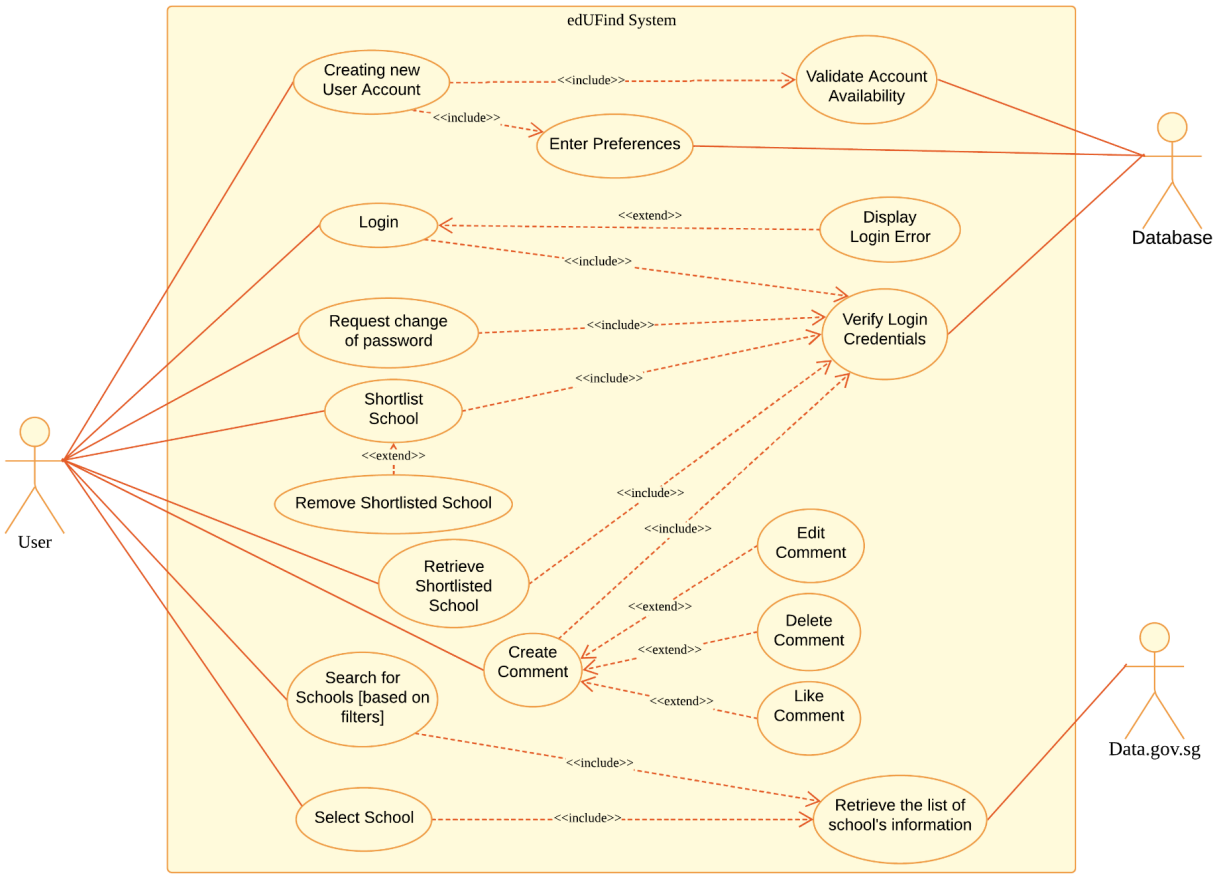
	their email is valid). This enables users to use other features of the website, such as shortlisting and commenting.
Registered User	A user that has created an account in the system and is logged into that account.
Comment	A written remark by a user to denote their views on a particular school. The author of the comment can delete or modify the comment while other registered users can 'like' comments.
Cut-off point	Refers to the minimum score required to enter a particular school based on previous year's data
Nature Code	Refers to the gender that the school allows to be enrolled. Values include "Co-Ed", "Boys only" or "Female only".
Type Code	Refers to the type of school. Values include "Government School", "Government Aided School", "Independent School", "Specialized Independent School" and "Specialized School".
Elective Program	Types of learning programmes that the school's adopt, such as Chinese Language Elective Program (CLEP), Art Elective Program (AEP), etc .
Education Level	The education requirement required to enter a school (Primary, Secondary and Junior College).
Main Level Code	Refers to education level reflected in the database. Values include "Primary", "Secondary", "Junior College", "Mixed Levels" and "Centralised Institute".
Co-Curricular Activities (CCA)	A non-academic activity that all students must undertake as part of their education There are 4 main categories of CCAs ● Clubs and societies ● Physical sports ● Uniformed groups

	• Visual and performing arts
Mother Tongue Language (MTL)	A mother tongue language is a second language (the first being English) that a student will learn in school (such as Chinese, Tamil and Malay).
Subject	The various knowledge studied or taught in a school that are currently offered to the enrolling student cohort.
Special Education Needs	Types of special needs the schools can support (E.g., dyslexia, ADHD, mild Autism Spectrum Disorder, mild hearing loss, visual impairment and physical impairment).

Appendix B: Analysis Models

Use Case Diagram

Refer to PDF File for the clear image



Use Case Descriptions

Use Case ID:	1		
Use Case Name:	Creating New User Account		
Created By:	Lim Kai Sheng	Last Updated By:	Andre Lim
Date Created:	28th January 2022	Date Last Updated:	22nd March 2022
Actor	User		
Description:	User creates a new account		
Precondition:	1. User account must not exist in the database 2. User must have an internet connection		
Postconditions:	1. User account must be updated in database 2. User password is encrypted in the database 3. User can now log in using the created account		
Priority:	Medium		
Frequency of Use:	0 - 1 time per lifetime		
Flow of Events:	1. User keys in a valid username, email, password, confirm password and gender in the respective fields 2. System verifies: a. Email is not already registered in the database b. The email entered is in a valid format c. Password and confirm password fields are identical d. The password is longer than 6 characters 3. The user clicks the “Next” button		
Alternative Flows:	AF-S3: System detects discrepancy between any of the verification 1. The system displays the respective error message		

	2. Return to Step 1
Exceptions:	-
Includes:	1. Validate Account Availability 2. Enter Preferences
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	2		
Use Case Name:	Validate Account Availability		
Created By:	Andre Lim	Last Updated By:	Ling Yin
Date Created:	28th January 2022	Date Last Updated:	22nd March 2022
Actor	System		
Description:	Only allow user to register an account if the email is valid		
Precondition:	1. User account must not exist in the database 2. User must enter an email in a valid format (for example: xxx@abc.com) 3. User must have an internet connection		
Postconditions:	1. User account must be updated in database 2. User can now log in using the created account		
Priority:	Medium		
Frequency of Use:	1 - 3 times per lifetime		
Flow of Events:	1. The system will validate the account availability in the database 2. The system will add the user account in the database		
Alternative Flows:	AF-S1: System detects the user account has already exists in the database 1. System will display error message “Account with corresponding email already exist” 2. User enters a new email 3. User clicks the “create account” button to re-attempt registration again.		

	4. Return to Step 1
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	3		
Use Case Name:	Login		
Created By:	Andrew Ng	Last Updated By:	Kai Sheng
Date Created:	28th January 2022	Date Last Updated:	22nd March 2022
Actor	User		
Description:	User logs into their created account using the registered email and password		
Precondition:	1. User must have a registered account in the database 2. User must have an internet connection		
Postconditions:	1. User has access to their account 2. User can use additional features of the website such as Shortlisting Schools, Commenting etc		
Priority:	Medium		
Frequency of Use:	1 - 3 times per lifetime		
Flow of Events:	1. User enters their email, and password in the login field 2. User clicks on the “login” button 3. System authenticates the account with the database		
Alternative Flows:	AF-S2: User left the required field empty 1. System will display an error message 2. User fills up the required field(s) 3. Return to Step 2 AF-S3: User enters an invalid email or password 1. System will display an error message 2. User re-enters the respective field(s)		

	3. Return to Step 2
Exceptions:	-
Includes:	1. Verify Login Credentials
Special Requirements:	-
Assumptions:	1. User will not forget their password
Notes and Issues:	-

Use Case ID:	4		
Use Case Name:	Display Login Error		
Created By:	Lim Kai Sheng	Last Updated By:	Andre Lim
Date Created:	28th January 2022	Date Last Updated:	22nd March 2022
Actor	System		
Description:	Displays “Invalid Email or Password entered” to the user		
Precondition:	1. User enters an invalid email or password when logging in 2. User must have an internet connection		
Postconditions:	1. User is prompted to re-enter a valid email and password		
Priority:	Medium		
Frequency of Use:	1 - 3 times per lifetime		
Flow of Events:	1. User enters an invalid email and/or password 2. Error message is displayed to the user 3. User is prompted to re-enter a valid email and password		
Alternative Flows:	-		
Exceptions:	-		
Includes:	-		
Special Requirements:	-		
Assumptions:	-		
Notes and Issues:	-		

Use Case ID:	5		
Use Case Name:	Verify Login Credentials		
Created By:	Andrew Ng	Last Updated By	Adi
Date Created:	28th January 2022	Date Last Updated:	22nd March 2022
Actor	System		
Description:	Authenticate account information via email and password		
Precondition:	1. User account must already exist in the database 2. User must have an internet connection 3. User is logged in 4. User is assigned a token		
Postconditions:	1. User can shortlist their shortlisted schools 2. User can retrieve their list of shortlisted schools 3. User can remove their shortlisted schools from the list		
Priority:	Medium		
Frequency of Use:	1 - 3 times per lifetime		
Flow of Events:	1. User attempts to perform an action such as: a. Shortlist b. Commenting 2. System checks if current User's assigned token is valid 3. System allows the User to perform the action		
Alternative Flows:	AF-S2: User account left idle for 24 hours 1. Token assigned to the user is expired 2. System will require the user to log in again		

Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	6		
Use Case Name:	Edit user account password		
Created By:	Andre Lim	Last Updated By	Andrew Ng
Date Created:	28th January 2022	Date Last Updated:	22nd March 2022
Actor	User		
Description:	User requests for a change of password		
Precondition:	1. User must be logged into their account 2. User must have an internet connection		
Postconditions:	1. Password must be updated in database		
Priority:	Medium		
Frequency of Use:	1 - 3 times per year		
Flow of Events:	1. User logs into their account 2. User clicks on their account 3. User selects the “Account” option 4. System prompts user to key in the old password, new password and confirm new password fields 5. System validates the old password 6. System validates the new and confirm password are identical 7. User’s account password is updated to the new password		
Alternative Flows:	AF-S5: User enters an invalid old password 1. System will display an error message 2. User re-enters all three field(s) 3. Return to Step 5 AF-S6: User enters a mismatch of new passwords 1. System will display an error message		

	2. User re-enters the respective field(s) 3. Return to Step 5
Exceptions:	-
Includes:	1. Verify Login Credentials
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	7		
Use Case Name:	Shortlist School		
Created By:	Lim Kai Sheng	Last Updated By	Ling Yin
Date Created:	28th January 2022	Date Last Updated:	22nd March 2022
Actor	User		
Description:	User can add schools they are interested in to their list of shortlisted schools		
Precondition:	1. User account must already exist in the database 2. User must be logged into their account 3. User must have an internet connection		
Postconditions:	1. User's preferred schools will be saved in the database		
Priority:	High		
Frequency of Use:	0 - 10 times per day		

Flow of Events:	1. User searches for schools based on preference 2. User clicks on the “Plus” button to bookmark their shortlisted schools. 3. System will save the user’s shortlisted schools into the database.
Alternative Flows:	-
Exceptions:	-
Includes:	1. Verify login credentials
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	8		
Use Case Name:	Remove Shortlisted School		
Created By:	Andre Lim	Last Updated By	Andre Lim
Date Created:	28th January 2022	Date Last Updated:	22nd March 2022
Actor	User		
Description:	Shortlisted schools can be removed by the user		

