Tutorial Group: SS10: Group 5

Team Name: MiloPeng

NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE & TECHNOLOGY

Contents

1. Product Description

1.1 Purpose

Our goal is to develop an app that makes university applications for Singaporeans much easier, more interactive and interesting. The app should allow users ease of access to information about universities, through search and filter functions that allows users to find universities that fit their requirements. Secondly, our app will also ask for user's academics and conditions, and attempt to match a suitable and eligible university for them based on whether they have a good chance of getting an offer in the university.

1.2 Scope

The application targets students considering both local and overseas studying, and aims to provide them with information available in a clear format through filtering and searching, and also provides algorithmic recommendations based on available categorical scores of universities from our API. Lastly, it serves as a platform for user reviews of universities.

1.3 Users and Stakeholders

With regards to YourFuture, the direct users are young adults looking for suitable tertiary educational institutes such as universities, to further their studies and participate in undergraduate research and projects. It is also geared toward current undergraduates and above, so that they may leave feedback and reviews about the present educational institutions that they are in. This works hand in hand with assisting our direct users in their goals with this app.

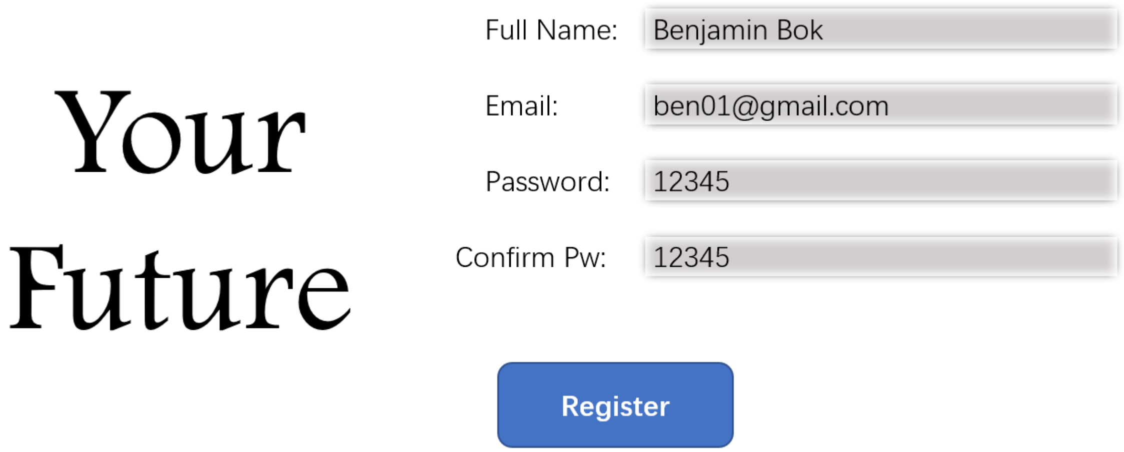
1.4 Assumptions

A few assumptions were made prior and during the development of YourFuture, that were necessary for us to work toward our aims. One of them is that our users will have already completed, or are in the midst of completing their pre-university studies. We also assumed that these users would have a shared set of criteria with regards to selecting the optimal university, such as the quality of teaching and the rate of tuition fees.

1.5 Constraints

There were not a lot of APIs and datasets that complimented our app's function and what we were looking to achieve.

