CUSP CRUSHER FINAL PROPOSAL

ELEC5619 Object Oriented Application Frameworks

OCTOBER 25, 2018

GROUP 24

Tu, Rufeng 470132199
Wang, Ziqi 460100227
Yao, Wayne 470324934
Zhang, Kun 420033455

Contents

Comments/Mark from draft proposal

Your total mark: 8.9 / 10		Your mark	Explanation
Theme intro	Introduction	-2	The overview of the system should be described as paragraphs, rather than bullet-points
	Significances	-2	The significant should be more detail to stress the importance of your project.
Group implementation	Project outline	-5	The architecture diagram is not clear, and very difficult to read. Not explanation is included for the diagram.
	Project plan	-2	For the plan, it would be good to have more detail about individual component development, such as key milestones.
	Data model		
Individual implementation	Individual features		
	Individual mockups		
Writing	Structure, format		
	English, correct sources		
Total marks		89	

1. Introduction

CUSP Crusher is an online information platform that helps the student users to select their Unit of Study ("unit" or "UoS" in this proposal) and to prepare for the quizzes and exams.

1.1 Aims

Our proposed project aims to create a web application platform that helps students find out information of a unit or a lecturer via reviews. It also allows users to provide reviews on a unit or a lecturer, as well as creating and viewing sample questions for the preparation of quizzes or exams.

1.2 Overview

The platform provides detailed information of all units and lecturers in School of Information Technology in USyd to any user. First time users need to register in order to get access to contents of the platform. Upon a successful log-in, registered users will be presented by a user profile page where they can view and edit their user information. The profile page also serves as a portal to direct users to other components of the website, including Unit Review, Lecturer Review and Sample Question.

A Unit Review page displays reviews and information about a unit, as well as allowing users to create, edit, and “upvote” reviews. A Lecturer Review page offers similar functionality, but with the context being a lecturer. To facilitate information access for users, a search function will be implemented where the users can find any unit review page by name or unit code, or any lecturer review page by name. The platform also allows registered users to create and share sample questions, view and solve questions made by others, as well as upvoting questions to their likings. Users can keep track on their own reviews and sample questions they have created in the user profile page.

1.3 Primary User

Our target users would be USYD students who are overwhelmed by different units at the beginning of a semester, who ask like crazy on social media “how’s this unit? Is it hard? Is it programming involved? Is the final torturing?”. In addition, our platform suits USYD students who want to get prepared for the quizzes and exams by self-testing online instead of going through slides for a hundred times and finding out that none of them appears in the exam.

2. Significances

CUSP, or Course and Unit of Study Portal is a subsite of The University of Sydney's main website that enables students to view details of courses and units offered by their Faculties. Albeit important, CUSP is poorly design with poor navigation, excessive irrelevant information, ill-formatted content and lacklustre visual design, that effectively hinders users experience to a large extent. The project is also inspired by the common scenarios in the online communication chatting group or social media group, where students interested in a certain unit constantly ask others about the detail information and some comments of the unit or the lecturer. Most of the time, the answers to those questions are not easy to find on traditional CUSP. Although there are some websites providing the comments on a certain unit, the rating systems are too general. For example, the website studentvip.com only provide one overall rating on the unit, and there is no rating or comments on the lecturer, nor do they have teaching history of lecturers [1].

Our proposed project, CUSP Crusher, aims to address the issues of CUSP, by creating a platform that helps users find out information of a unit or a lecturer in a faster, more concise and more comprehensive manner. The platform will contain information that our primary users, USyd students concern the most (e.g. lecturer's name) and leave out the insignificant ones (e.g. objectives) so that they will not be befuddled by the massive content of a page. By providing a ratings system, in terms of "Stress", "Difficulty" and "Usefulness" of a unit and "Teaching Ability" and "Responsiveness" of a lecturer, the platform offers the users a more intuitive way to evaluate the unit or the lecturer. Combing with the reviews, the users will be able to get a comprehensive view with first-hand experiences from past students, and hence be able to make more informed decisions on whether to select the unit.

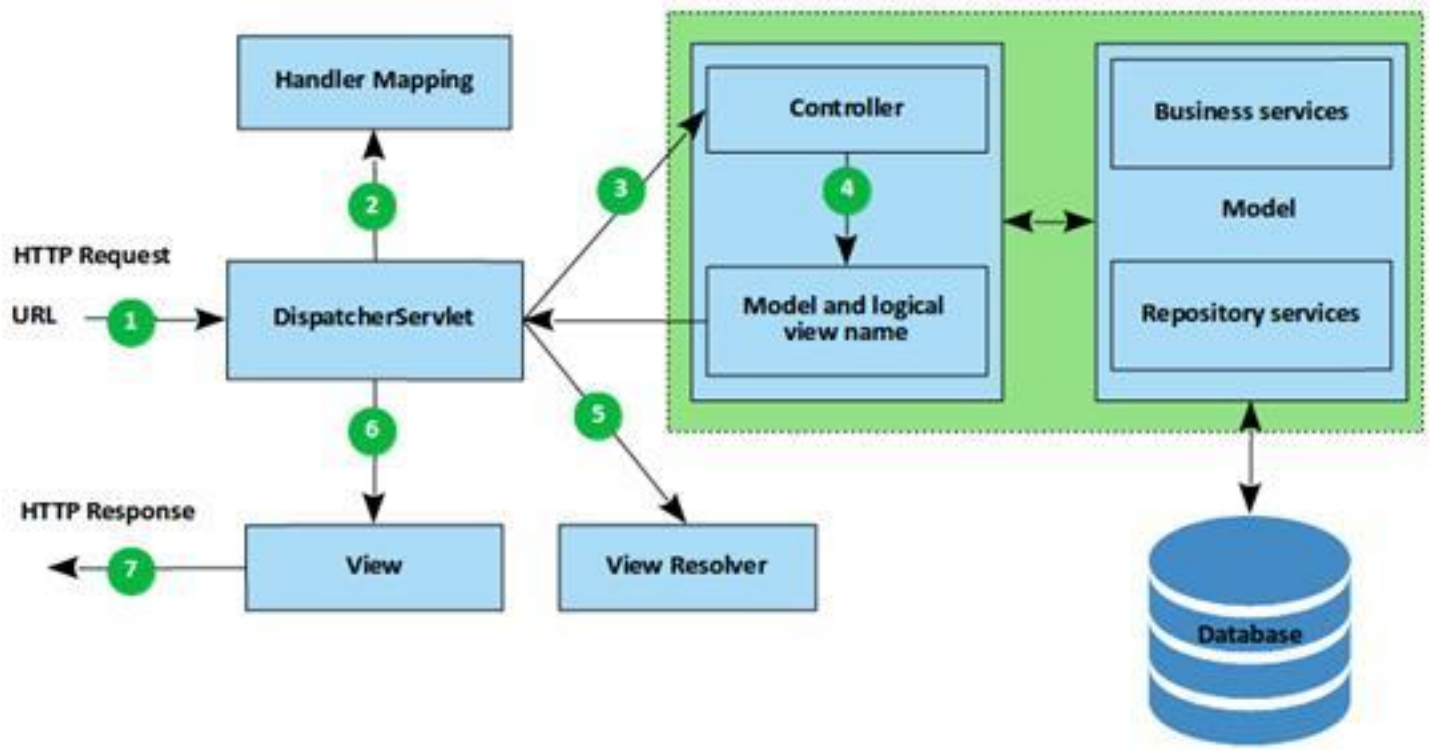
In addition, our website provides a feature that user could create sample questions, share with others and have self-tests with private and public sample question bank. This function is also unseen in all the existing unit rating websites. Instead of only reading through the slides and working on past exams, the users can use sample questions as an alternative to find out how well they have prepared their exams; they can also learn how other students have prepared and find out what examinable knowledge is missing from their own preparation.