Precondition:	1. User account must already exist in the database 2. User must be logged into their account 3. User must have an internet connection
Postconditions:	1. User's selected school will be removed from their shortlisted list
Priority:	High
Frequency of Use:	0 - 5 times per day
Flow of Events:	1. User clicks on the "My Shortlist" button to view their shortlisted schools 2. User selects the shortlisted schools that they wants to remove 3. User clicks on the "Delete" icon 4. System will drop the user's shortlisted schools from the database
Alternative Flows:	-
Exceptions:	-
Includes:	1. Verify login credentials
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	9		
Use Case Name:	Retrieve Shortlisted School		
Created By:	Andrew Ng	Last Updated By	Lim Kai Sheng
Date Created:	28th January 2022	Date Last Updated:	10th February 2022
Actor	User		
Description:	Displays list of shortlisted schools		
Precondition:	1. User account must already exist in the database 2. User must be logged into their account 3. User must have an internet connection		
Postconditions:	1. A list of User's shortlisted schools will be displayed		
Priority:	High		
Frequency of Use:	0 - 5 times a day		
Flow of Events:	1. User selects "My Shortlist" 2. List of shortlisted schools for the user is fetched from the database 3. System displays the retrieved shortlisted school(s) to the User		
Alternative Flows:	-		
Exceptions:	-		
Includes:	-		
Special Requirements:	-		

Assumptions:	-
Notes and Issues:	-

Use Case ID:	10		
Use Case Name:	Create Comment		
Created By:	Lim Kai Sheng	Last Updated By	Ling Yin
Date Created:	28th January 2022	Date Last Updated:	11th March 2022
Actor	User		
Description:	User is able to post their comments about schools		
Precondition:	1. User account must already exist in the database 2. User must be logged into their account 3. User must have an internet connection		
Postconditions:	1. Comments are displayed in the comment section of the respective school		
Priority:	Medium		
Frequency of Use:	0 - 5 times per year		
Flow of Events:	1. User types in their comments in comment box 2. User clicks “post” button 3. Comment is now publicly displayed in the comment section of the school		
Alternative Flows:	-		
Exceptions:	-		
Includes:	1. Verify Login Credentials		
Special Requirements:	-		

Assumptions:	-
Notes and Issues:	-

Use Case ID:	11		
Use Case Name:	Edit Comment		
Created By:	Andre Lim	Last Updated By	Ling Yin
Date Created:	28th January 2022	Date Last Updated:	30th January 2022
Actor	User		
Description:	User can edit their posted comment		
Precondition:	1. User account must already exist in the database 2. User must be logged into their account 3. User must have already posted a comment 4. User must have an internet connection		
Postconditions:	1. User's comment is edited and displayed in the comment section of the respective school		
Priority:	Medium		
Frequency of Use:	0 - 4 times per year		
Flow of Events:	1. User navigates to his comments 2. User clicks on "edit" button to start editing his post 3. User modifies his comment 4. User clicks on "confirm" button to modify his original comments		
Alternative Flows:	AF-S2: User decides not to edit his post 1. User clicks on "cancel button" 2. Comment is not modified		
Exceptions:	-		

Includes:	1. Verify Login Credentials
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	12		
Use Case Name:	Delete Comment		
Created By:	Andrew Ng	Last Updated By	Lim Kai Sheng
Date Created:	28th January 2022	Date Last Updated:	16th March 2022
Actor	User		
Description:	User can delete their posted comment		
Precondition:	1. User account must already exist in the database 2. User must be logged into their account 3. User must have already posted a comment 4. User must have an internet connection		
Postconditions:	1. User's comment is removed from the comment section of the respective school		
Priority:	Medium		
Frequency of Use:	0 - 4 times per year		
Flow of Events:	1. User navigates to his comments 2. User clicks on "delete" button 3. User clicks "confirm" button to delete their comment 4. Comment is removed		
Alternative Flows:	AF-S3: User decides not to delete their comment 1. User clicks on the "cancel" button 2. Comment is not deleted		
Exceptions:	-		
Includes:	1. Verify Login Credentials		

Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Use Case ID:	13		
Use Case Name:	Like Comment		
Created By:	Lim Kai Sheng	Last Updated By	Andre Lim
Date Created:	28th January 2022	Date Last Updated:	2nd March 2022
Actor	User		
Description:	User can like other posted comment		
Precondition:	1. User account must already exist in the database 2. User must be logged into their account 3. User must have an internet connection		
Postconditions:	1. “Liked” should be reflected on the intended comment		
Priority:	Medium		
Frequency of Use:	0 - 4 times per year		
Flow of Events:	1. User clicks on the “like” button 2. The system will update the comment’s “Liked” counts		
Alternative Flows:			
Exceptions:	-		
Includes:	-		
Special Requirements:	-		
Assumptions:	User cannot un-like the comments they has made.		
Notes and Issues:	-		

Use Case ID:	14		
Use Case Name:	Search for Schools [based on filters]		
Created By:	Andre Lim	Last Updated By	Andre Lim
Date Created:	28th January 2022	Date Last Updated:	2nd March 2022
Actor	User		
Description:	Filter schools based on user's preference		
Precondition:	1. User must have an internet connection		
Postconditions:	1. A list of schools will be displayed under the search bar based on the user's search query and filters		
Priority:	High		
Frequency of Use:	0-40 times per day		
Flow of Events:	1. User enters his query in the Search bar 2. System will provide suggestions based on User's query 3. User clicks on the "Search" button without adding any filters 4. System displays list of schools based on User's query		
Alternative Flows:	1. User selects the pre-defined filter from the drop-down 2. System will filter the schools based in user's input criteria 3. System will display the filtered schools		
Exceptions:	-		
Includes:	-		
Special Requirements:	-		

Assumptions:	-
Notes and Issues:	-

Use Case ID:	15		
Use Case Name:	Select School		
Created By:	Andrew Ng	Last Updated By	Adi
Date Created:	28th January 2022	Date Last Updated:	22nd March 2022
Actor	User		
Description:	User selects a school and its information is displayed		
Precondition:	1. User must have an internet connection		
Postconditions:	1. Selected school's information is displayed		
Priority:	High		
Frequency of Use:	0 - 20 times per day		
Flow of Events:	1. System displays a list of all schools 2. User selects a school 3. System displays the selected school's details		
Alternative Flows:	AF-S1: Filters have been applied 1. System displays a list of schools meeting the criteria of the filters 2. Return to Step 2		
Exceptions:	-		
Includes:	1. Retrieve the list of school's information		
Special Requirements:			
Assumptions:	-		

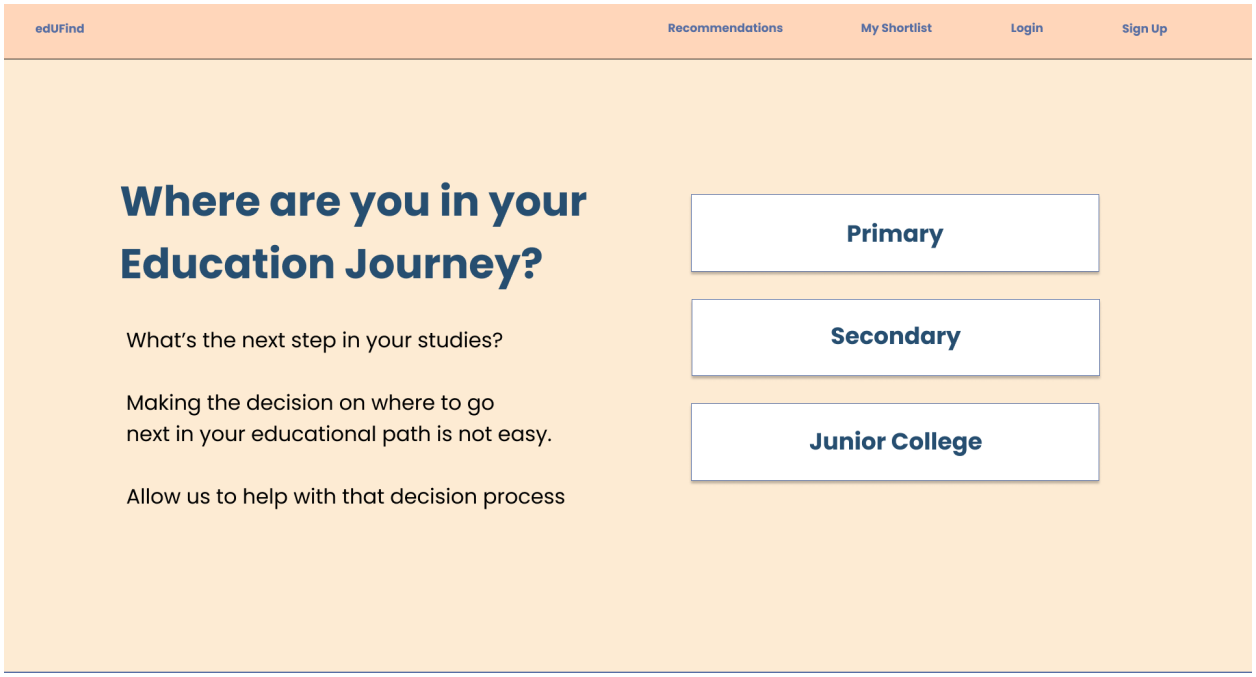
Notes and Issues:	-

Use Case ID:	16		
Use Case Name:	Enter Preferences		
Created By:	Andrew Ng	Last Updated By	Lim Kai Sheng
Date Created:	10th February 2022	Date Last Updated:	20th March 2022
Actor	User		
Description:	System recommends a list of schools to the user based on their preferences		
Precondition:	1. User account must exist in the database 2. User must be logged into their account 3. User must have an internet connection		
Postconditions:	1. Display a list of schools based on user's preferences		
Priority:	Medium		
Frequency of Use:	1-3 time per account creation		
Flow of Events:	1. User selects Gender, Region, Level of Education, Mother Tongue and CCA Category		

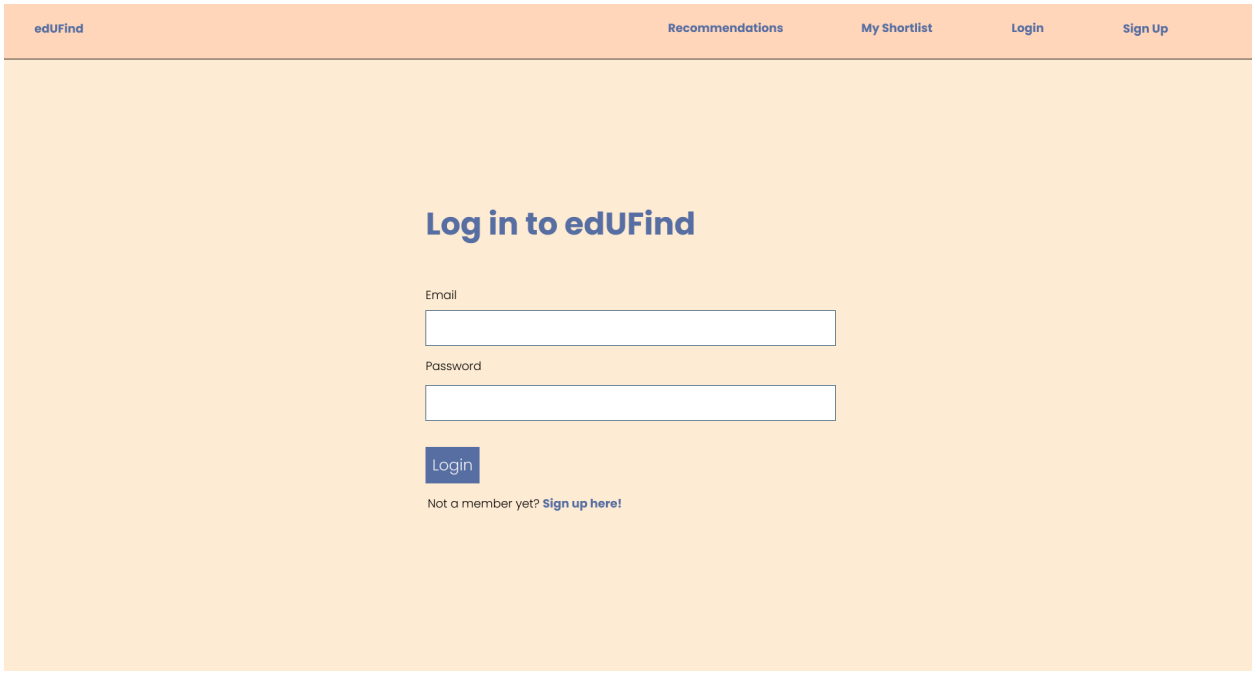
	2. System displays a list of schools as recommendations
Alternative Flows:	-
Exceptions:	-
Includes:	-
Special Requirements:	-
Assumptions:	-
Notes and Issues:	-