1.6 Initial UI mockups

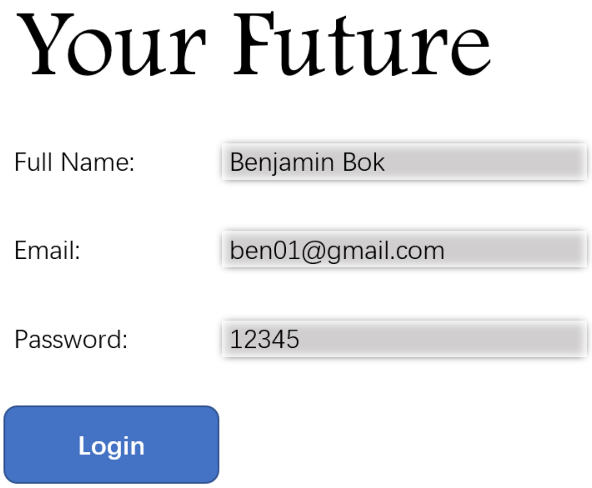


Full Name: Benjamin Bok

Email: ben01@gmail.com

Password: 12345

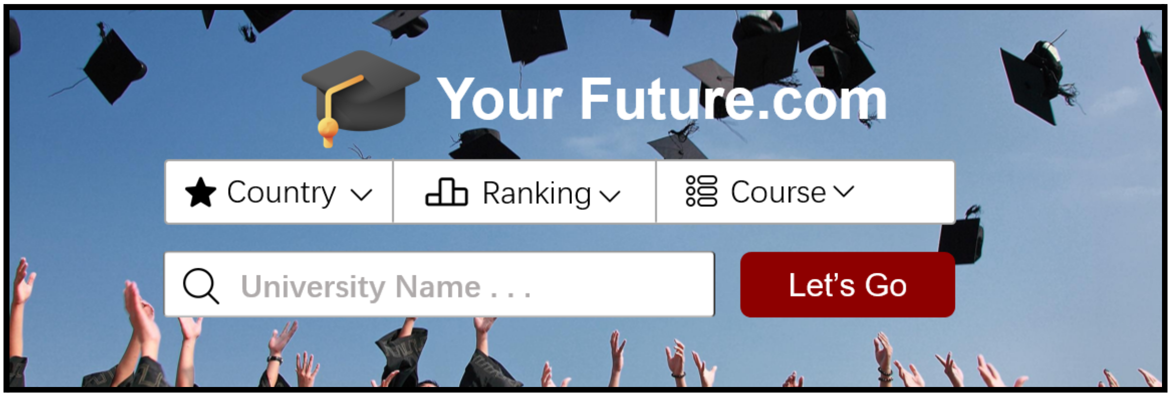
Confirm Pw: 12345

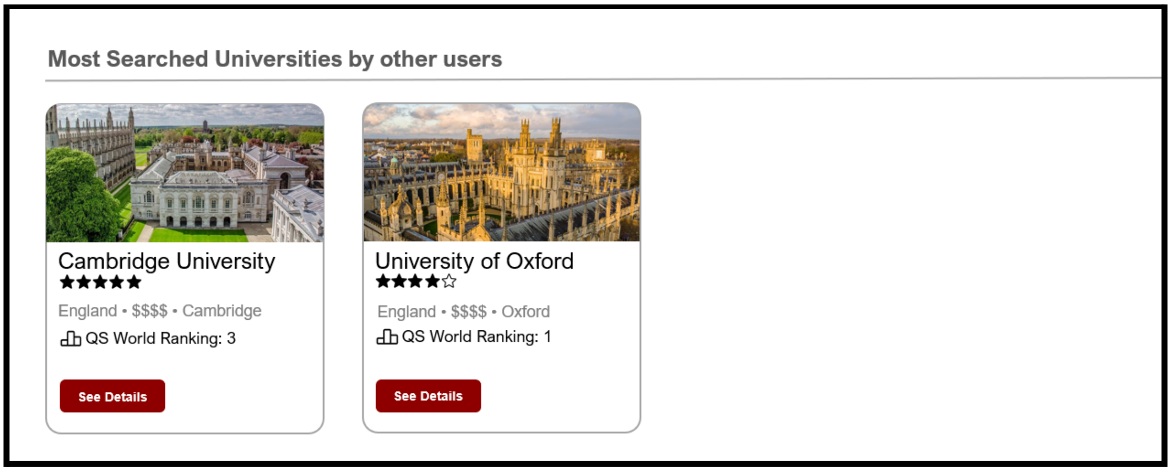


Full Name: Benjamin Bok

Email: ben01@gmail.com

Password: 12345

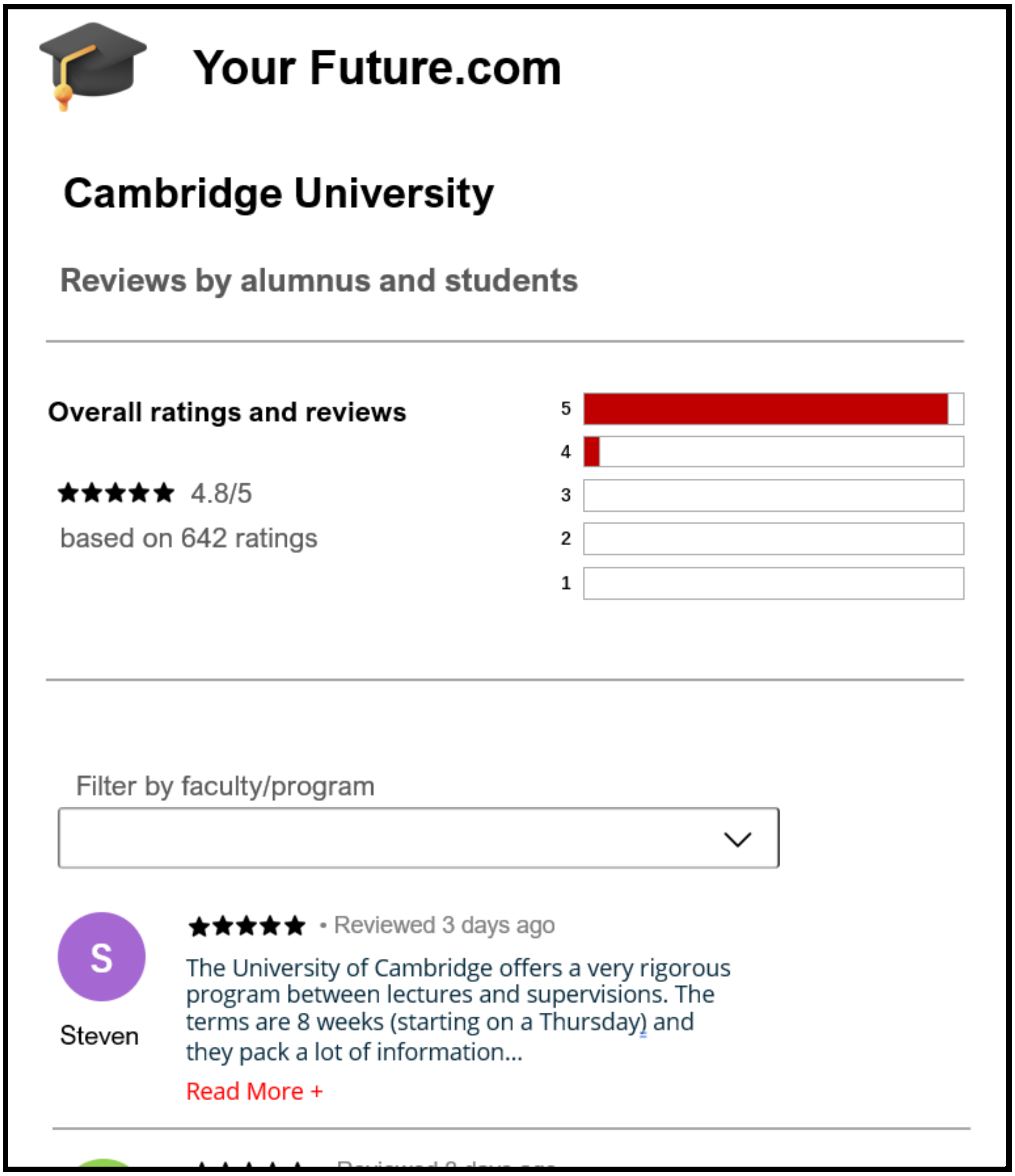




Most Searched Universities by other users

Cambridge University
★★★★★
England • $$$$ • Cambridge
QS World Ranking: 3
See Details

University of Oxford
★★★★☆
England • $$$$ • Oxford
QS World Ranking: 1
See Details



Your Future.com

Cambridge University

Reviews by alumnus and students

Overall ratings and reviews

★★★★★ 4.8/5

based on 642 ratings

Filter by faculty/program

★★★★★ • Reviewed 3 days ago

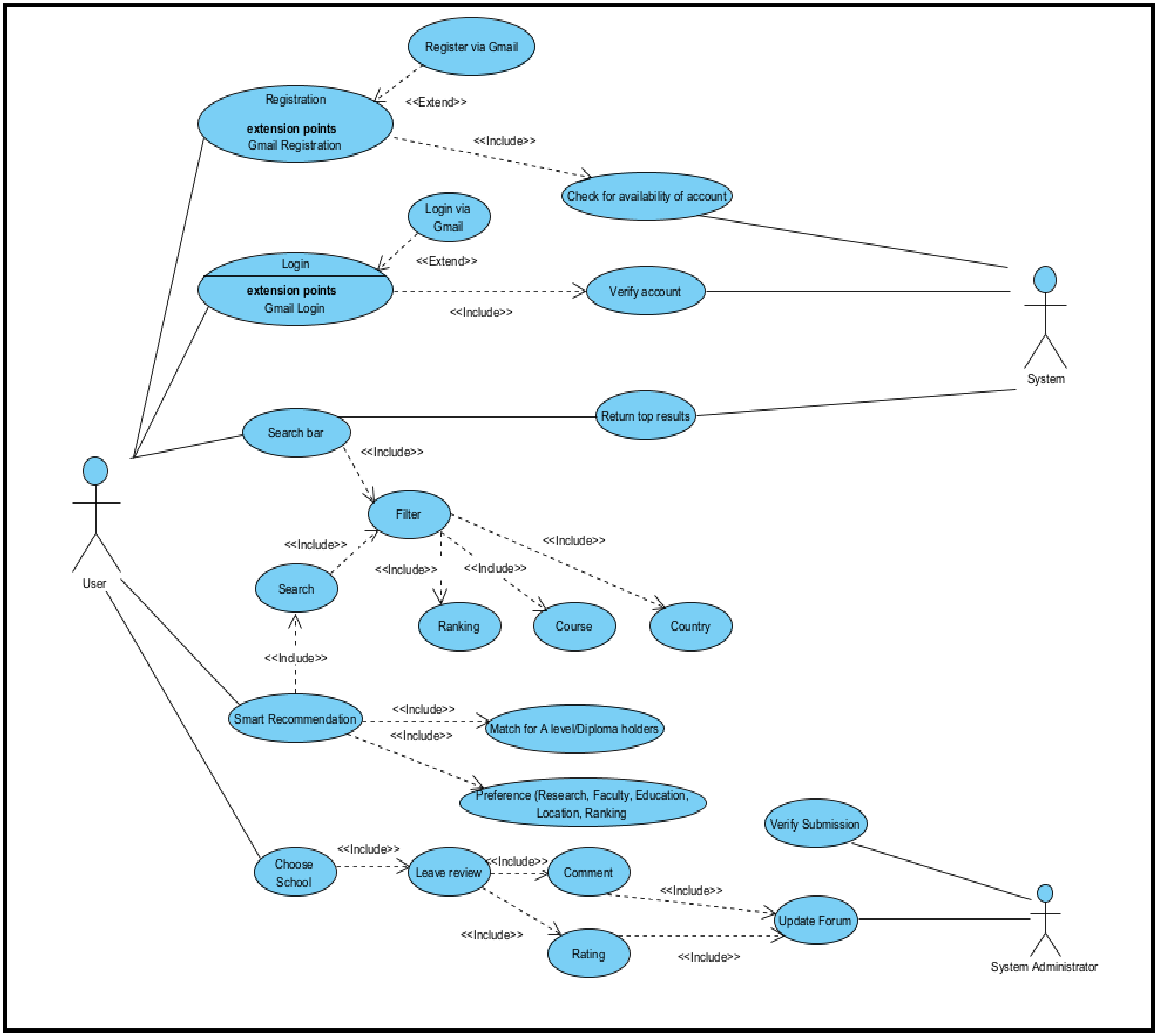
Steven

The University of Cambridge offers a very rigorous program between lectures and supervisions. The terms are 8 weeks (starting on a Thursday) and they pack a lot of information...