3. Project Outline

3.1 System Architecture

The project primarily uses Spring, Maven and Hibernate as backbone frameworks, Tomcat 8.5 as the Java Servlet Container, h2 for data persistence and Thymeleaf as the template engine. On top of that, Spring Boot is chosen to simplify the bootstrapping and development of the project. Languages, tools and frameworks on the client-side include HTML, CSS/SCSS, Bootstrap, JavaScript (ES5/6), jQuery, etc.

The server-side system architecture follows the typical Spring MVC design.



System architecture Description

Spring MVC, which builds on top of Java Servlet, handles each request in a separate thread. In our project, the Request Dispatcher of Tomcat Servlet Container Dispatcher forwards an incoming HTTP request to a corresponding controller by consulting a predefined handler mapping. The controller interprets user's requests and maps them into actions to be performed by Model. The Model consists of business services and repository Services. The business services are responsible to perform business logics and will fetch any required data from the database through repository services which manage our h2 database through ORM framework Hibernate that extends Spring Data JPA. Upon successful execution, business services will return the results to the requesting controllers who then forward them to corresponding views. The Thymeleaf view template engine parses the results and template pages written in Thymeleaf syntax into browser readable contents which are send back to the client-side as a HTTP response.

3.2 Component division

As mentioned in previous description, CUSP Crusher is an online information platform that helps the student users to select their UoS and to prepare for the quizzes and exams. Based on this guideline, the main component can be divided as follows.

3.2.1 The user registration system (managed by Wayne)

The main features of this part include registration, login and management of user profile. In this part, users are allowed to manage their personalized information. Also their previous given comments and lecture rating will be displayed as well.

3.2.2 The units reviewing system (managed by Kun)

The main features included in this part are showing key static information of a unit, showing unit reviews and rating the units in terms of difficulty, stress and practicality. With the help of this system, users will be able to decide whether a UoS fits their needs for the study of the current semester.

3.2.3 The lecturers reviewing system (managed by Rufeng Tu)

The main features of this system are showing the teaching history of a lecturer, rating the lecturer in terms of year of the review, teaching ability as well as responsiveness. The users can use information provided by this system as reference while selecting a UoS given by a certain lecturer.

3.2.4 The questions creation and reviewing system. (managed by Ziqi Wang)

The main features in this part include Reviewing question creation, taking quiz, viewing your questions and question discussion board. Users can create question and take quiz related to a UoS for the purpose of either improving their understanding or preparing for exams.

4. Project Plan

4.1. Project planning stage (14/08/2018 - 20/08/2018)

At this stage, brainstorming about what we're going to do, defining what kind of problem we are going to resolve and how we'll solve it. Define the development methodology (probably waterfall model). The output of this stage is the raw idea that is embed in everyone's mind about our website.

4.2 Development plan stage (25/08/2018 - 25/09/2018)

4.2.1 Logic design and individual feature implementation (25/08/2018 - 28/08/2018)

Assign team member with different features and logic. Team member can volunteer to take the responsibility of a specific feature and discuss the difficulty he might be faced with. Also, the workload is another consideration. We should guarantee that the workload for each member is roughly the same.

4.2.2 Data model design (29/08/2018 - 25/09/2810) Milestone

Determine the overall database structure design. Upon everyone agreeing on a main framework of the data model, they can add necessary attributes and entities to the framework based on their assigned feature. The result of this activity is a complete ERD and ER relation model that will be used in the following development. This stage is a milestone.

4.3. Development (26/09/2018 - 16/10/2018)

4.3.1 Individual development (26/09/2018 - 14/10/2018)

All team members are dedicated to the implementation their features in this stage. We'll need to complete a frontend page, which is responsible for the presentation of data and our business logic. We'll also need to program the actual backend logic. E.g., clicking this button will trigger an operation in database and redirect user to another specific page with specific content. Due to the nature of those separated features, we may or may not have overlapped needs such as some database operation or some common utility functions. Hence, it's important not to implement the same function twice. To resolve this problem, we decided to interactive with each other in API fashion. Once a member has some API ready, he announces it to the whole team. Note that dangling function will be gathered in a single place (e.g. tools/ or utils/) for maintenance purpose.

4.3.2 Individual testing (10/10/2018 - 14/10/2018)

Team members are to unit testing their code to make sure the functionality and usability of their unit with proper testing code. This stage mainly focuses on if APIs are stable and robust and if APIs can handle error-prone input correctly. Bugs that are to influence main business logic are the highest priority issue and must be fixed as soon as possible. Upon the closure of this substage, we can proceed to integrity testing.