Initial UI Mockup

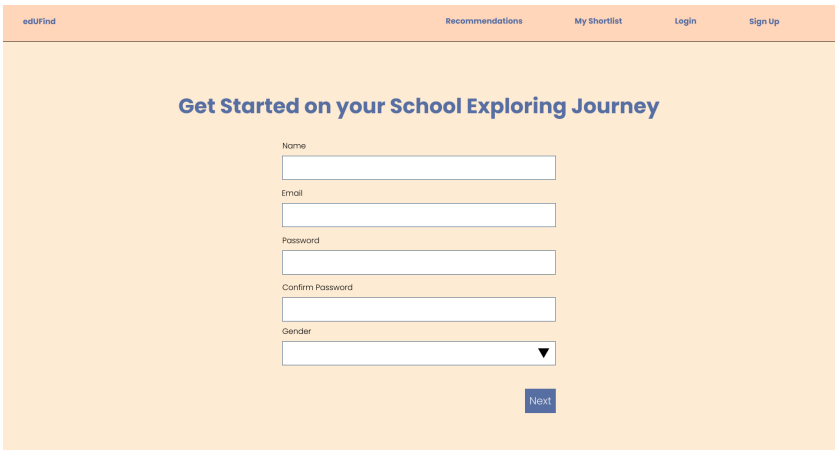
Home Page

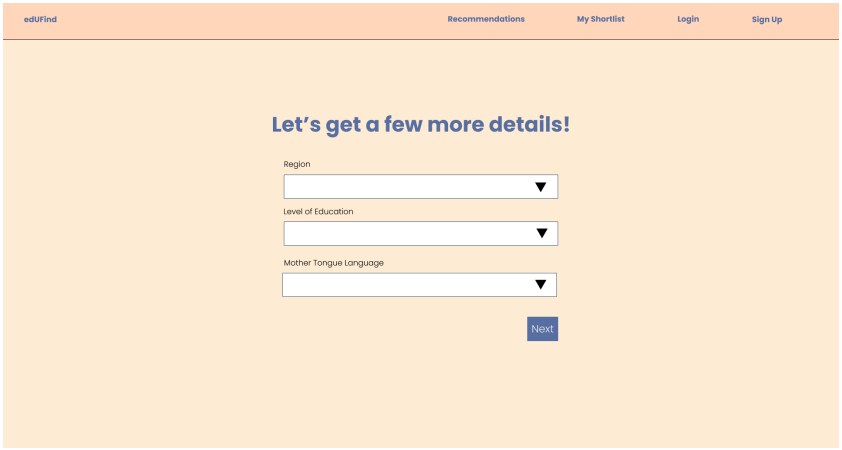


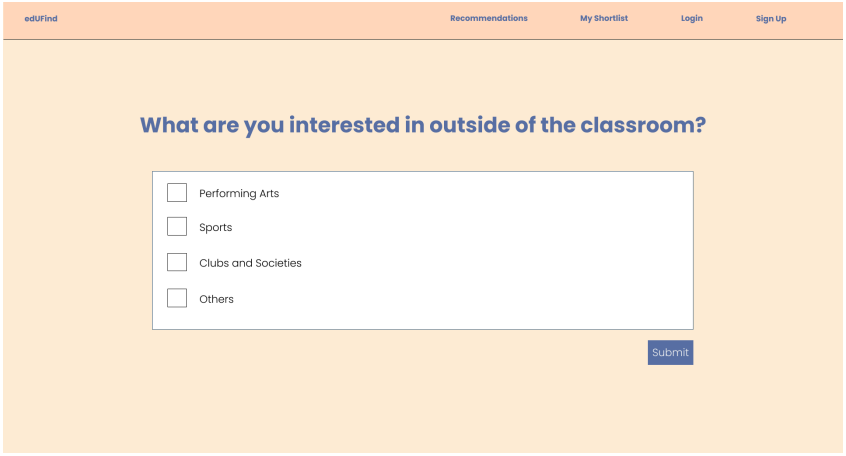
Login Page



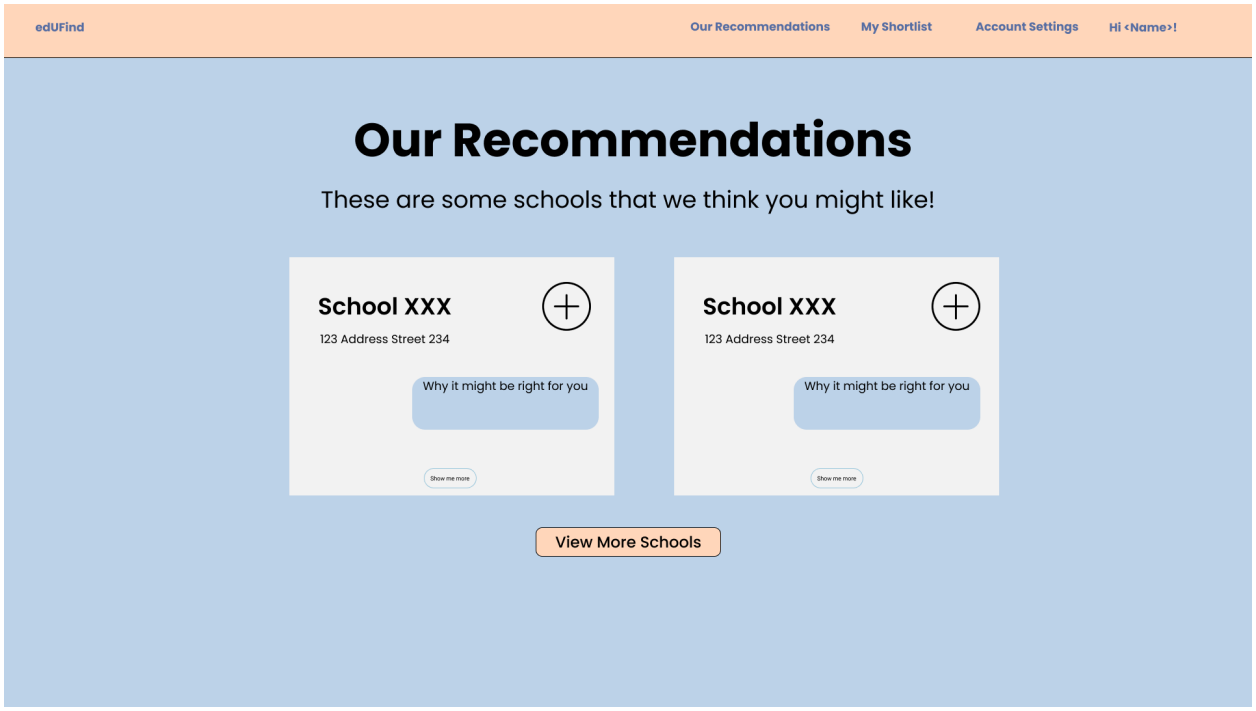
Sign Up Procedures



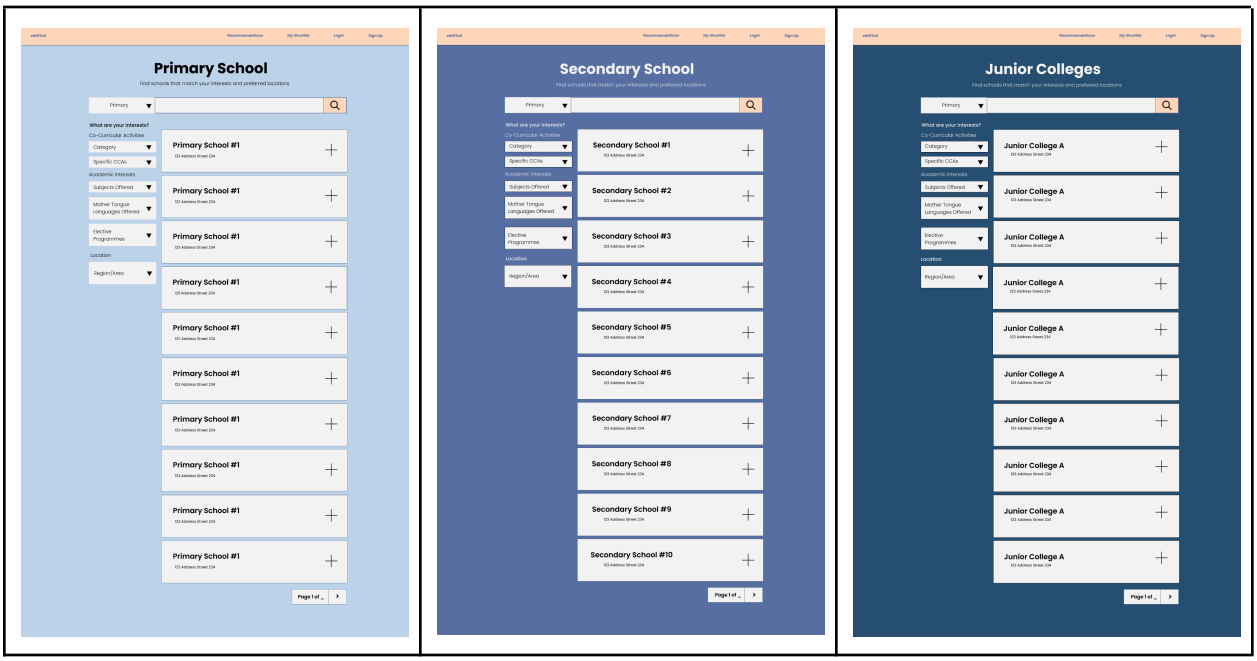




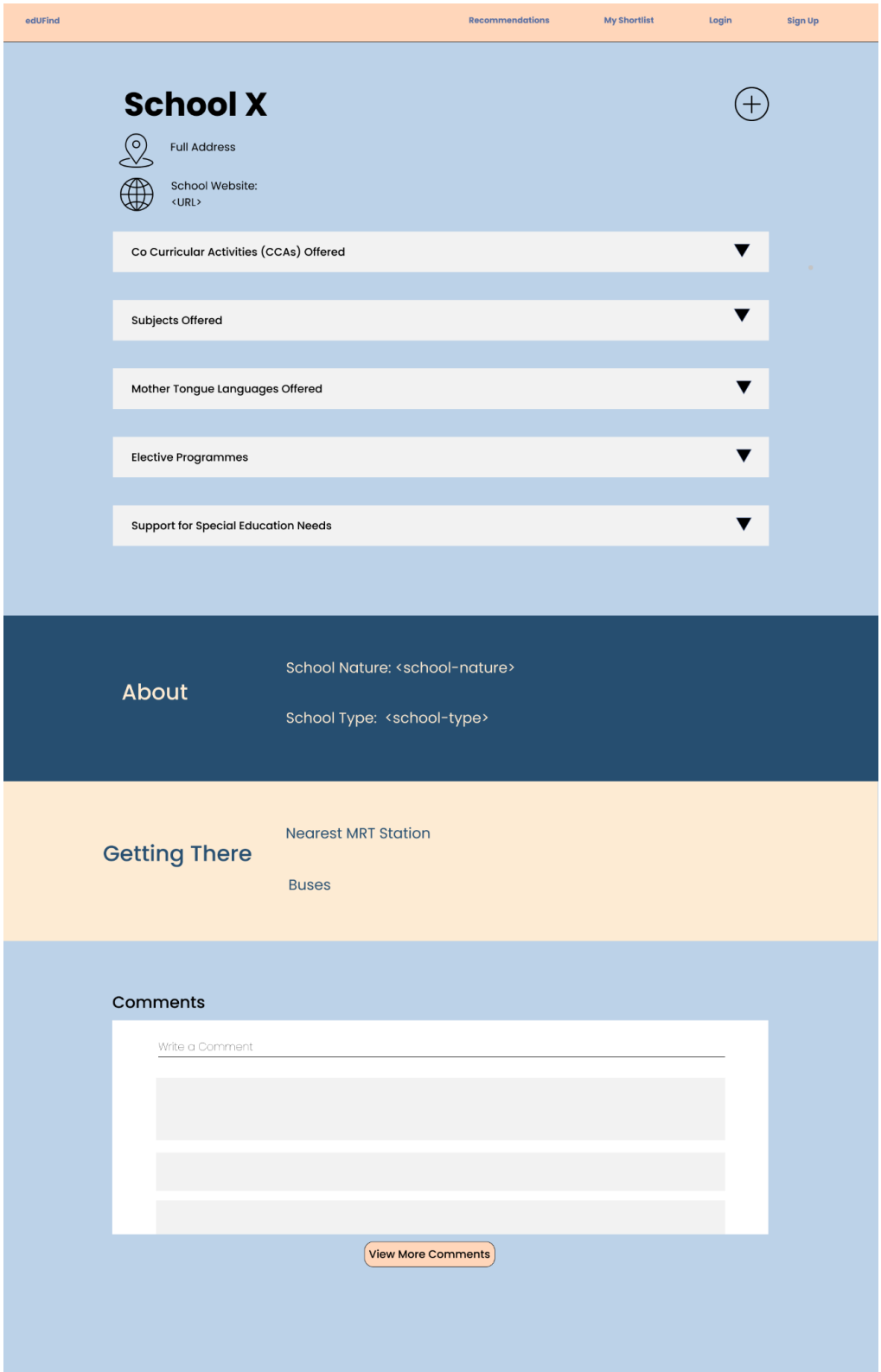
Recommendations



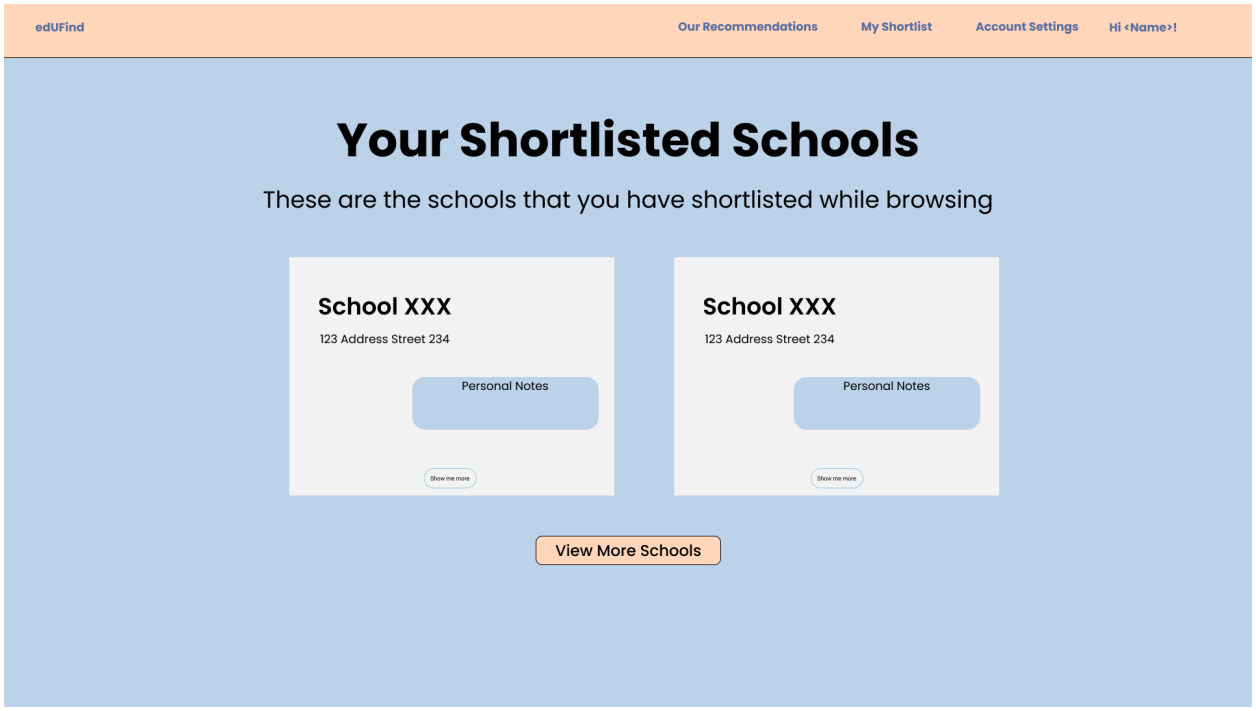
\Primary Schools, Secondary Schools and Junior Colleges