Read More +

2. Functional Requirements

2.1 Use-case diagrams



2.2 Use-case descriptions

Use Case ID:	1		
Use Case Name:	Leaving Review of a University/Course		
Created By:	Sharan	Last Updated By:	Sharan
Date:	6/2/2022	Date Last Updated:	6/2/2022

Actor:	User, Review system
Description:	User leave a review of a university
Preconditions:	User must be registered and logged in to his/her account
Postconditions:	User must be enrolled in the school to post his review
Flow of Events:	1. User click on ‘leave a review’ to post feedback on a university’s subpage 2. User leave a rating [on a scale of 1 to 5] by selecting 1-5 number of stars 3. User write a comment 4. User choose a category for the review (e.g., “general”, “accommodation”, etc.) 5. User choose course to review about (e.g. “Computer Science”) 6. System checks if user’s profile is a enrolled student or alumni of the school, if yes, user’s review is uploaded
Alternative Flows:	1. User is not a student or alumni of reviewed school 2. System prompts an error message and redirect user to

	home page

Use Case ID:	2		
Use Case Name:	User finds recommendation of university using the site's recommendation tool		
Created By:	Kay	Last Updated By:	Kay
Date:	6/2/2022	Date Last Updated:	6/2/2022

Actor:	User, System
Description:	User goes through a series of surveys to let the system know of their requirements and details, which the system will use algorithms to give recommendations of universities to the user.
Preconditions:	User must be registered and logged in to his/her account
Postconditions:	User finishes survey and match, and system comes up with recommendation (possible to have no matching universities)
Flow of Events:	1. User clicks on "Recommendation Tool" button 2. System Checks if user is logged in, if yes, directs user to a survey 3. User types in search requirements like Course, country, start-year, before clicking on "Next" 4. System check that user filled in all required fields, if yes, redirect user to "personal match" page

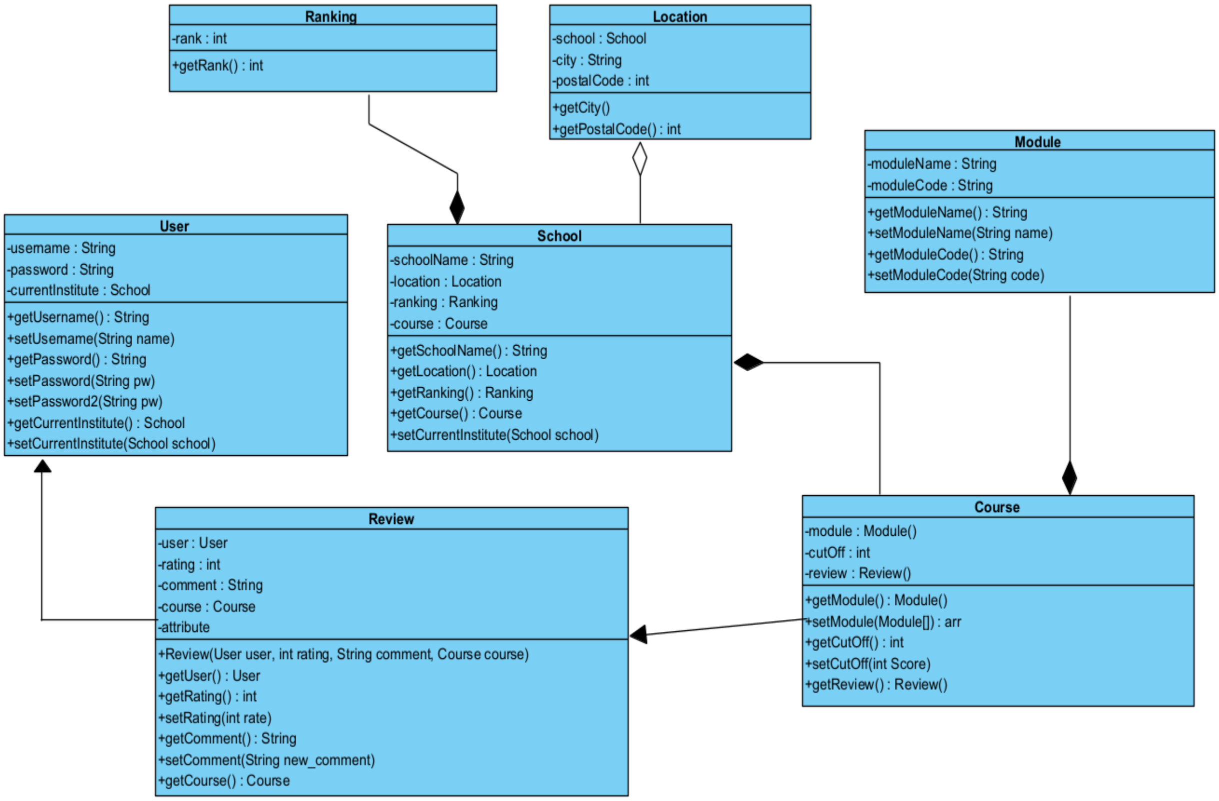
	5. Users fill in information about themselves like prior study level, test results, grades, before clicking on "Next" 6. System check that user filled in all required fields, if yes, direct to "preference page" 7. Users fill in preferences on a scale of 1 to 10 using a sliding tool for categories including "school reputation", "cost", "convenience", and click on "find recommendation" 8. System matches relevant information the user gave and recommends schools to the user from its database.
Alternative Flows:	1. User clicks on "Recommendation Tool" button 2. System Checks if user is logged in, if yes, directs user to a survey 3. User exits survey by clicking on home page icon 4. System saves user's choices for future resuming
Exceptions:	1. System checks if user is logged in, if no, system prompts user to log in before using the function

Use Case ID:	3		
Use Case Name:	User use home page's search function to find schools		
Created By:	Kay	Last Updated By:	Kay
Date:	8/2/2022	Date Last Updated:	8/2/2022

Actor:	User, System
Description:	User uses the home page's search and filter interface to search for universities that satisfy a few conditions
Preconditions:	User must be registered and logged in to his/her account
Postconditions:	User presses the search button and gets results
Flow of Events:	1. User selects course "computer science" from the "course" drop-down filter 2. User selects "United States" from the "country" filter 3. User selects "Within top 100" from the "ranking" filter 4. User clicks search button 5. System comes up with corresponding school results
Alternative Flows:	Alt 1:

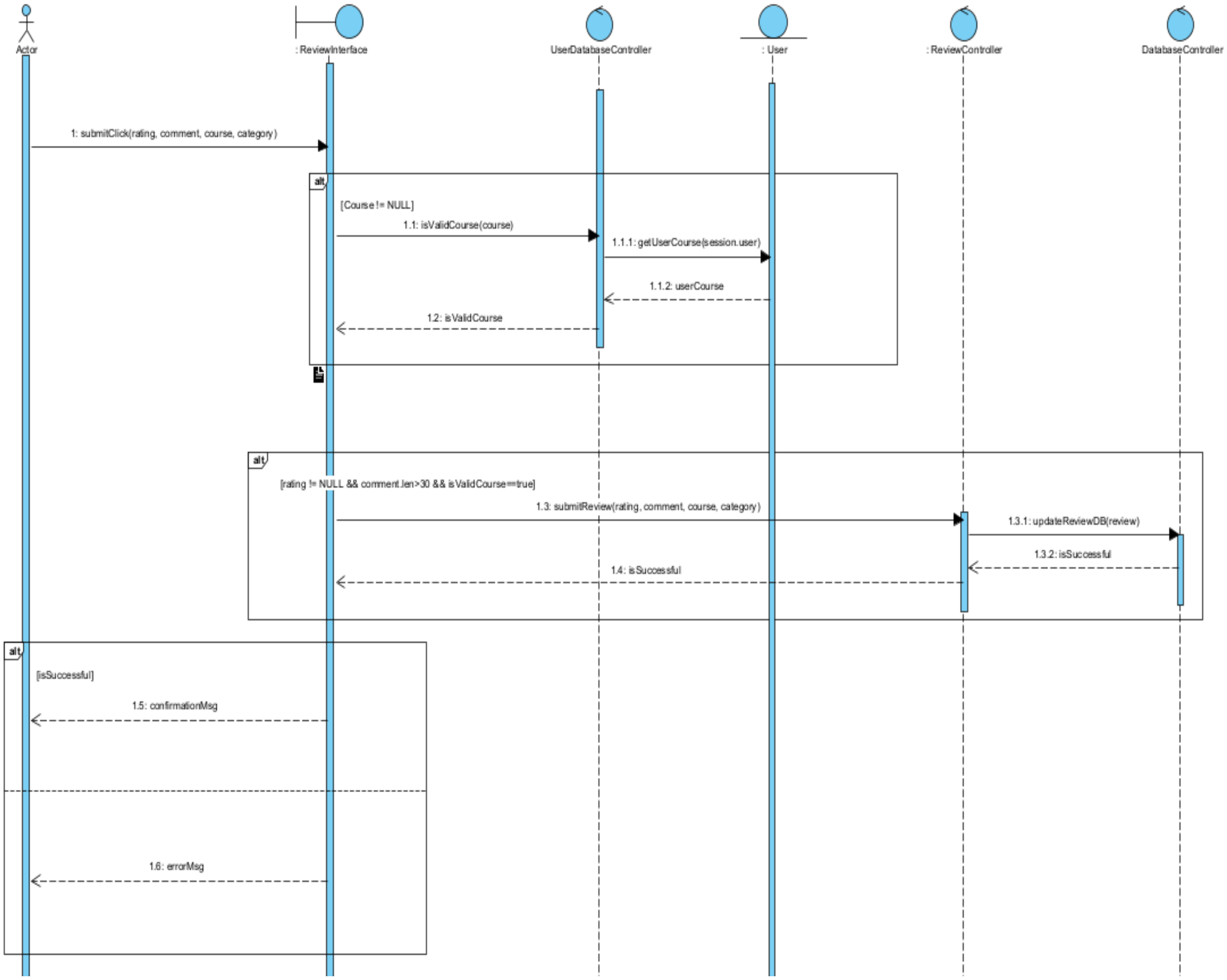
	1. User types in partial name of university 2. System matches name and finds corresponding universities from database 3. System displays them real time in the home page without user clicking on search button Alt 2: 1. User omits certain field (e.g. left “country” blank) 2. User presses search button 3. System makes assumptions and return search result (e.g. assume “All Countries”)
Exceptions:	1. No schools match the user's requirement, the system prompts a pop-up message to tell the user no matches were found.