4.3.3 Integrity testing (15/10/2018 - 25/10/2018)

The integrity testing mainly focuses on whether different units of code properly works together to produce expected result, whether the coherence and cohesion of the product of different members make the whole website an integrity object. Most importantly, is the functionality of the website is

fulfilled. I.e., whether the correct input yields reasonable result or more precisely, whether the coordination of different APIs is smooth. This stage is considered as milestone as it marks the birth of stable version of our website.

4.4 Post maintenance (26/10/2018 - 29/10/2018)

The post maintenance stage involves the completion of code document. Team members are expected to produce a copy of detailed document of their code on the functionality, exception handling and possible breakage of API in the future. The closure of this stage is a milestone as it formally indicates the termination of this project.

4.5 Individual Key Milestones

The milestones of the project are summarized in the table as follows.

Date	**Milestone Description**	**Responsible Individual**
09/01/2018	Project Plan enacted	Wayne Yao
09/03/2018	Data model design finalized.	Wayne Yao
09/24/2018	UI design finalized.	Kun Zhang
09/25/2018	System architecture design finalized.	Kun Zhang
09/10/2018	Lecturer Review component development completed.	Kun Zhang
14/10/2018	User Registration and Profile component development completed	Wayne Yao
14/10/2018	Unit Review component development completed	Rufeng Tu
14/10/2018	Question component development completed	Ziqi Wang
17/10/2018	Unit tests completed for each component.	Each member for each component
21/10/2018	Integration completed.	Rufeng Tu
24/10/2018	Integration test completed.	Ziqi Wang
27/10/2018	Project documentation written.	Kun Zhang
31/10/2018	Project delivered.	Wayne Yao
2/11/2018	Project demonstration video made and delivered.	Kun Zhang

5. System Data Model

The ER Diagram of the whole database is the following: (next page)

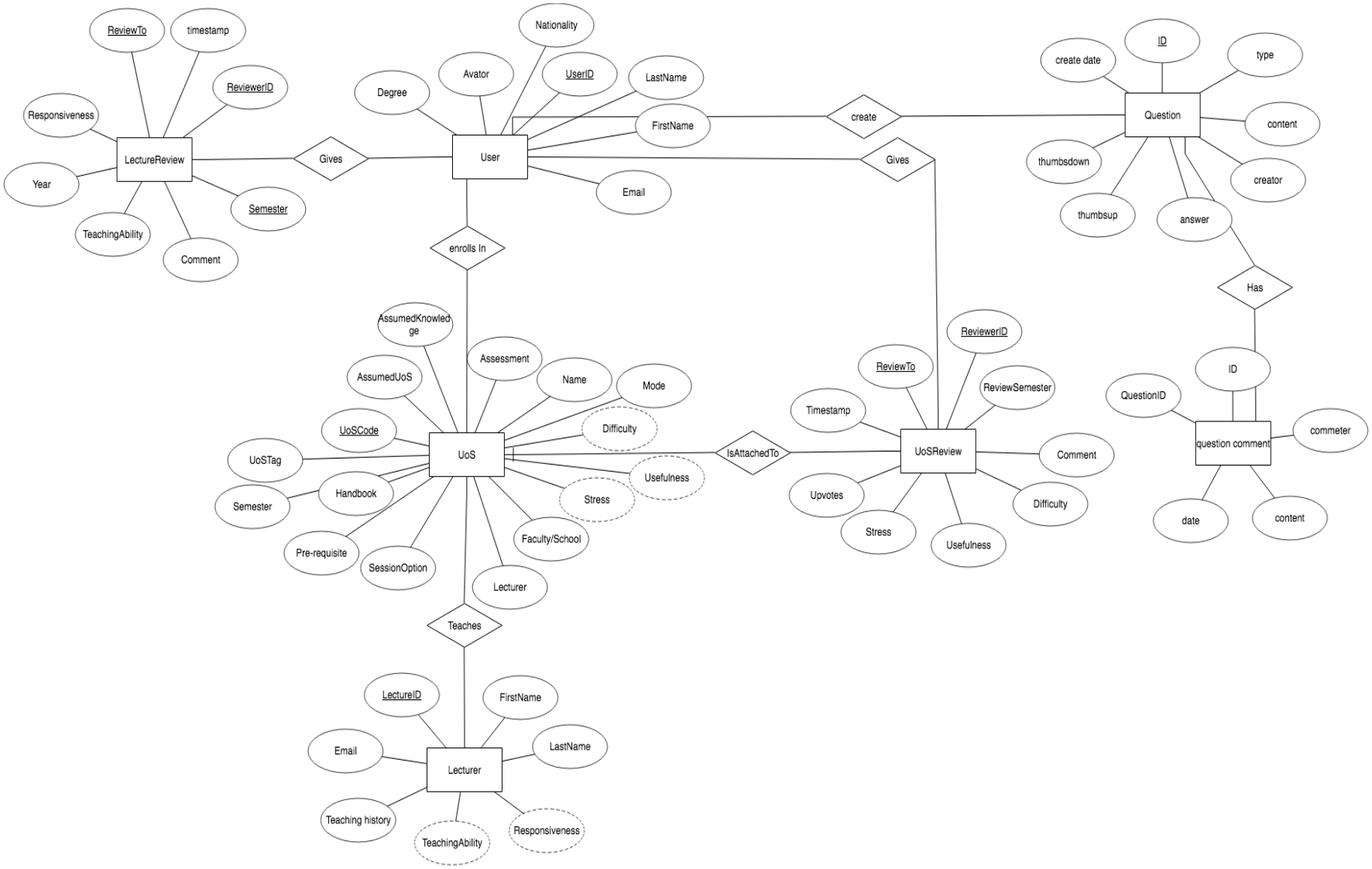
Some notes about this ERD:

1. We create 7 entity types to map our business logic to the data type we're going to store. Their names should be self-explained.