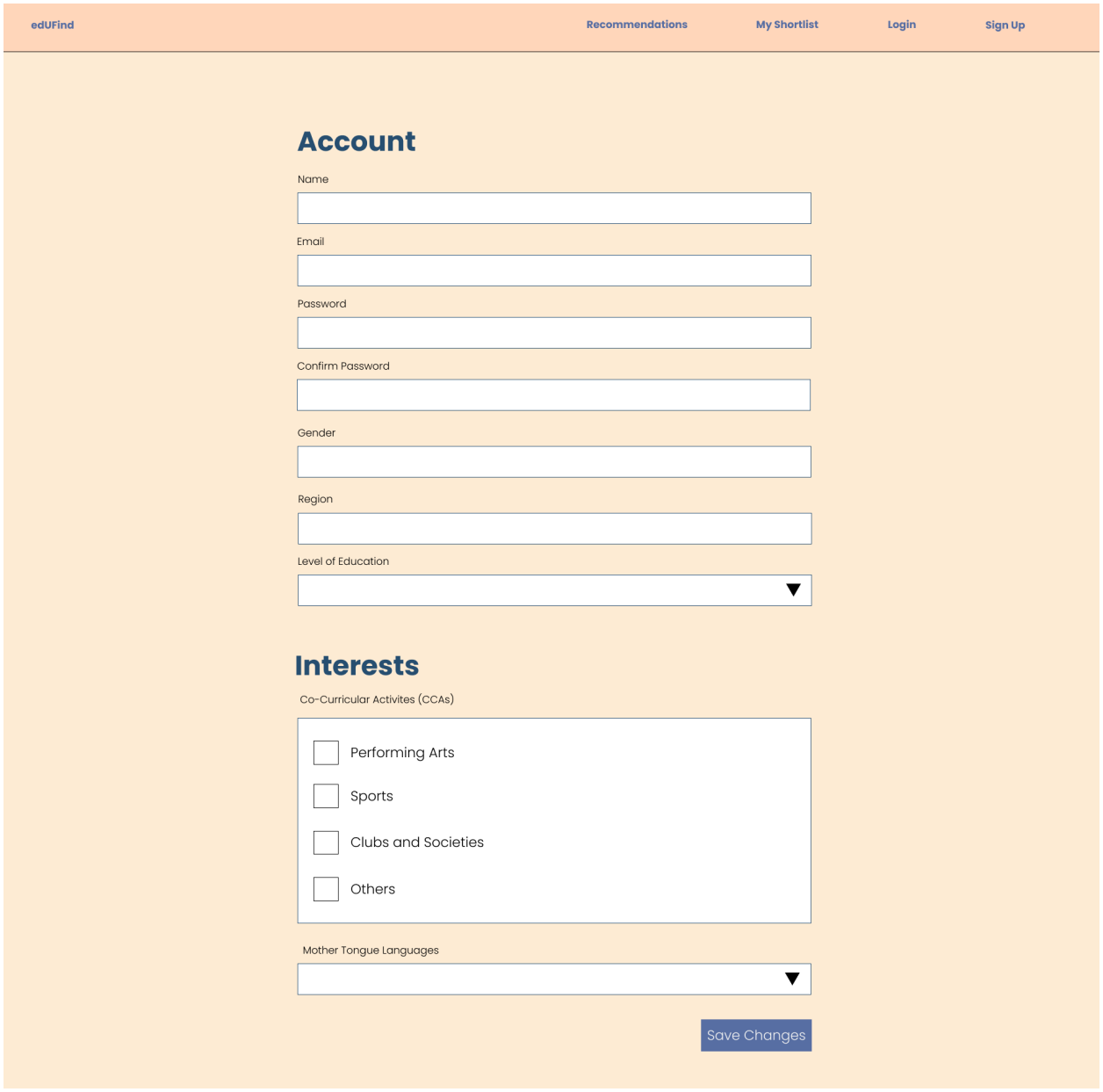
School details



Shortlisted Schools

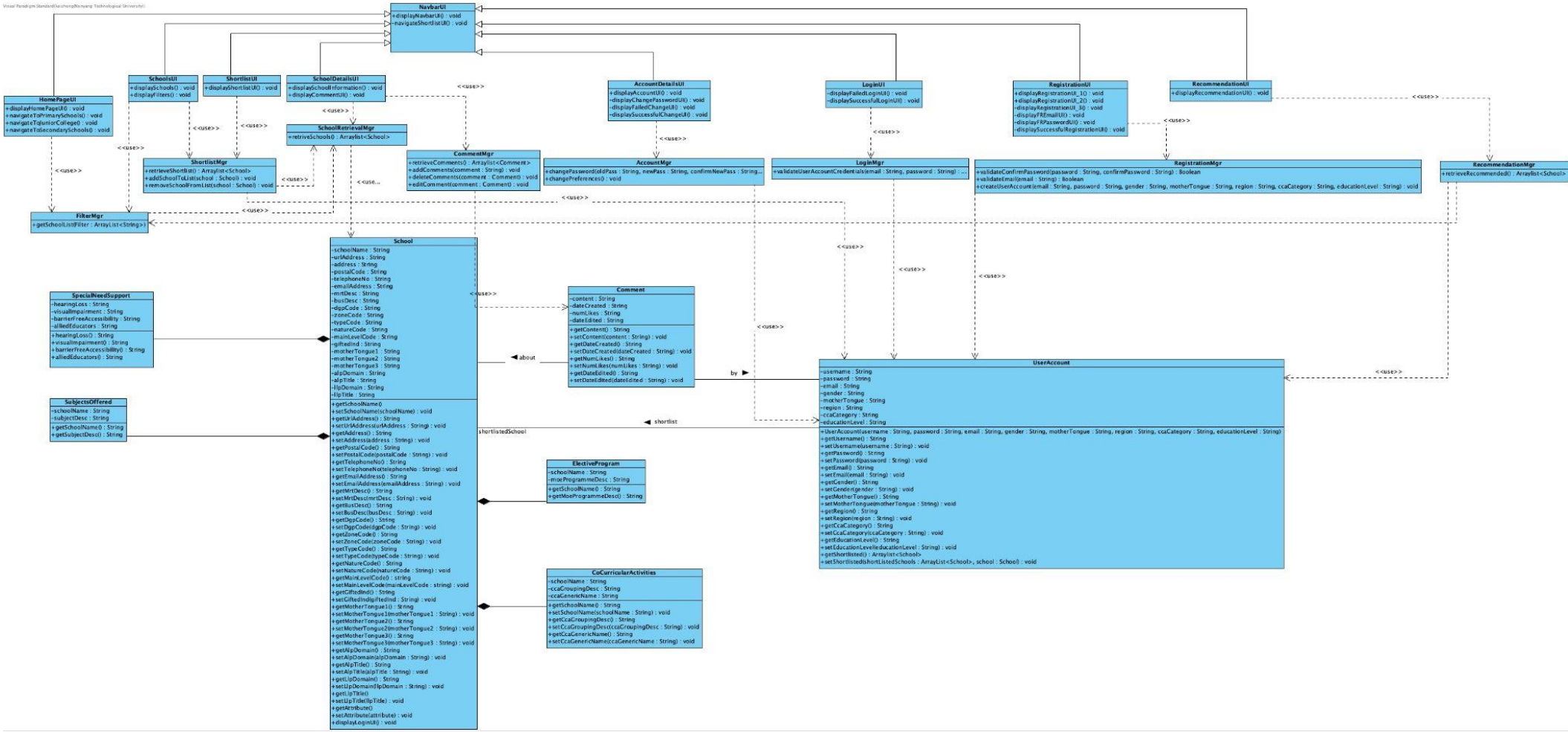


Account Settings



Class Diagram

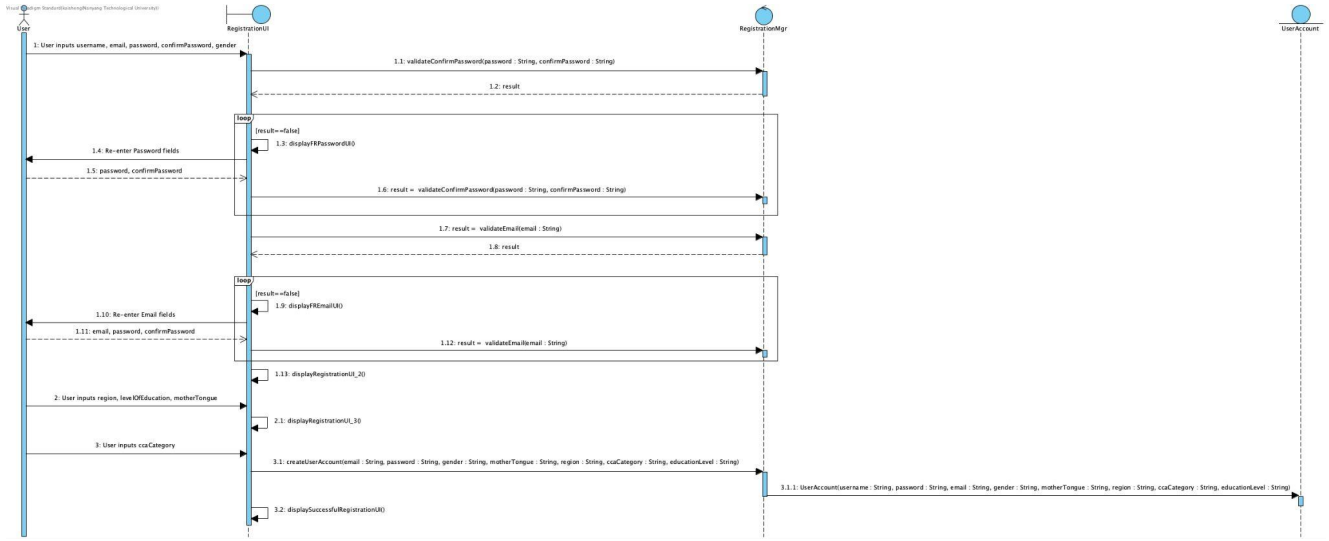
Refer to Class Diagram for the JPG File



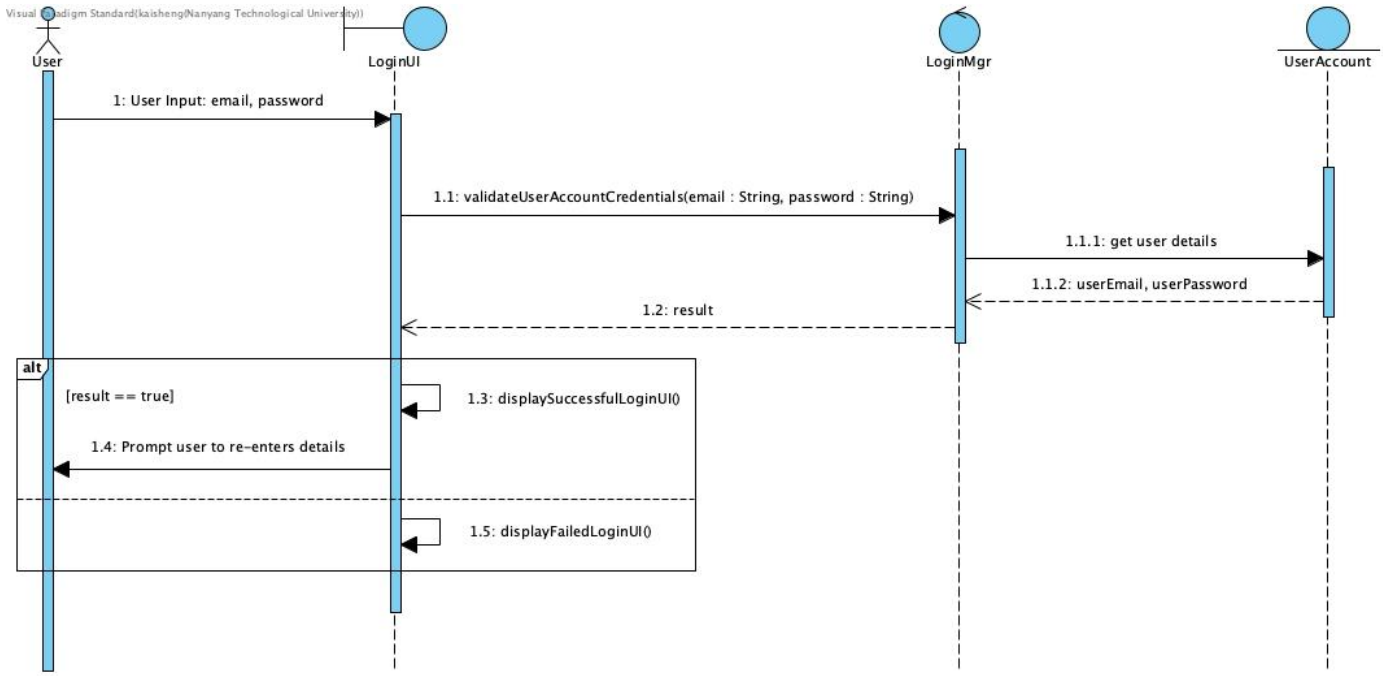
Sequence Diagram

Refer to Sequence Diagram folder for the JPG Files

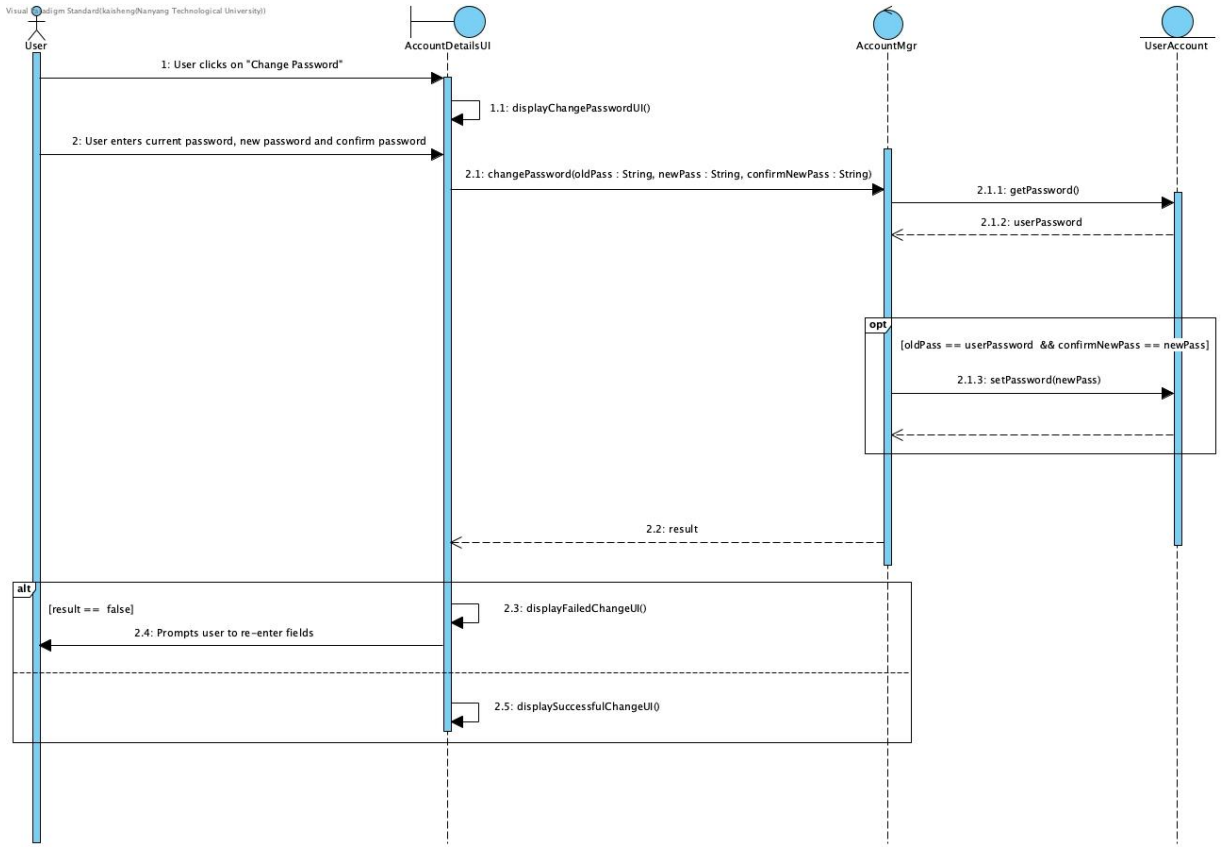
Registering User Account



Login

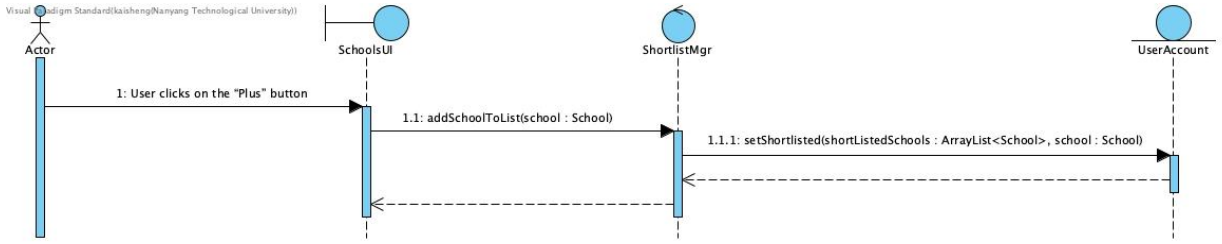


Request Password Change

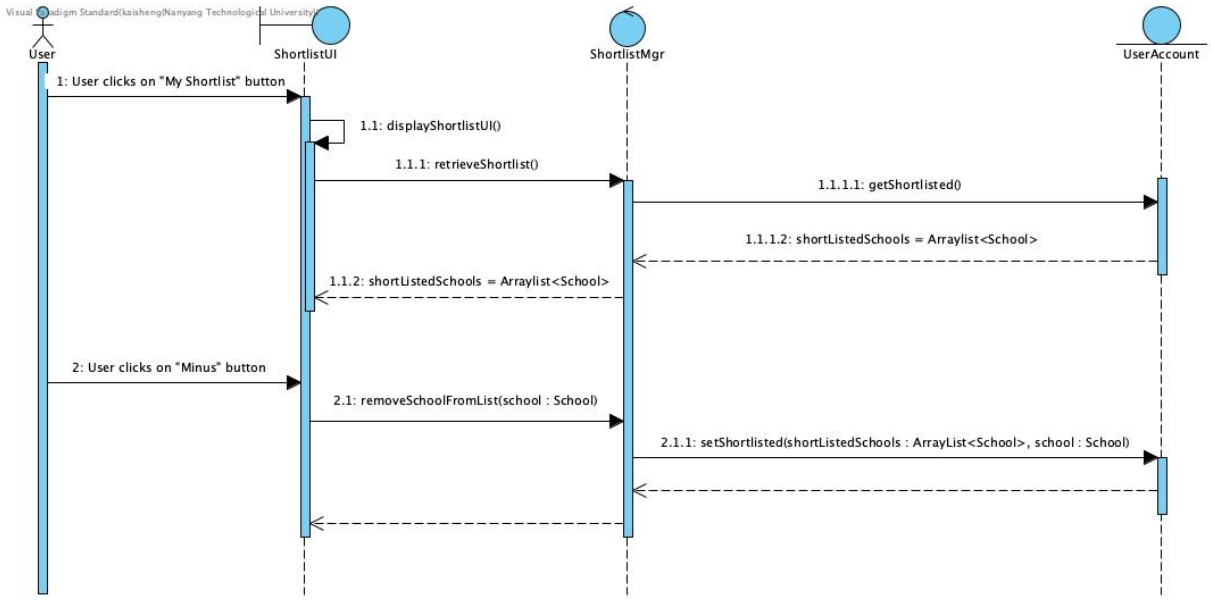


Shortlist

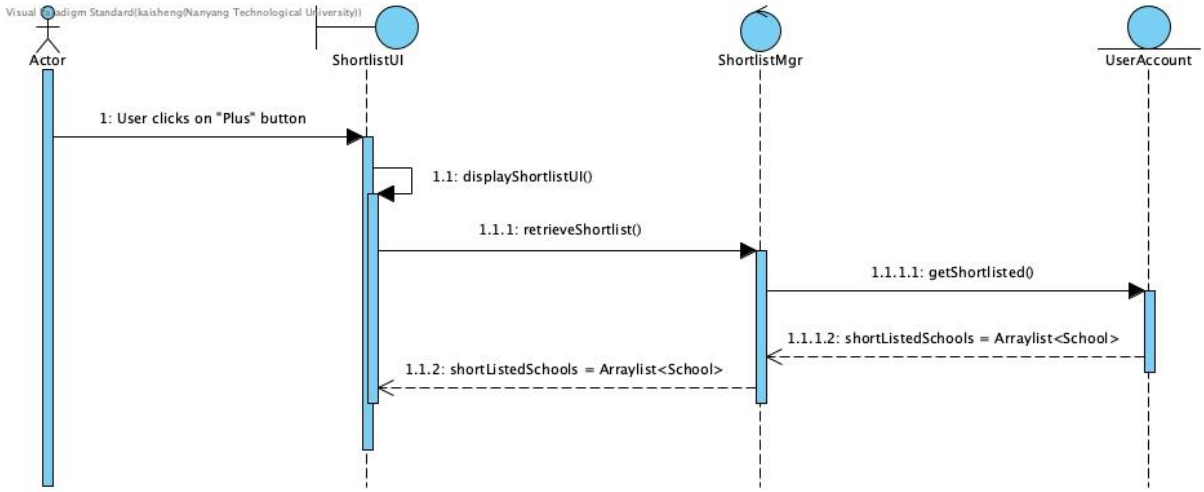
Schools



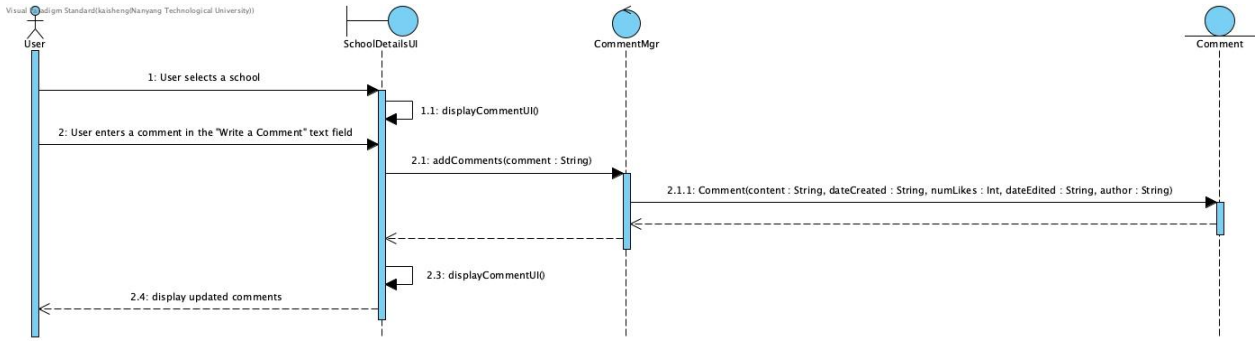
Remove Shortlisted Schools



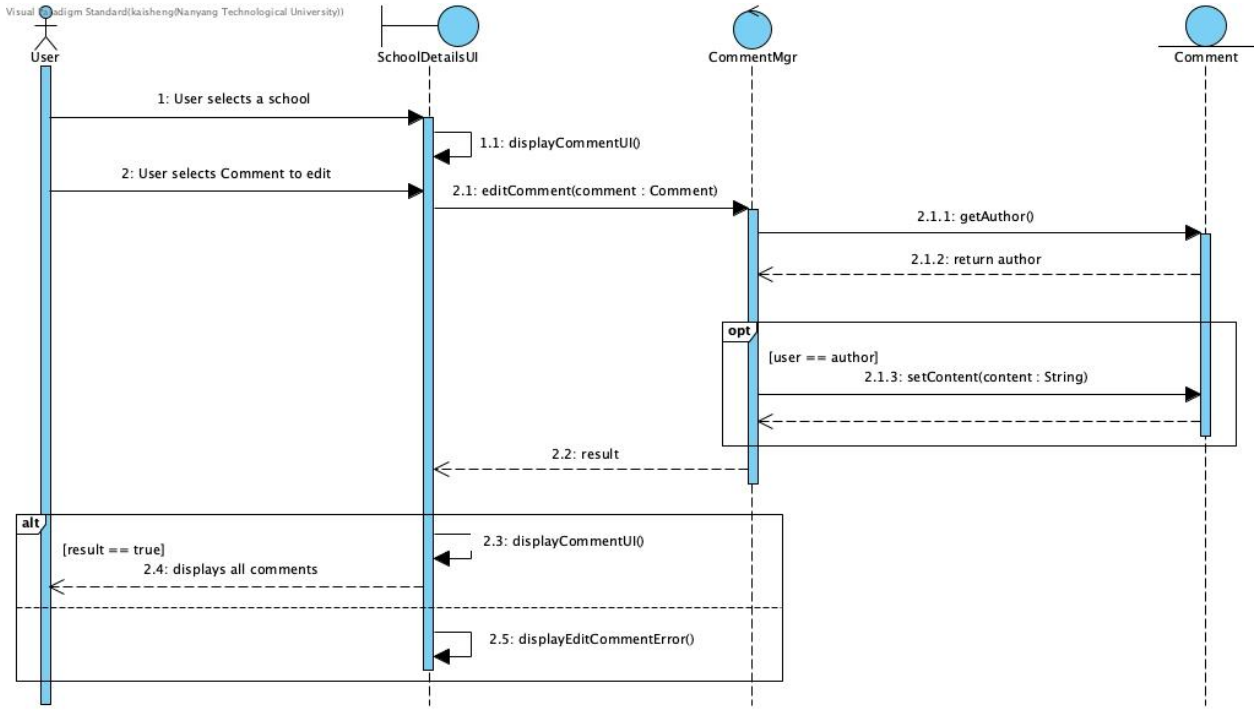
Retrieve Shortlisted Schools



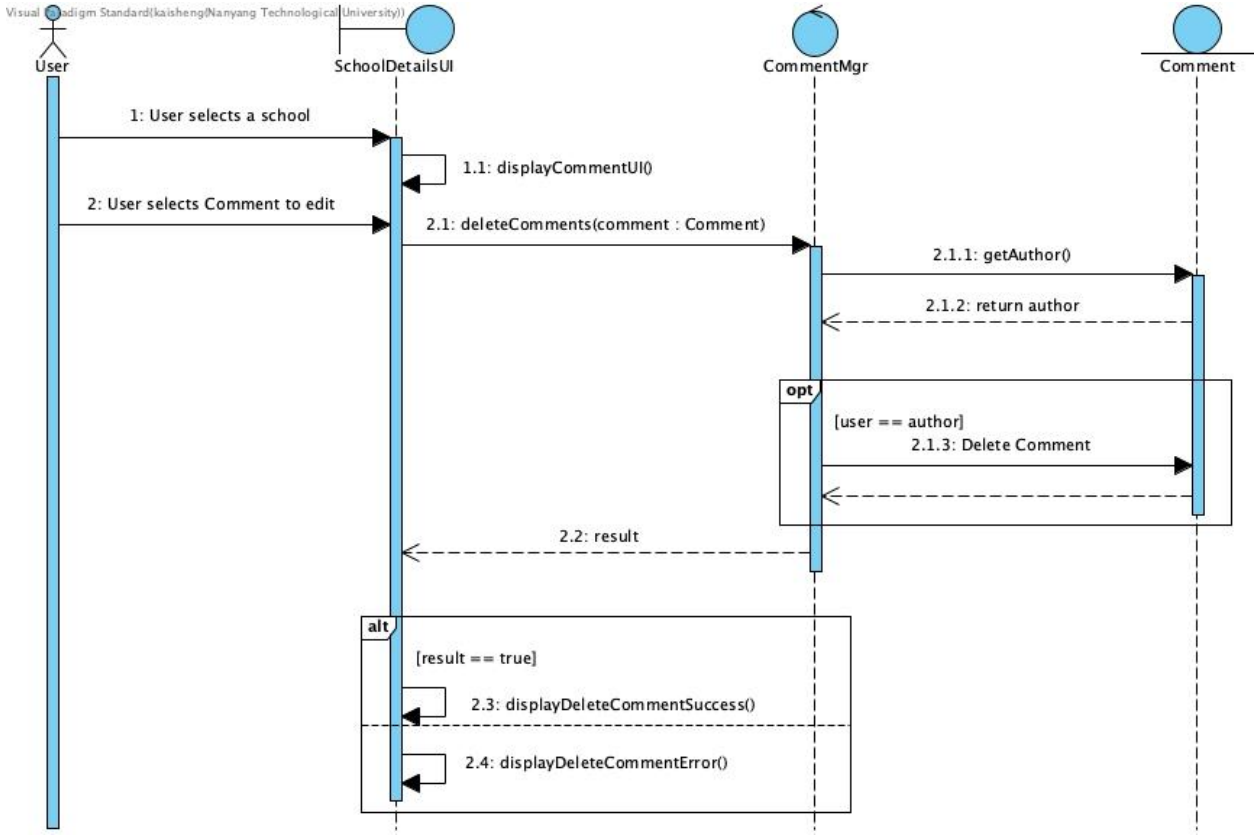
Create Comment



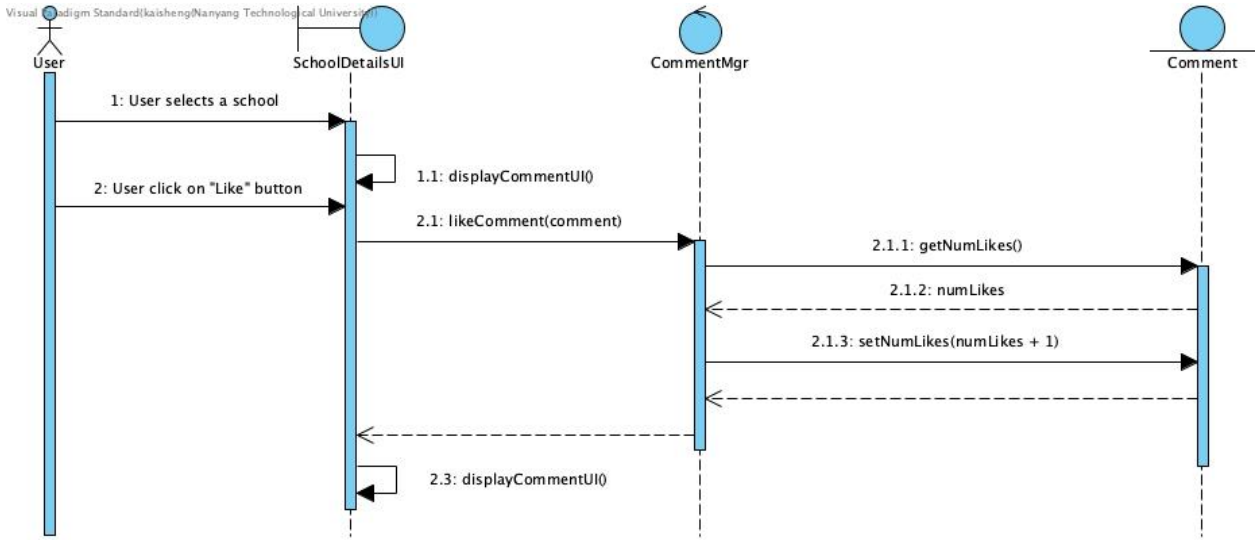
Edit Comment



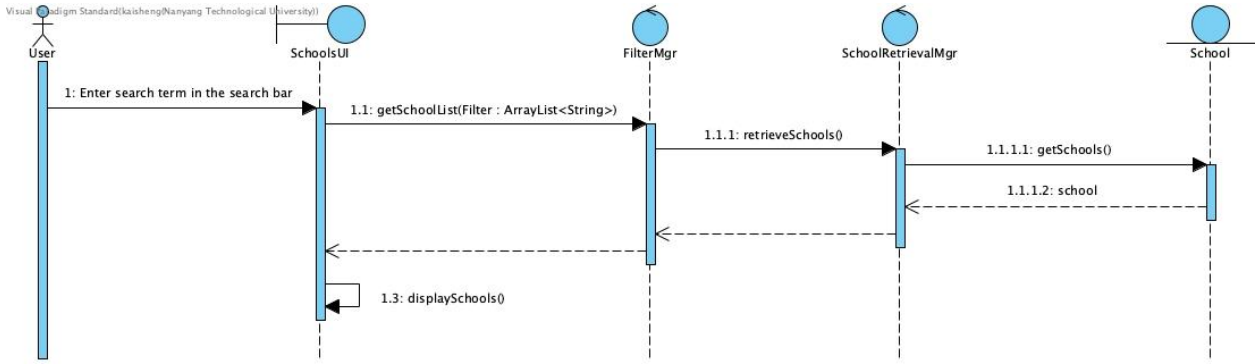
Display Comment



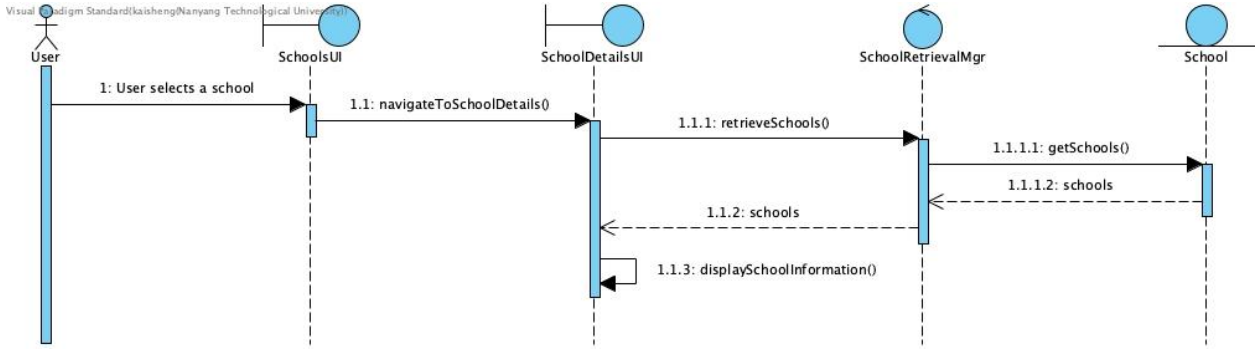
Like Comment



Search School

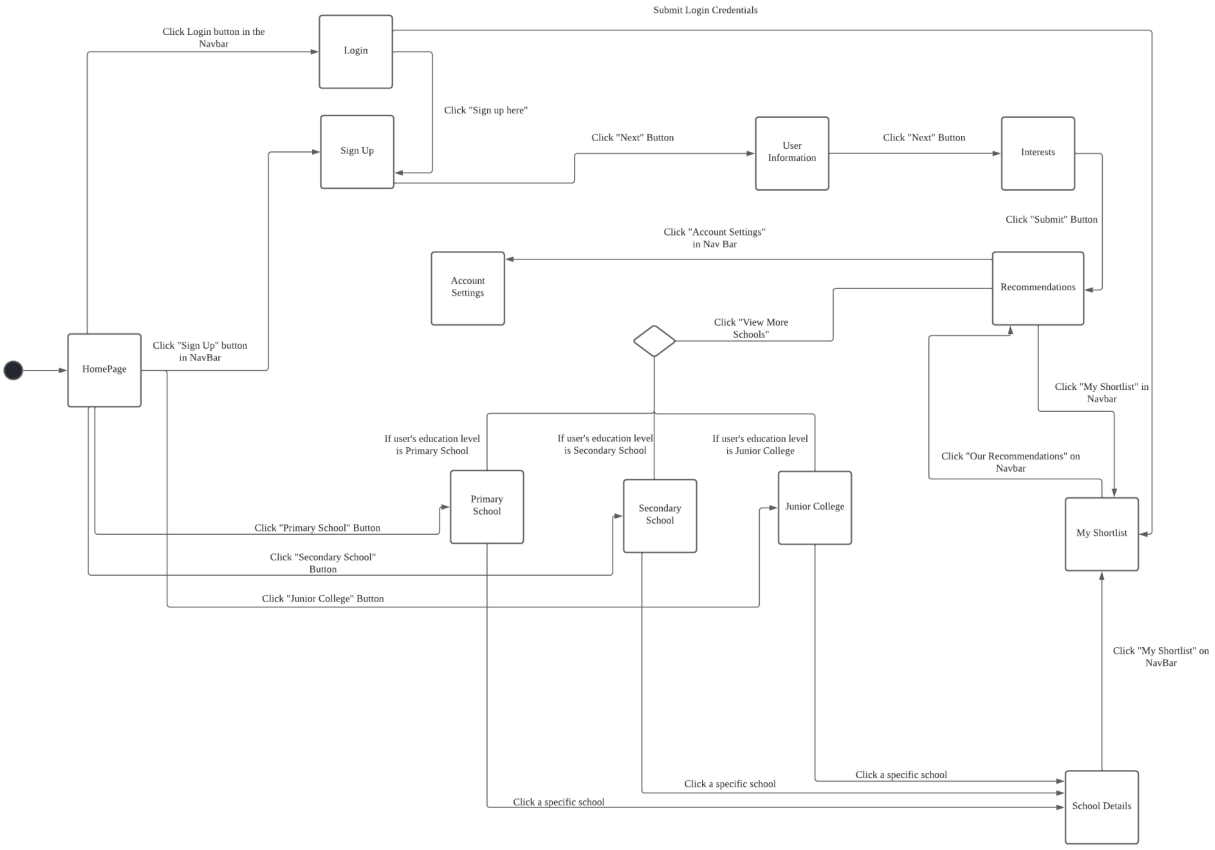


Select School



Dialog Map

Refer to Dialog Map.png for clearer image



BlackBox Testing

Login

Generic Case			
Test ID	**Test Input**	**Expected Output**	**Actual Output**
1	Login with valid account username and password	The system returns back to homepage with navbar indicating successful login	The system returns back to homepage with navbar indicating successful login
2	Login with incomplete fields (email, password)	The system prompts the user to fill up the required fields for login	The system prompts the user to fill up the required fields for login
3	Login with wrong email regex (regular expression)	The system prompts the user to re-enter the email. It will display the corresponding error message	The system prompts the user to re-enter the email. It will display the corresponding error message
4	Login with email length more than 300 characters	The system prompts the user to re-enter the email. It will display the corresponding error message	The system prompts the user to re-enter the email. It will display the corresponding error message