2.3 Class diagram

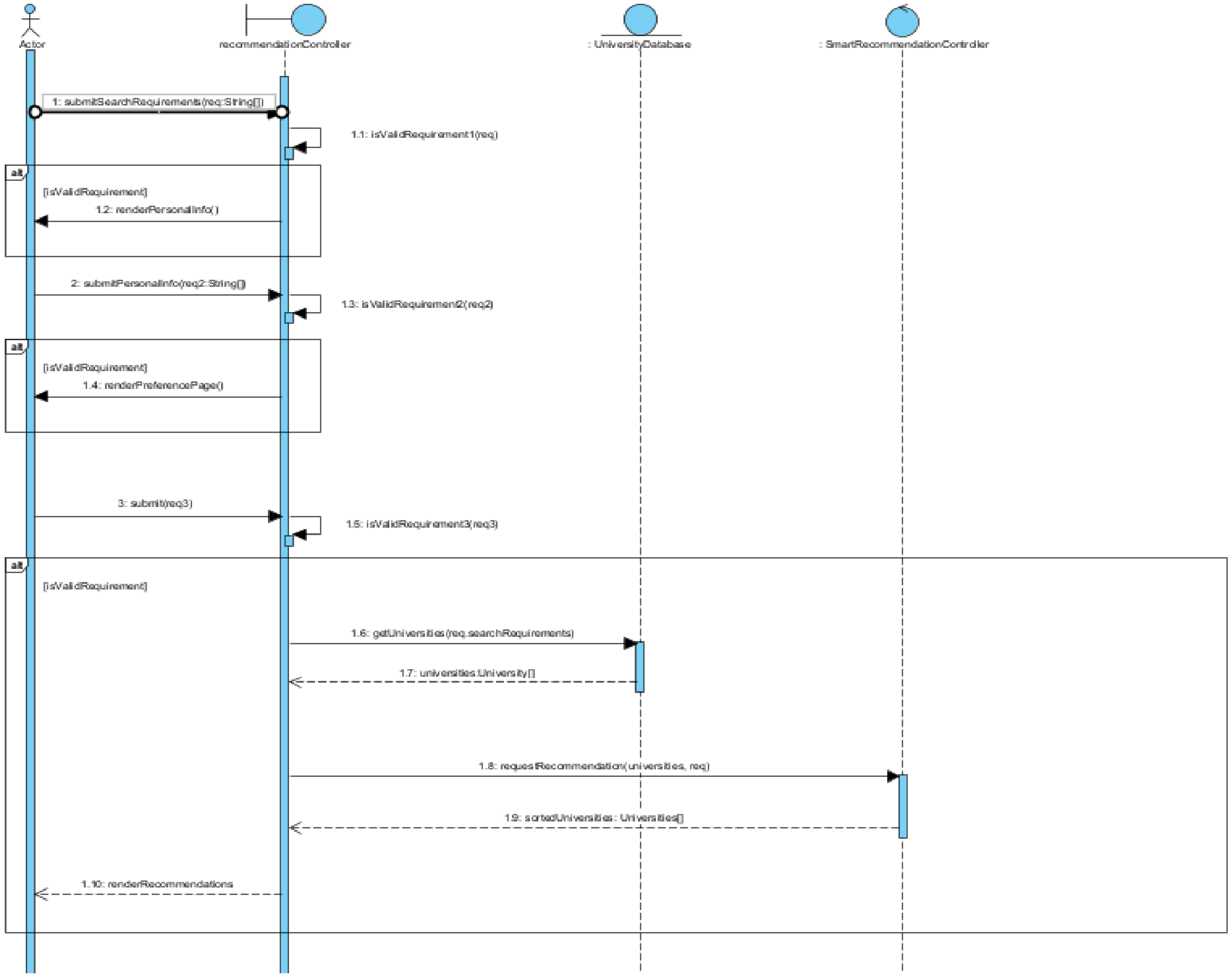


2.4 Sequence diagrams

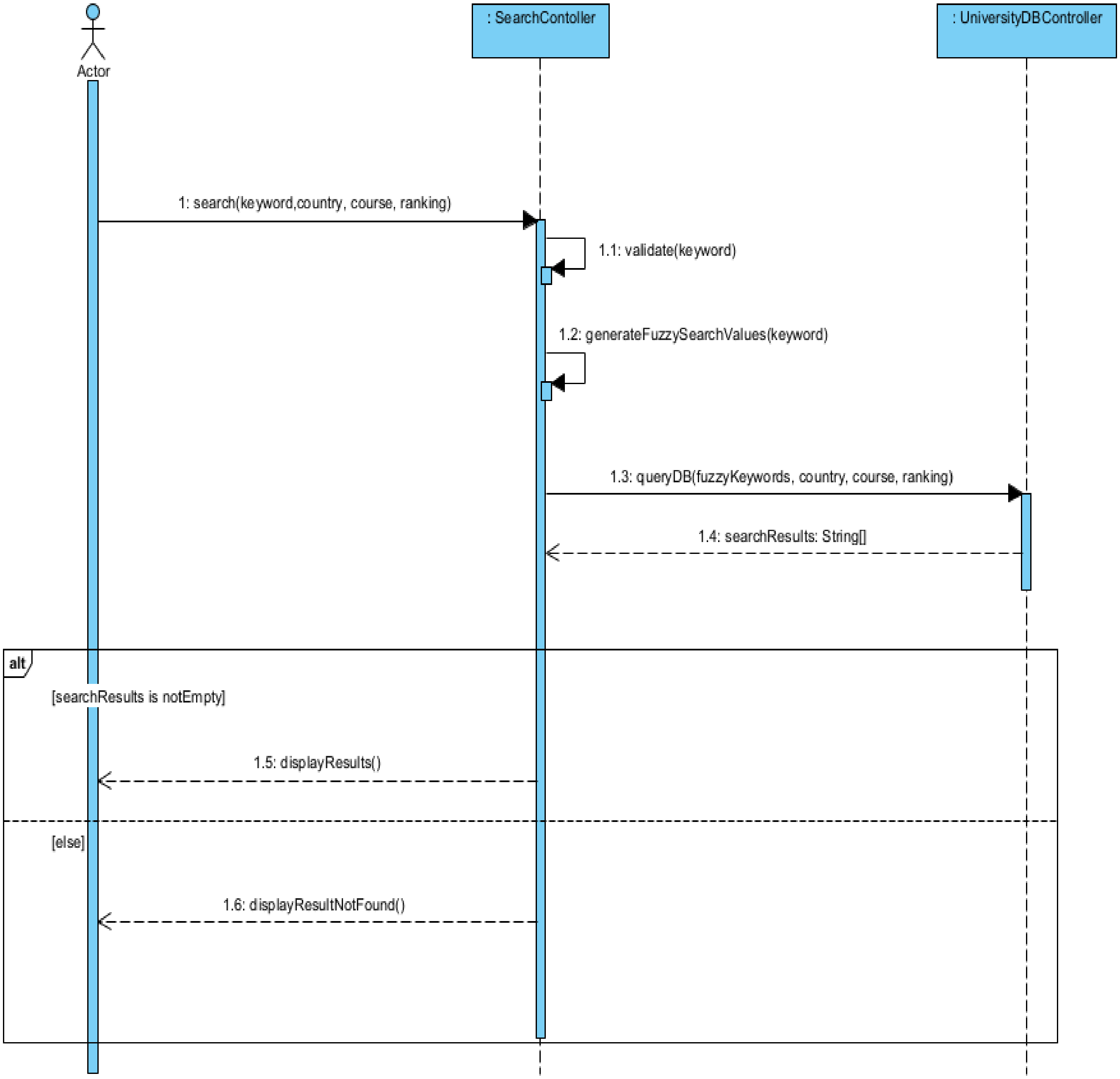
1. User leaves a review or rating of a university


Actor
: ReviewInterface
UserDatabaseController
: User
: ReviewController
DatabaseController
1: submitClick(rating, comment, course, category)
alt
[Course != NULL]
1.1: isValidCourse(course)
1.1.1: getUserCourse(session.user)
1.1.2: userCourse
1.2: isValidCourse
alt
[rating != NULL && comment.len>30 && isValidCourse==true]
1.3: submitReview(rating, comment, course, category)
1.3.1: updateReviewDB(review)
1.3.2: isSuccessful
1.4: isSuccessful
alt
[isSuccessful]
1.5: confirmationMsg
1.6: errorMsg