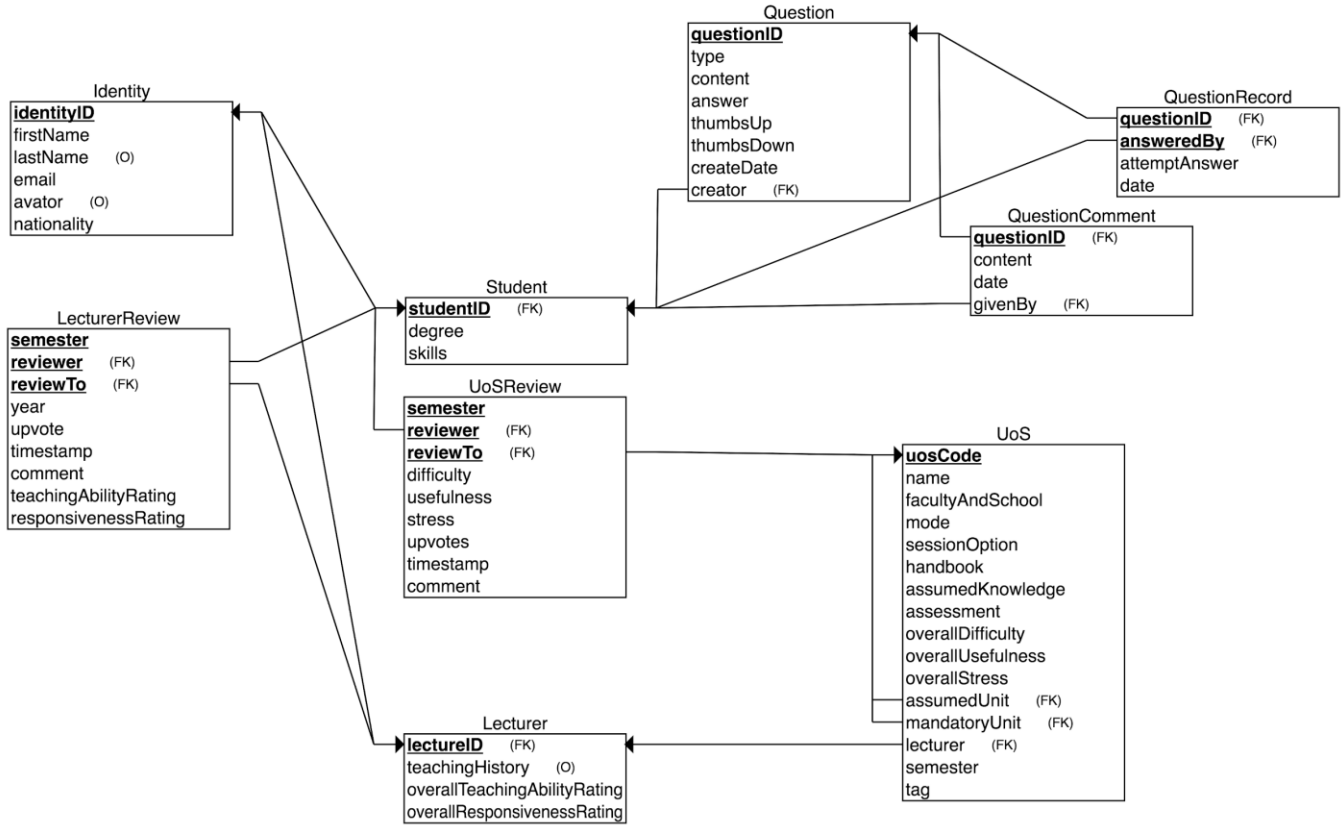
- User: our registered student.
- UoS: our units that is to be reviewed and commented.
- Lecturer: our lecturer that is also to be reviewed.
- LecturerReview and UoSReview: entity type that can represent the meta data of action "review". E.g., A UoS review may contain such information: who gave this review (reviewID), which unit this review is related to (reviewTo), what aspects of this unit is actually reviewed (usefulness, stress, difficulty), did this review upvote this unit (upvote). The same logic applies to LecturerReview except that the subject is lecturer now.
- Question: a user created question used by self-testing and is public to others (so that they can use this question to test themselves as well).
- Question comment: comment on a specific question.

2. The tool we use to create this ERD doesn't use standard notation to represent the cardinality of the relationship of two participating entity type.So we formally make an explanation here:

- A user can give 0 or more review to UosReview/LectureReview. A UoSReview/lectureReview must be given by exactly one user in the same semester.
- A user can enrolls in 0 or more UoS and vice versa.
- A user can create 0 or more questions. A question can only be created by exactly one user.
- A lecturer teaches 0 or more UoS. A UoS must be taught by at least one lecturer.
- A UoS is attached to 0 or more UoSReview. A UoSReview must be attached to exactly one UoS.
- A question has 0 or more question comment. A question comment must be attached to exactly one question.


ReviewTo
timestamp
ReviewerID
Responsiveness
Year
TeachingAbility
Comment
Semester
LectureReview
Gives
Degree
Avator
Nationality
UserID
LastName
FirstName
User
Email
enrolls In
create
Gives
create date
ID
type
content
creator
thumbsdown
thumbsup
answer
Question
Has
ID
QuestionID
question comment
commeter
date
content
AssumedKnowledge
Assessment
AssumedUoS
Name
Mode
Difficulty
UoSCode
UoS
UoSTag
Handbook
Semester
Pre-requisite
SessionOption
Faculty/School
Lecturer
Usefulness
Stress
IsAttachedTo
ReviewerID
ReviewTo
ReviewSemester
Timestamp
UoSReview
Comment
Upvotes
Stress
Usefulness
Difficulty
Teaches
LectureID
FirstName
Email
LastName
Lecturer
Teaching history
TeachingAbility
Responsiveness

The ER relation model translated from ER diagram is the following:



As indicated in the picture, this ER model contains 9 tables that are directly translated from ERD. Note that due to the limit of the tool we use to create ERD, we cannot represent “ISA” relation in the ER diagram but we implement it in our ER model: the entity type “Student” and “Lecturer” is a “Identity”. This separation makes logic clear. Besides these three tables, the “UoS”, “UoSReview”, “LecturerReivew” constitute the core tables that support our main business logic. Users can comment, review, find units or update, delete their history with the support of these six tables. The three tables at up right corner (“Question”, “QuestionRecord” and “Question comment”) are responsible for another feature of our website. The quiz and self-testing feature will need them. Users can create and answer questions, view answer and comment of this question with the support of these three tables. Semantic constraints and data type will be implemented and determined in our actual SQL transaction.