5	Login with an unregistered email address	The system prompts the user to re-enter the email. It will display the corresponding error message	The system prompts the user to re-enter the email. It will display the corresponding error message
6	Login with password less than 6 characters, or greater than 50 characters	The system prompts the user to re-enter the password. It will display the corresponding error message	The system prompts the user to re-enter the password. It will display the corresponding error message

7	Login with incorrect password	The system prompts the user to re-enter the password. It will display the corresponding error message	The system prompts the user to re-enter the password. It will display the corresponding error message

In Particular				
Test Case	**Email**	**Password**	**Expected Output**	**Actual Output**
1	test@gmail.com	testpass	Successful login	Successful login
2	Empty(“”)	testpass	Please fill in all required fields	Please fill in all required fields
2	test@gmail.com	Empty(“”)	Please fill in all required fields	Please fill in all required fields
3	test	testpass	Email must be in name@email.com format	Email must be in name@email.com format
4	test@gmail.com[>300]	testpass	Email is too long	Email is too long
5	unregistered@gmail.com	testpass	User account does not exist	User account does not exist
6	test@gmail.com	pass[<6]	Invalid password length	Invalid password length
6	test@gmail.com	pass[>50]	Invalid password length	Invalid password length
7	test@gmail.com	wrongpass	Entered passwords do not match	Entered passwords do not match

Registration

Generic Case			
Test ID	**Test Input**	**Expected Output**	**Actual Output**
1	Register with valid account username and password	The system displays the main menu for user to continue the operation	The system displays the main menu for user to continue the operation
2	Register with incomplete fields (username, email, password)	The system prompts the user to fill up the required fields for registration	The system prompts the user to fill up the required fields for registration
3	Register with wrong email regex (regular expression)	The system prompts the user to re-enter the email. It will display the corresponding error message	The system prompts the user to re-enter the email. It will display the corresponding error message