2. User finds recommendation of university using the site’s recommendation tool

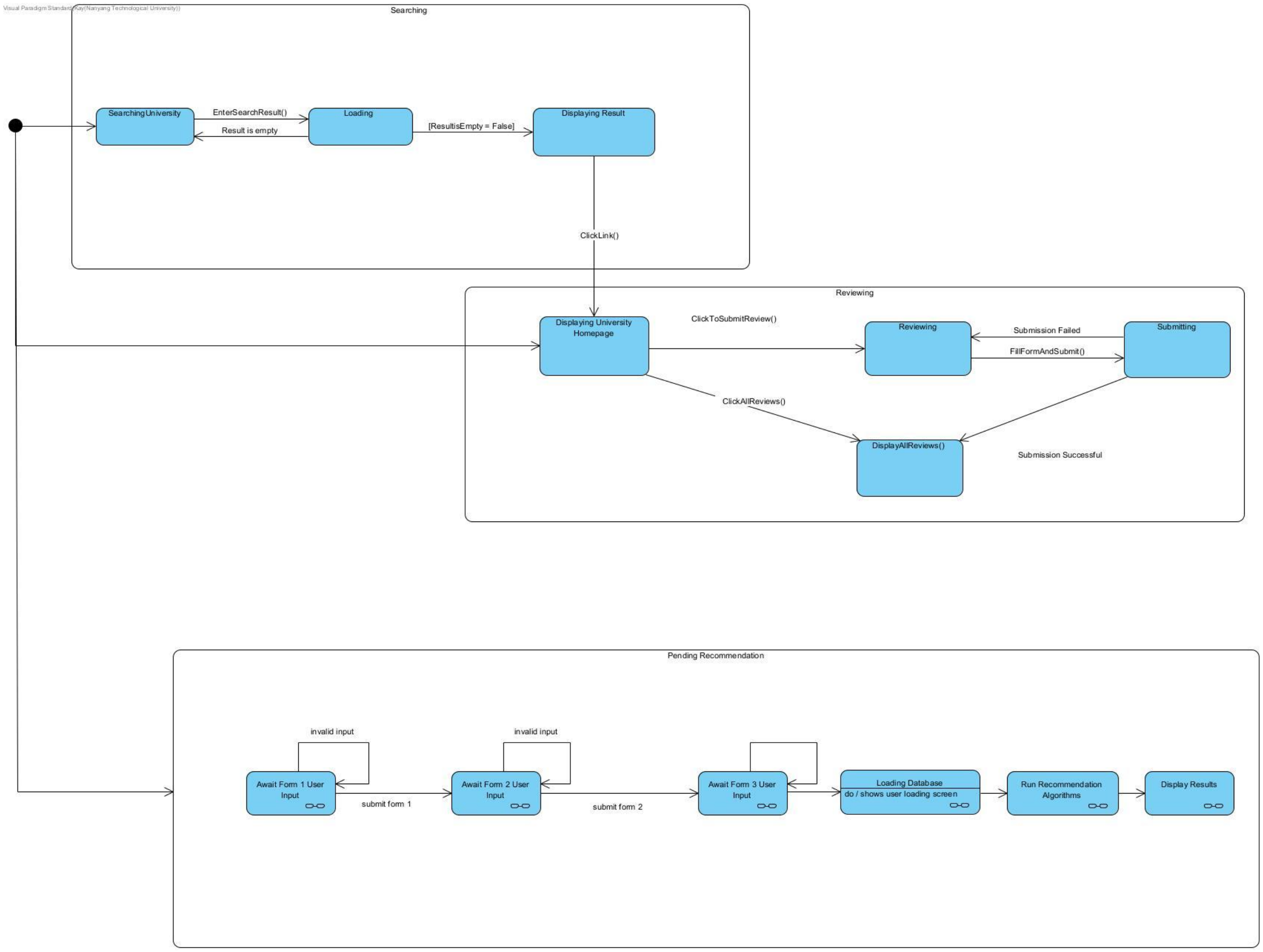

Actor
recommendationController
: UniversityDatabase
: SmartRecommendationController
1: submitSearchRequirements(req:String[])
1.1: isValidRequirement1(req)
alt
[isValidRequirement]
1.2: renderPersonalInfo()
2: submitPersonalInfo(req2:String[])
1.3: isValidRequirement2(req2)
alt
[isValidRequirement]
1.4: renderPreferencePage()
3: submit(req3)
1.5: isValidRequirement3(req3)
alt
[isValidRequirement]
1.6: getUniversities(req.searchRequirements)
1.7: universities:University[]
1.8: requestRecommendation(universities, req)
1.9: sortedUniversities : Universities[]
1.10: renderRecommendations

3. User uses homepage's search function to find university


Actor
: SearchContoller
: UniversityDBController
1: search(keyword,country, course, ranking)
1.1: validate(keyword)
1.2: generateFuzzySearchValues(keyword)
1.3: queryDB(fuzzyKeywords, country, course, ranking)
1.4: searchResults: String[]
alt
[searchResults is notEmpty]
1.5: displayResults()
[else]
1.6: displayResultNotFound()

2.5 Dialog Map

2.5 Dialog Map



3. Non-Functional Requirements

Performance

- System should not crash when the application is open or running.
- System should display a user-selected page or information within 5 seconds of clicking.
- System should only update overnight (SGT) and finish updating before 8am.
- System should be able to maintain and handle high workload

Security

- System must ensure the integrity of the user account information
- System must behave in a predictable and correct manner
- System should mask the password field

Usability

- System must allow ease of returning to the previous page being browsed by the user.
- System must display error messages and feedback when the user makes an incorrect input.
- System must have a consistent layout (text size, colour, font)

Compatibility

- System must be compatible common browsers like Chrome and Mozilla Firefox

Reliability

- System must restore to full functionality within 5 minutes of system reboot.

4. Interface Requirements

4.1 User

Registration page:

- User must be allowed to create an account with their email address
- User must create a 6-18 character length password to register their account.
- System must validate user’s email address and password.
- User must be allowed to retrieve their password by clicking “Forgot Password”.
- System must prompt the user to enter their email address to send the user a link to change password.
- System must send an email with a verification code to the email address of the user to allow them to change password.
- User must be allowed to input the following information during account registration:
    - Name
    - Date of Birth
    - Nationality
    - Educational background
    - Current University (if any)
    - Address

Login page:

- The user must be able to login with their email address and password
- System must confirm that login information is correct
- System must display an error message to the user to re-enter their email address and password if either of it is confirmed to be incorrect
- System must log the user in when login details are validated as correct.
- System must redirect the user to the home page when login is successful.

Search page:

The user must be able to search for schools using the following filters:

- Courses offered by the school
- Distance of school from where they live (Location of school)

User must be allowed to retrieve the information of the school selected. This information includes:

- Courses offered by school
- Reviews of school by former students
- Location of school

Main Page:

- Users must be able to search for universities based on the courses they are looking for.
- Users must be able to search for universities based on the country the university is in.
- The system must recommend to the user the universities and schools that fit all of the user's criteria as accurately as possible.
- The system must recommend to the user the most-sought courses by other users using the search function.

Reviews and Ratings:

- Users should be able to leave review comments for a university.
- Users should be able to rate their current university 1-5 stars out of 5.
- Users should be able to view all comments and reviews left by other users of a university, and be able to filter and sort out comments.

4.2 Hardware

The YourFuture app was designed to be accessible via internet browsers. This would mean that any host containing such programs , be it PCs or mobile phones and tablets, will be able to use YourFuture.

4.3 Software

There is a list of software requirements that are necessary for the proper function of YourFuture. Firstly for the frontend, whereby the user interface (UI), visual displays and interactive objects such as a login input textbox, an open-source JavaScript library by the name of React was used. React allows developers to insert HyperText Markup Language (HTML) code inside the JavaScript code, and is also a viable option for high-quality user interfaces for apps.

YourFuture utilises a MySQL database that is provided and hosted over the Huawei Cloud Service. This database contains the necessary datasets for the app to send select and retrieve requests from. The cloud's services are also utilised to enable the YourFuture app to be hosted on web servers.

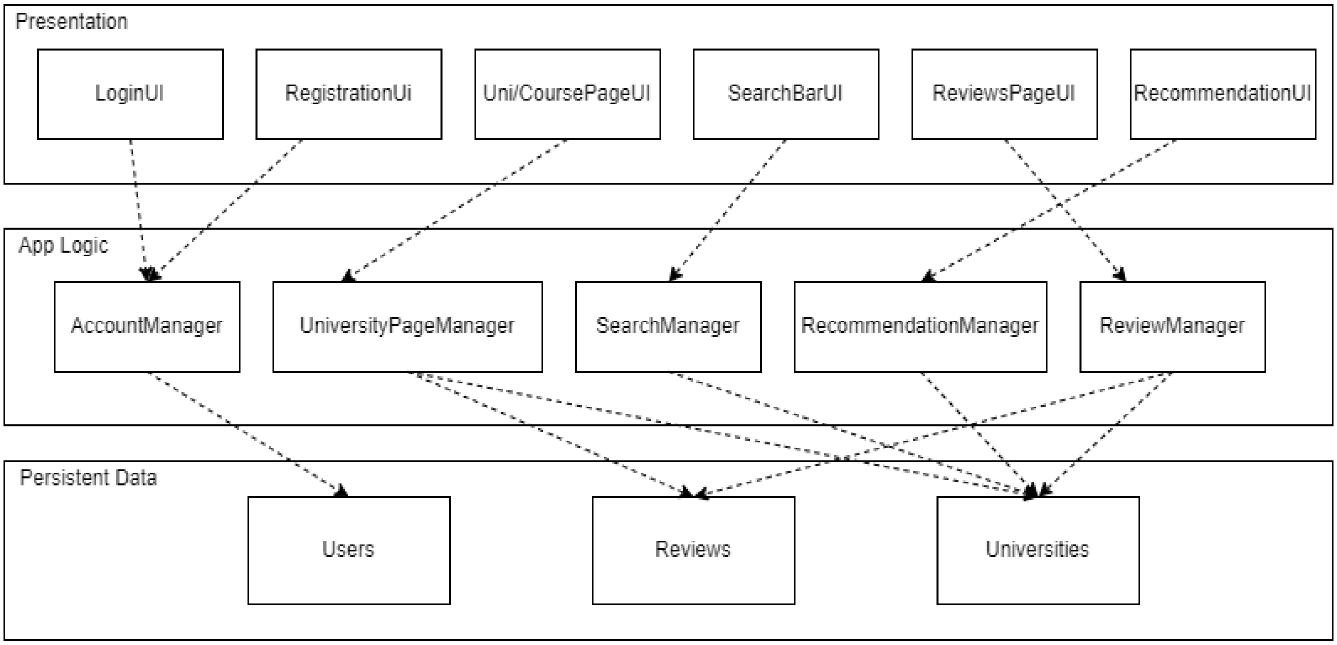
The web application microframework known as Flask was also used. It is useful in providing tools and libraries for web development, such as routing, request handling and so on. It is also key for database object-relational mapping, which means being able to query and manipulate data from a given database, like the one mentioned earlier, using an object oriented programming language; Python.