6. Individual Component Implementation

6.1 Login/Registration and User Dashboard (Wayne)

I'm mainly responsible for the design and implementation of login page and user dashboard presentation with proper security measures. The features of these two pages are the following:

Login/Registration

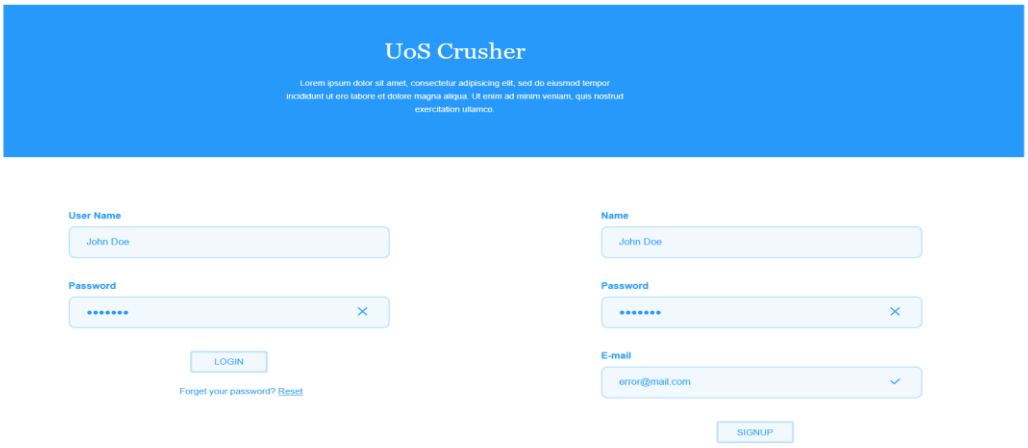
In the login page, a user will be able to:

1. Register with university email. After sign-up, they will be redirected to their initial profile page and asked to provide necessary information to complete the sign-up. Once finished they will then be redirected to their profile page.
2. Retrieve password with their email by clicking “forget password?”. In the password reset page a user then will be able to update his password.
3. Login with correct password. For existing users, they will be redirected to their profile page.

Password is hashed using Bcrypt and database only stores the hashes in case of possible database compromises. Following a successful login, session data will be stored on the server.

Proper error handling is also included. Users will be notified by specific error handling pages, should their login or registration fail, or 404, 500 error be encountered.

The mock-up of sign-up and login page is the following:



As described by this mock-up, user can either login or sign-up. Both action can be done in the same page with their username or email.

User Dashboard

In the user dashboard page, a user will be able to:

1. **View** all his information including avatar, degree info, DOB, Email, his skills, chosen units etc. All information should be provided by user. Refer to the “system data model” for detailed fields. Note that we don’t have permission from university to validate the correctness.

2. **Update** the above information provided by user with “update” button. Some fields should not be modified logically such as year entered or nationality but most of fields are alterable.

3. View his previous given comments (both to unit and lecture) and lecture rating.

4. **Modify or delete** previous given comments and lecture rating.

5. Detailed features such as adding or deleting a unit of study cannot be completely covered in this proposal but they’re intuitive, plus the name “dashboard” should be self-explained. So, for anything that should be in a dashboard, we’ll implement it at a basic level.

User Dashboard

The mock-up of user dashboard is the following

The left column contains user avatar and his basic information. A user can edit any of it in place. The right-side column is the user review history including which unit was reviewed or commented by him and detailed review data. Also, a user can choose to edit or delete these review and comment right in place. The top bar includes redirection like homepage that might be useful for users.

Search

Following a successfully login, users will be presented with a main landing page where they can search for review pages of a unit or a lecturer. All results related to the user search inputs will be

displayed on the same page, if a fussy search is performed. The search outcome will be displayed in a separate page containing a list of hyperlinks to the destined pages.

6.2 Unit Review (Kun Zhang)

One of the core functions of our web application is to let users browse and writer reviews about IT-related units at University of Sydney. The main features of the component are presented as follows:

1. Show key static information of a unit.

The key information is excerpted from CUSP, including short description, lecturer(s), assumed knowledge, etc. We found that CUSP unit description pages might contain too much information, making it difficult for users, especially potential students, to locate the what they need. So instead, we exclude some not-so-relevant data, like attributes and objectives, for the purpose of clarity and simplicity.

2. Show unit reviews.

Reviews of a unit will be displayed orderly. Each review component consists of ratings and a description, buttons allowing the reviewer to modify or delete the review, as well as displaying number of upvotes and a button to upvote.

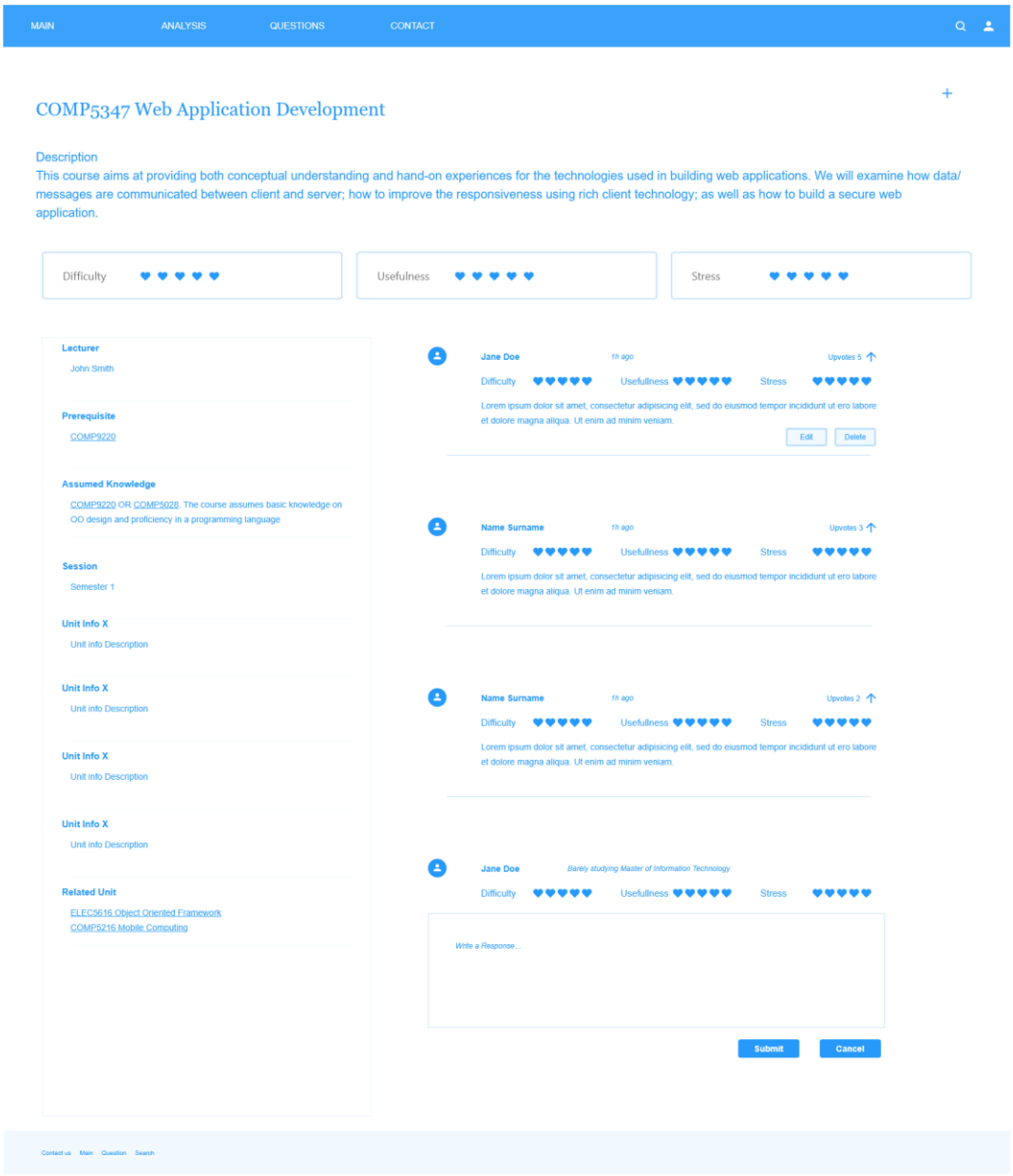
3. Allow users comment on as well as rate the unit on a scale of 1 to 5, with respect to the following:

- Difficulty: how hard is it to get satisfactory assessment outcomes.
- Stress: measures the workload of the unit with a rating 5 being very stressful.
- Practicality: describes how well a unit prepares the reviewer for job hunting.
  For instance, suppose COMP5347 were difficult, not stressful and very useful, a user would rate it 5, 3 and 5 respectively.

4. Allow users to edit and delete their reviews.

On the page, users can only modify and delete their own reviews. In other words, “Edit” and “Delete” button of a review will not be visible to the current user, if the review was made by other users.

Features described above are visualized in the mock-up as follows.


MAIN
ANALYSIS
QUESTIONS
CONTACT
COMP5347 Web Application Development
Description
This course aims at providing both conceptual understanding and hand-on experiences for the technologies used in building web applications. We will examine how data/ messages are communicated between client and server; how to improve the responsiveness using rich client technology; as well as how to build a secure web application.
Difficulty
Usefulness
Stress
Lecturer
John Smith
Prerequisite
COMP9220
Assumed Knowledge
COMP9220 OR COMP5028. The course assumes basic knowledge on OO design and proficiency in a programming language
Session
Semester 1
Unit Info X
Unit info Description
Unit Info X
Unit info Description
Unit Info X
Unit info Description
Unit Info X
Unit info Description
Related Unit
ELEC5616 Object Oriented Framework
COMP9216 Mobile Computing
Jane Doe
fh ago
Upvotes 5
Difficulty
Usefullness
Stress
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut ero labore et dolore magna aliqua. Ut enim ad minim veniam.
Edit
Delete
Name Surname
fh ago
Upvotes 3
Difficulty
Usefullness
Stress
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut ero labore et dolore magna aliqua. Ut enim ad minim veniam.
Name Surname
fh ago
Upvotes 2
Difficulty
Usefullness
Stress
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut ero labore et dolore magna aliqua. Ut enim ad minim veniam.
Jane Doe
Barely studying Master of Information Technology
Difficulty
Usefullness
Stress
Write a Response...
Submit
Cancel
Contact us
Main
Question
Search

6.3 Lecturer Review (Rufeng Tu)

User story

The user story describes a set of user-oriented scenarios:

- I as a student who is interested in a certain lecturer, want to know how the lecturer is rated by others, so that I could avoid irresponsible lecturer and choose the one that suits me best.
- I as a student who took a class from a lecturer, would like to give rating and some comments to the lecturer, so that I can help other students to know about this lecturer.
- I as a student who already submitted a review, has changed my mind, would like to delete or edit the rating and/or comments of my own to the lecturer, so that I can keep the review updated.
- I as a student who think some review is correct, has changed my mind, would like to delete or edit the rating and/or comments of my own to the lecturer, so that I can keep the review updated.

In light of these, the lecture review section consists the following features:

1. Display Lecturer Information and reviews

The lecturers in SIT could be displayed when user viewing the info page of the selected lecturer. The information of the lecturers includes:

- Avatar
- Lecturer's Name
- Nationality
- Teaching History
- Email
- Overall Teaching Ability
- Overall Responsiveness
- Reviews on the Lecturer
- Upvotes of the lecturer

Among which, the Overall Teaching Ability and Overall Responsiveness are calculated from the reviews by the user to the lecturer. The Reviews on the Lecturer is collected from the Reviews submitted by other students.

2. Submitting Reviews on the Lecturer

The user in SIT who knows the lecturer could submit reviews about this lecture. The reviews include:

- Year of the Review
- Teaching Ability Rating
- Responsiveness Rating
- Comments

Among which, the Teaching Ability and Responsiveness will contribute to the overall rating of this lecturer. And please be noticed that the year of review is not the same as the time of submission.