4	Register with email length more than 300 characters	The system prompts the user to re-enter the email. It will display the corresponding error message	The system prompts the user to re-enter the email. It will display the corresponding error message
5	Register with an already registered email address	The system prompts the user to re-enter the email. It will display the corresponding error message	The system prompts the user to re-enter the email. It will display the corresponding error message
6	Register with password less than 6 characters, or greater than 50 characters	The system prompts the user to re-enter the password. It will display the corresponding error message	The system prompts the user to re-enter the password. It will display the corresponding error message

7	Register with password mismatch	The system prompts the user to re-enter the password. It will display the corresponding error message	The system prompts the user to re-enter the password. It will display the corresponding error message

In Particular						
Test Case	**Name**	**Email Address**	**Password**	**Confirm Password**	**Expected Output**	**Actual Output**
1	testuser	test@gmail.com	testpass	testpass	Successful login	Successful login
2	Empty(“”)	test@gmail.com	testpass	testpass	Please fill in all required fields	Please fill in all required fields
2	testuser	Empty(“”)	testpass	testpass	Please fill in all required fields	Please fill in all required fields
2	testuser	test@gmail.com	Empty(“”)	testpass	Please fill in all required fields	Please fill in all required fields
2	testuser	test@gmail.com	wrongpass	Empty(“”)	Please fill in all required fields	Please fill in all required fields
3	testuse	test	testpass	testpass	Email must be in name@email.com format	Email must be in name@email.com format

4	testuse	test@gmail.com[>300]	testpass	testpass	Email is too long	Email is too long
5	testuser	exist@gmail.com	testpass	testpass	Email has been used	Email has been used
6	testuser	test@gmail.com	pass[<6]	pass[<6]	Invalid password length	Invalid password length