4.4 Communication

Communication architecture in the YourFuture application follows a client-server model. A REST-enabled web service is used and served over HTTP.

5. Architecture Design

5.1 System Design Architecture Diagram



5.2 Design Pattern

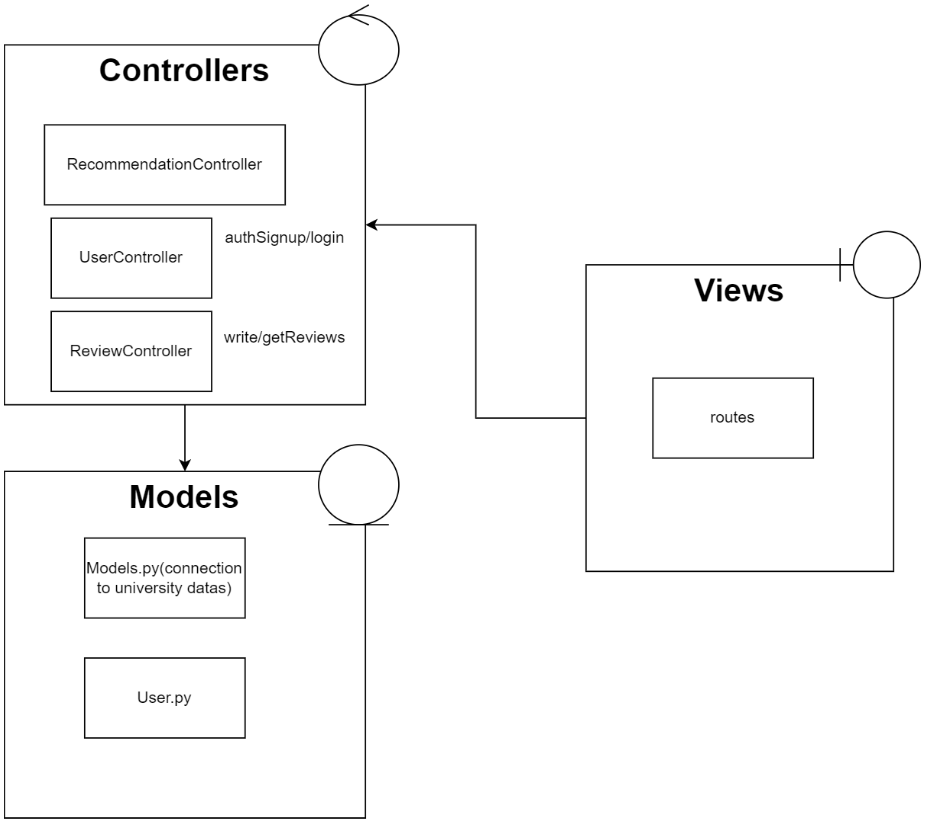
MVC Design Pattern

Problem

There is a lot of repeated and similar code within the logic to connect to the database and make requests from the backend, search requests and recommendation requests have a large portion of code that is similar. These causes a lot of repetitive code and make the backend api very length

Solution

We adopted the MVC design pattern, where the flask api is divided into 3 groups: routes (which is equivalent to views, although this does not render html but just presents the clean data as json), models and controllers. The main routing needs of the api is handled by a Flask blueprint that is registered in the main application. The database connections themselves are established in files in the models folder through mySQLAlchemy, and lastly the operations to connect and make write or read operations to the database is handled by functions exported from the controller folder. This separation of logic allows cutting down of repetitive code, and also allows us to change the universityRequest and recommendationRequest more easily.



6. Data Dictionary

Term	Definition
User	A user is a person who signed up for the application ● Users use the application in the user mode ● Each user must have its username, password and email address ● Users can register his/her education status and address when signing up
Recommendation	A customised list of university recommendation ordered by what the user inputs of their level of interest in specific aspects of a school including quality of education
Education Quality	refers to the quality of lessons and modules of a school
Faculty Quality	refers to the reputation, teaching standards of the teaching crew, professors
Research Performance	refers to the activeness and achievements of the school in terms of academic researches

email	Email address that the user uses to sign in ● Should be in the valid email address format ● Required attribute
ranking	The school's ranking according to the QS University Ranking, either generic or by subject depending on the user's choice
rating	A user rating that is tabulated as the average of all user reviews. It is a value from 1 to 5.
program	A program is an area of study offered by a university
course	Contains all the courses offered by the university, cut-off points and maximum capacity of each course
location	country / city that the school belongs to
Search	Search universities in the schoolList according to the matching rate with the search word
Review	Review of a school on the app by a verified student or alumni user, including a comment and a rating of the school

7. Testing

7.1 Black Box Testing

1. Login

Test id	Scenario	Expected Result	Actual Result
1	Login with unregistered account email	System prompts that email is unregistered	System prompts that email is unregistered
2	Login with a wrong password	System prompt that password is incorrect	System prompt that password is incorrect
3	Login with valid email and password	System prompt that login is successful	System prompt that login is successful

3. Smart Recommendation

- displays recommended universities based on user form

Test Input **Course / Country / user emphasis on education, school facility and research performance**					**Expected Result**	**Actual Result**
Computer Science	All	8	3	3	List of CS schools with moderated scores focusing on education quality	List of CS schools with moderated scores focusing on education quality
Computer Science	All	3	3	8	List of CS schools with moderated scores focusing on school faculty quality	List of CS schools with moderated scores focusing on school faculty quality
All	Singapore	1	1	1	List of Singapore schools with scores that are unmoderated	List of Singapore schools with scores that are unmoderated

4. Searching

- displays universities that fulfil condition

Test id	**Scenario**	**Expected Result**	**Actual Result**
1.	Leaving 'country' blank and search for a valid course(eg. Computer Science)	List of all CS schools is displayed	List of all computer science schools is displayed
2.	Specifying 'USA' and specifying course 'Computer Science	List of all CS schools in USA displayed	List of all CS schools in USA displayed

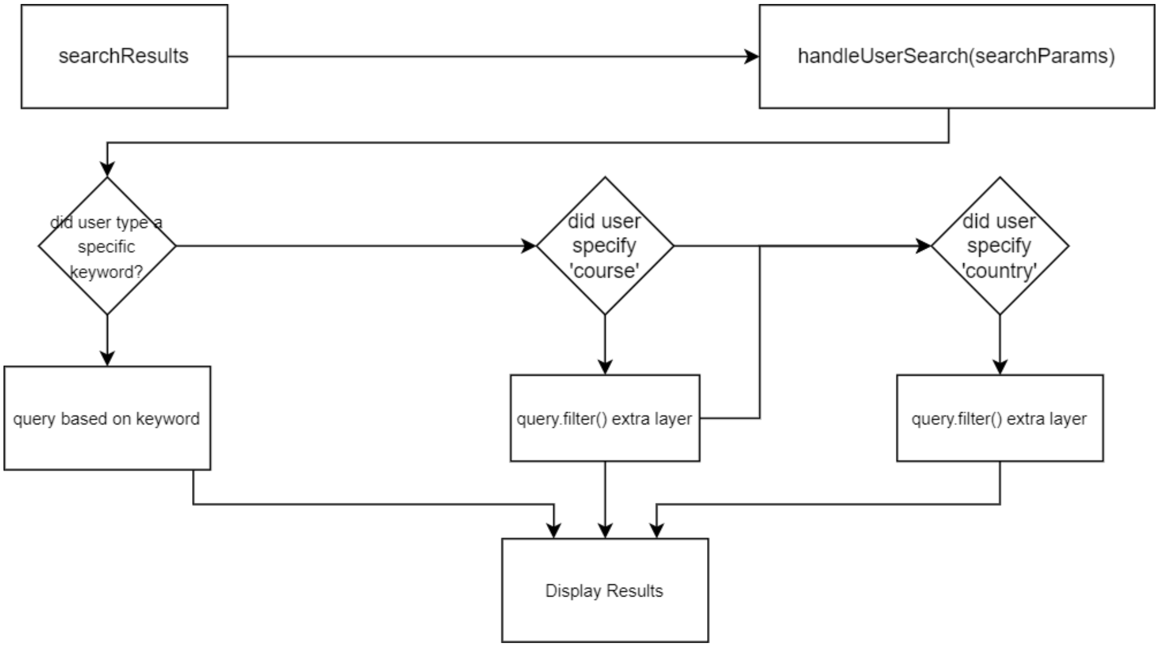
5. Writing Review

Test id	**Scenario**	**Expected Result**	**Actual Result**
1	User writes a review that is more than 200 words	Review is submitted	Review is submitted