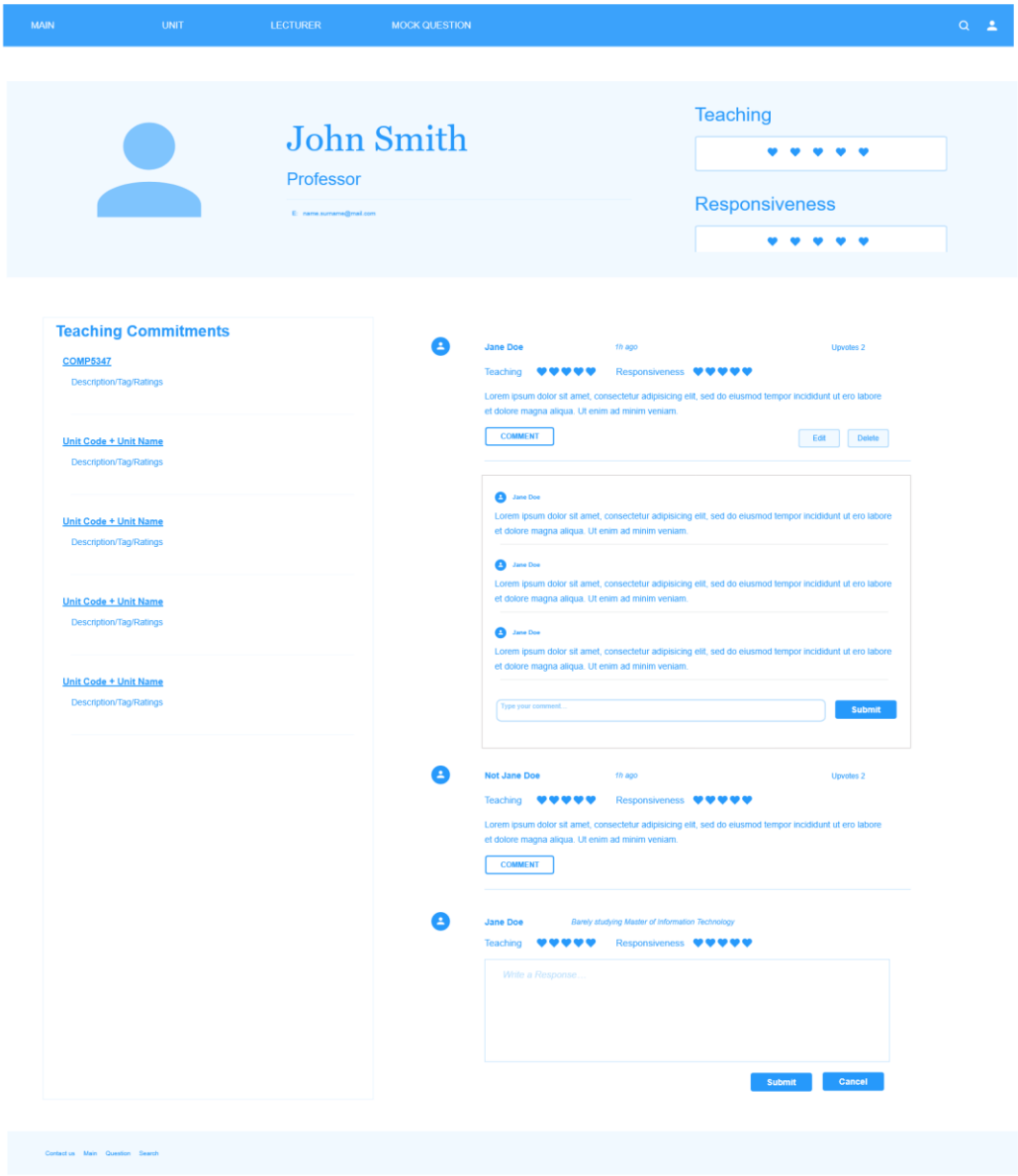
3. Edit the Reviews on the Lecturer

Review editing will have the following functionalities:

- Only the review submitted by user could be deleted or edit by the user.

- For editing, the time of new submission time will replace the old one.

The abovementioned features are shown in the mock-up below.



6.4 Sample Question (Ziqi Wang)

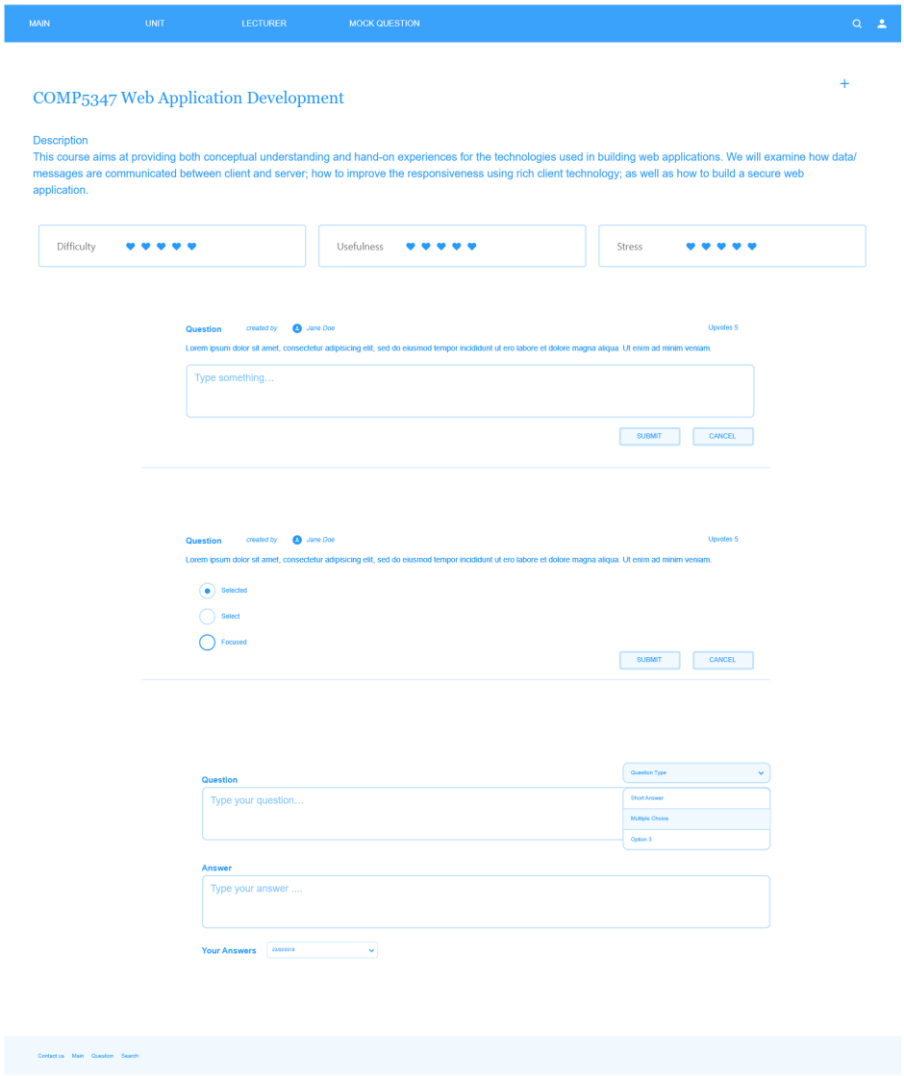
The sample question component features the following:

1. Reviewing question creation

When users are learning a course, he can create question that he thinks is useful for future use. Either he thinks it will helps him to understand the content of the course or the question may occur in a future exam, he can use our website to create a list of questions and save them to the database. The user can check out a list of questions based on either unit code or creators of the questions. Given the user being the creator of a question, he can also modify and delete the question.