6	testuser	test@gmail.com	pass[>50]	pass[>50]	Invalid password length	Invalid password length
7	testuser	test@gmail.com	wrongpass	testpass	Entered passwords do not match	Entered passwords do not match
7	testuser	test@gmail.com	testpass	wrongpass	Entered passwords do not match	Entered passwords do not match

Edit Shortlist

Generic Case			
Test ID	**Test Input**	**Expected Output**	**Actual Output**
1	Shortlist a school	The system saves the school into shortlist	The system saves the school into shortlist
2	Removal of shortlisted school	The system removes the school into shortlist	The system removes the school into shortlist

View Shortlist

Generic Case			
Test ID	**Test Input**	**Expected Output**	**Actual Output**
1	Display Shortlisted School [Empty]	The system displays shortlist page with empty result	The system displays shortlist page with empty result
2	Displau Shortlisted School [Filled]	The system displays shortlist page with intended results	The system displays shortlist page with intended results

WhiteBox Testing

Login

Method: userLogin()

Parameters: email, password

Return: Json response

Purpose: Authenticate, and assign users with a valid JWT token for Log-in required functions.

Basic Paths:

a) 0, 1, 3, 5, 7, 9, 11, 13 (baseline)

Test Cases:

a) False outcome of first decision point.

- Input: email = null, or password = null
- User left the compulsory field (email, password) empty.

b) False outcome of second decision point

- Input: email = “email”
- User entered the email without the right regular expression. Specifically, no ‘dot’, ‘at’, and characters.

c) False outcome of third decision point

- Input: email = “email”*300 + “@gmail.com”
- User entered email with more than 300 characters.

d) False outcome of fourth decision point

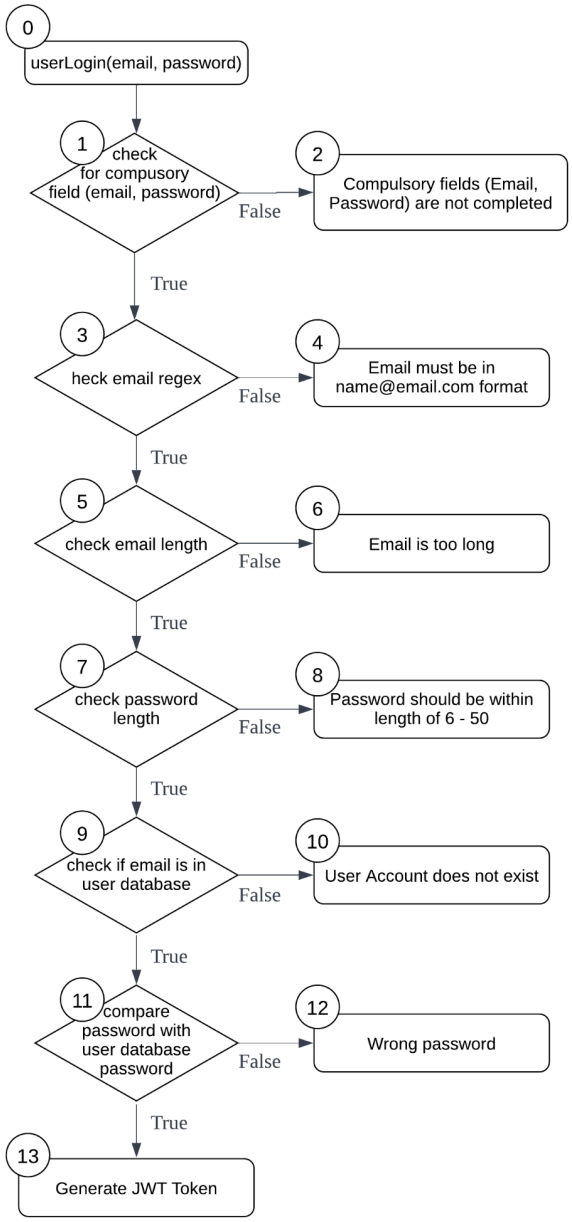
- Input: password = pass
- User entered password that is less than 6 characters or more than 50 characters.

e) False outcome of fifth decision point

- Input: email = exist@gmail.com
- User entered an email that is not registered in the user database.

f) False outcome of sixth decision point

- Input: password = wrongpass
- User entered the wrong password.