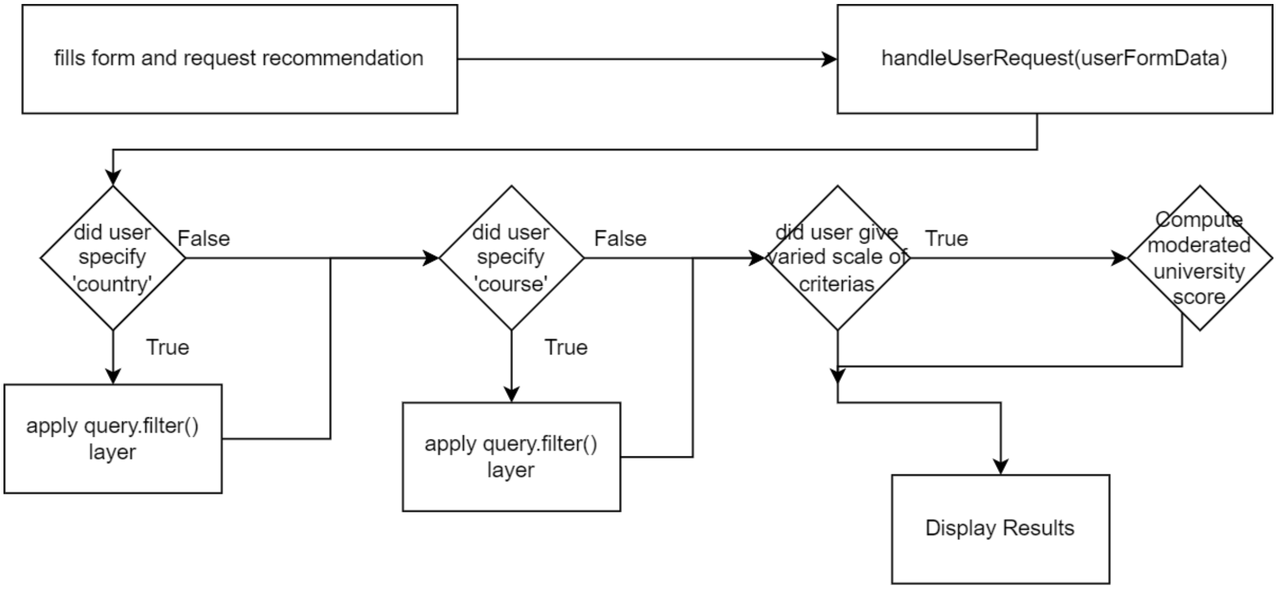
2.	User writes a review that is less than 200 words	User is not allowed to submit the review	User is not allowed to submit the review
3.	A user who did not login writes a valid review	An anonymous review is submitted	An anonymous review is submitted

7.2 White Box Testing

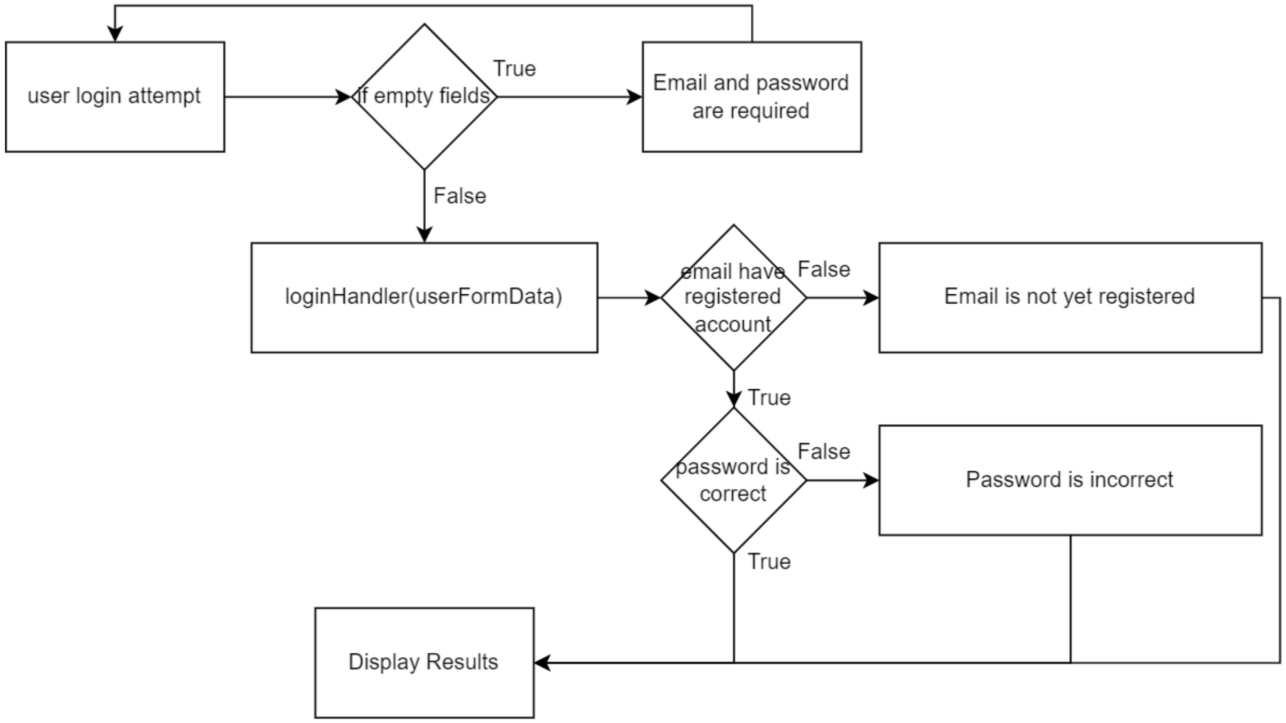
1. Search


searchResults
handleUserSearch(searchParams)
did user type a specific keyword?
did user specify 'course'
did user specify 'country'
query based on keyword
query.filter() extra layer
query.filter() extra layer
Display Results

2. Recommendation


fills form and request recommendation
handleUserRequest(userFormData)
did user specify 'country'
False
True
apply query.filter() layer
did user specify 'course'
False
True
apply query.filter() layer
did user give varied scale of criterias
True
Compute moderated university score
Display Results

3. Login:


user login attempt
if empty fields
True
Email and password are required
False
loginHandler(userFormData)
email have registered account
False
Email is not yet registered
True
password is correct
False
Password is incorrect
True
Display Results

8. Appendix

For more information and detailed demo of the YourFuture app, please refer to the YouTube link below for our video demo:

https://youtu.be/--jp1kW5iX4