Execution Paths: a) 0, 1, 2 b) 0, 1, 3, 4 c) 0, 1, 3, 5, 6 d) 0, 1, 3, 5, 7, 8 e) 0, 1, 3, 5, 7, 9, 10 f) 0, 1, 3, 5, 7, 9, 11, 12 --- g) 0, 1, 3, 5, 7, 9, 11, 13 Cyclomatic Complexity: [Conditions + 1] = [6 + 1] = 7 [Edges - Nodes + 2] = [13 - 8 + 2] = 7	

Register

Method: registerUser()

Parameters: username, email, password, gender, region, motherTongueLangauge, educationLevel, ccaInterest

Return: Json response

Purpose: Register, and assign user with a valid JWT token for current Log-in session.

Basic Paths:

a) 0, 1, 3, 5, 7, 9, 10, 12 (baseline)

Test Cases:

a) False outcome of first decision point.

- *Input: username = null, email = null, password = null.*
- The user left the compulsory field (username, email, password) empty.

b) False outcome of second decision point

- *Input: email = "email"*
- The user entered the email without the right regular expression. Specifically, no 'dot', 'at', and characters.

c) False outcome of third decision point

- *Input: email = "email"*300 + "@gmail.com"*
- The user entered an email with more than 300 characters.

d) False outcome of fourth decision point

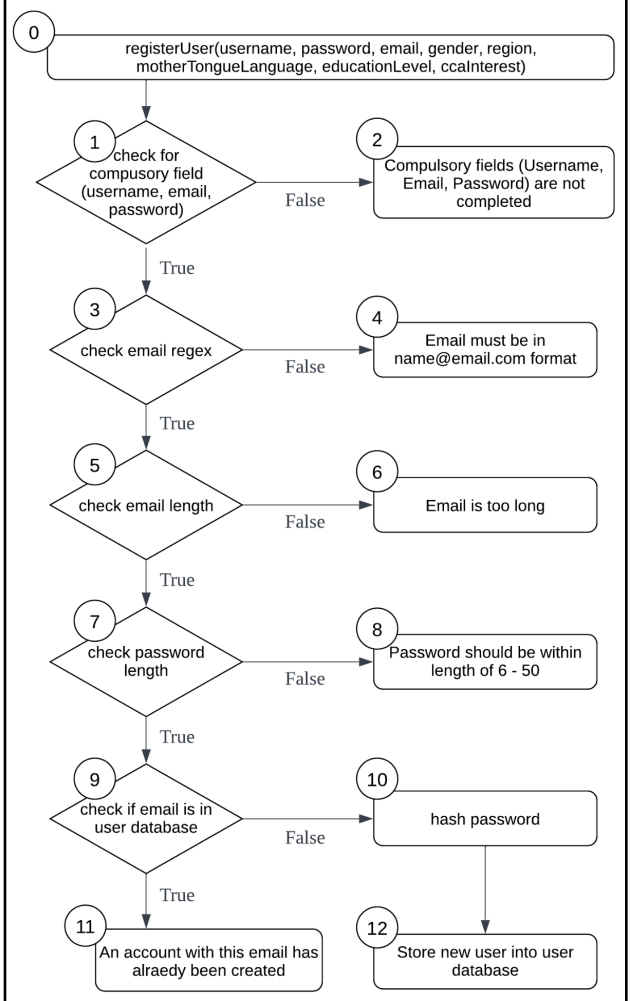
- *Input: password = pass*
- The user entered a password that is less than 6 characters or more than 50 characters.

e) False outcome of fifth decision point

- *Input: email = exist@gmail.com*
- The user entered an email that is already registered in the user database. The email has already been registered.

Execution Paths:

a) 0, 1, 2



b) 0, 1, 3, 4 c) 0, 1, 3, 5, 6 d) 0, 1, 3, 5, 7, 8 e) 0, 1, 3, 5, 7, 9, 11 --- f) 0, 1, 3, 5, 7, 9, 10, 12 Cyclomatic Complexity: [Conditions + 1] = [5 + 1] = 6 [Edges - Nodes + 2] = [12 - 8 + 2] = 6	

Add Shortlist

Method: addShortlist()

Parameters: userId, schoolName, schoolNotes

Return: Json response

Purpose: Add the intended shortlist school with comments to the current user database

Basic Paths:

a) 0, 1, 2, 4, 6, 8(baseline)

Test Cases:

a) False outcome of first decision point.

- *Input: ACCESS_TOKEN_SECRET = null*
- The backend server doesn't have and store JWT secret.

b) False outcome of second decision point

- *Input: Token = null*
- User session JWT Token is empty. The user can be in this stage if they have either cleared their browser cookie or logged out.

c) False outcome of third decision point

- *Input: JWTToken = invalidToken. Vis-as-vis*
- User session JWT Token has either expired or is invalid in general. The former is more common.

Execution Paths:

a) 0, 1, 2, 3

b) 0, 1, 2, 4, 5

c) 0, 1, 2, 4, 6, 7

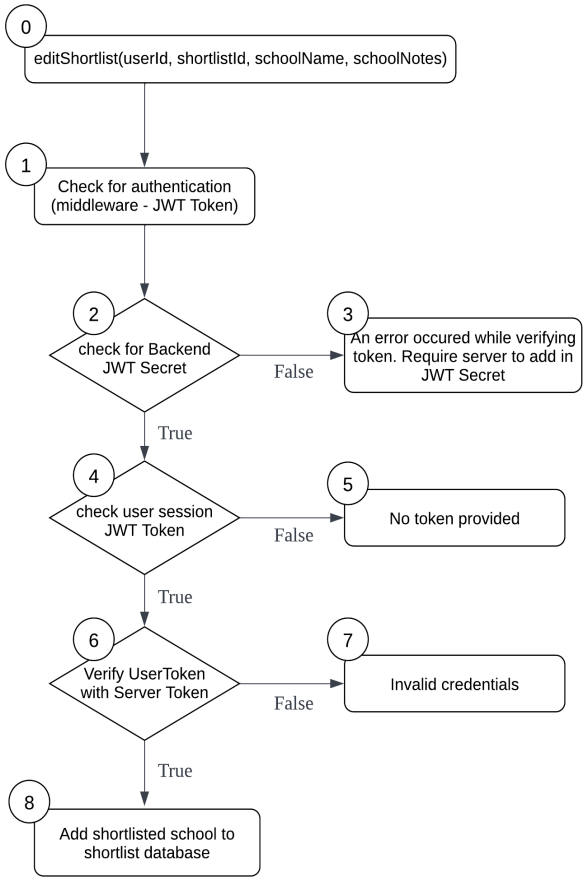
---

d) 0, 1, 2, 4, 6, 8

Cyclomatic Complexity:

[Conditions + 1] = [3 + 1] = 4

[Edges - Nodes + 2] = [8 - 6 + 2] = 4



View Shortlist

Method: getShortlisted()

Parameters: userID

Return: Json response

Purpose: Retrieved all shortlisted schools tagged to the current user database

Basic Paths:

a) 0, 1, 2, 4, 6, 8(baseline)

Test Cases:

a) False outcome of first decision point.

- *Input: ACCESS_TOKEN_SECRET = null*
- The backend server doesn't have and store JWT secret.

b) False outcome of second decision point

- *Input:* JWTToken = null
- User session JWT Token is empty. The user can be in this stage if they have either cleared their browser cookie or logged out.

c) False outcome of third decision point

- *Input: JWTToken = invalidToken*
- User session JWT Token has either expired or is invalid in general. The former is more common.

Execution Paths:

a) 0, 1, 2, 3

b) 0, 1, 2, 4, 5

c) 0, 1, 2, 4, 6, 7

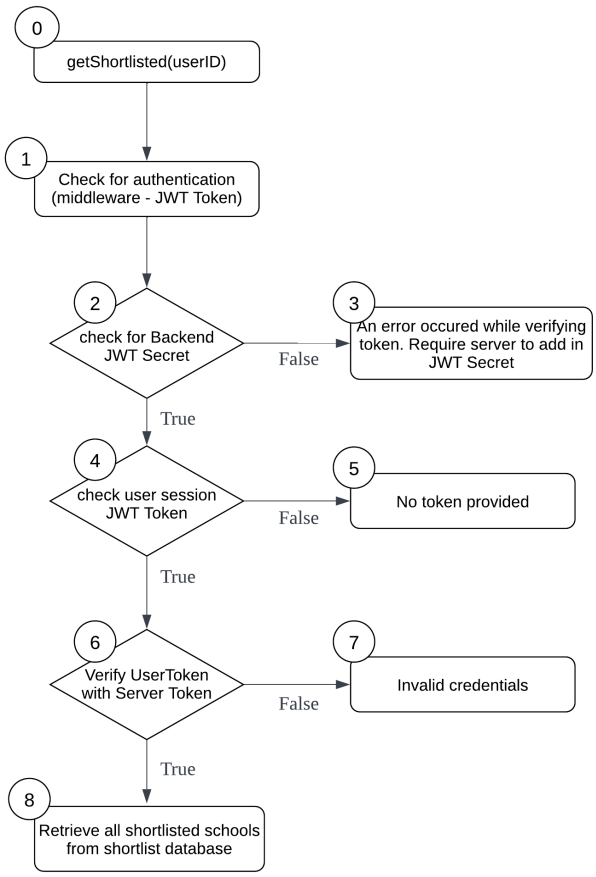
---

d) 0, 1, 2, 4, 6, 8

Cyclomatic Complexity:

[Conditions + 1] = [3 + 1] = 4

[Edges - Nodes + 2] = [8 - 6 + 2] = 4



Initial Functional Requirements

1. Home Page

1.1. The system must be able to display the list of schools based on their corresponding education level offered, indicated by the school's main level code

1.2. The system must provide be able to allow the user to use the search box

1.3. The system must allow the user to be able to filter schools of their choosing based on certain filters

1.4. The system must allow the user to login to their account

1.5. The system must allow the user must to view their shortlisted schools

1.6. The system must allow the user to register their account

2. Search

2.1. The system must allow the user to find and filter schools of their choosing based on

2.1.1. Co-Curricular Activities (CCA)

2.1.2. Elective Programmes

2.1.3. Location

2.1.4. Mother Tongue Language (MTL) offered

2.1.5. Subjects Offered

2.2. The system must allow the user to use the search box to search for schools directly

3. Creating User Account

3.1. The system must allow the user to create an account by providing the following information

3.1.1. Username

3.1.2. Email

3.1.3. Password

3.2. The system must validate that all required information is provided by the user

4. Login

4.1. The system must allow the user to be able to login to their account using their email and password

4.2. The system must be able to check that the email and password are correct

5. Manage Profile

5.1. The system must allow the user to change their password to ensure account security

5.2. The system must allow users to edit their preferences (9. Recommendations)

6. Shortlisting schools (For registered users)

6.1. The system must allow the user to shortlist schools

6.2. The system must allow the user to delete shortlisted schools

6.3. The system must allow the user to view shortlisted schools

7. Comments (For registered users)

7.1. The system must allow the user to create comments

7.2. The system must allow the user to edit comments

7.3. The system must allow the user to delete comments

7.4. The system must allow the user to like comments

8. Displaying School Specific Data

8.1. The system must be able to display the following information about multiple schools based on the filters selected by the user

8.1.1. Name

8.1.2. Address

8.2. The system must be able to display the following information about the school selected by the user

8.2.1. School Name

8.2.2. School Website

8.2.3. Address

8.2.4. MRT Description

8.2.5. Bus Description

8.2.6. Type Code

8.2.7. Nature Code (Co-Ed, All boys, All Girls)

8.2.8. MTL Offered

8.2.9. Subjects Offered

8.2.10. Elective Programmes

8.2.11. Support for Special Educational Needs

9. Recommendation

9.1. System will enquire registered user for their preferences on first time log-in

9.1.1. Region

9.1.2. Mother Tongue

9.1.3. Gender

9.1.4. CCA Categories

9.2. System must display schools relevant to user's selected preferences

10. API

10.1. Interfacing with data.gov.sg

10.2. Interfacing with storage database

Non-Functional Requirements

1. Usability: Each tab will be clearly described so that the users can understand easily.
2. Reliability: System should be able to support up to 50 pax.
3. Performance: The system will be able to respond to user interactions within 2 seconds.
4. Supportability: Documentation and coding comments are consistent and traceable (with version control). The git update will be well commented and described appropriately.