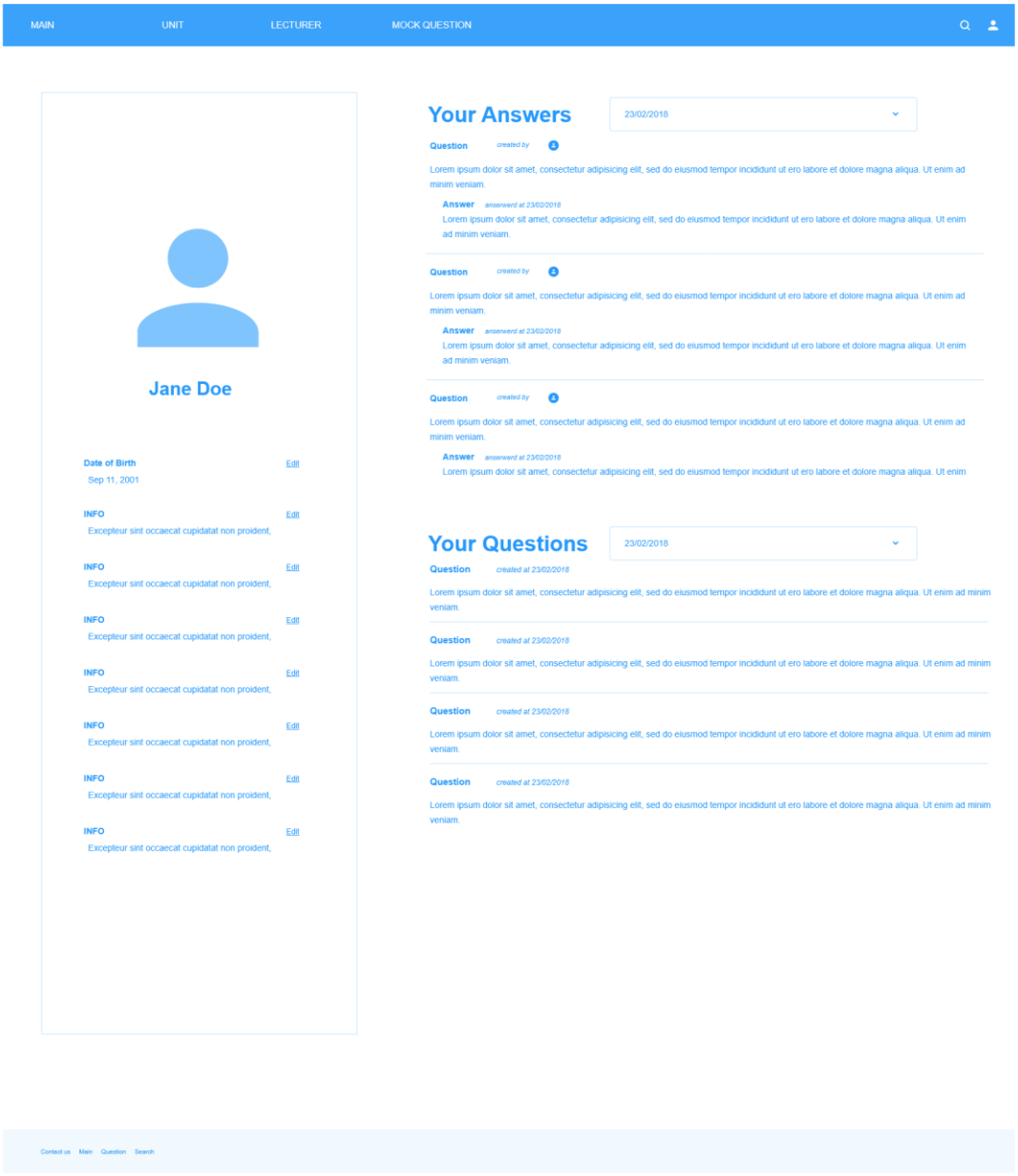
2. Taking quiz

As mentioned before, users are allowed to create questions for a course. Also, they can take quiz consisting of the questions created by themselves or from our public database. You can compare your answer to the sample answer as benchmark. Users can give thumb-ups and thumb-downs to the sample questions on a per-question per-user basis.



3. Viewing your questions

After a user create his own questions. He is allowed to view the list of the questions he created. Also, he is allowed to change the detail of the questions and the sample answers.



7. Future Works

7.1 User Interface Design

The user interface design of the project primarily uses Vanilla Bootstrap with few additional CSS and JavaScript libraries. Navigation of the application is functional but not optimized. For the betterment of user experience, more advanced UI design and implementation are needed for the future. A customized template using Bootstrap and jQuery would be easily applied to our project. The involvement of a UI Designer into the project is highly suggested.

7.2 Front-end Framework

It had been discussed during the initiation phase of the project that a Front-end framework, such as React or Angular could have been useful to make the UI more structured and modularized. Further investigation revealed that React might not have been optimal for Java EE Stack. For simplicity purpose, no front-end framework had been utilized. Modularization of our Views was achieved by Thymeleaf (HTML and Thymeleaf Syntax) at the stage. If the project grew more complex in the future, it is suggested that Angular or Vue be used.

7.3 CUSP Data Scrapper

As CUSP currently does not provide an API to the public, we simply input data from the CUSP website into our database. A possible solution would be to implement a data scrapper running on a separate process/thread on the webserver that automatically pulls down data and store them in our database. Alternatively, the data scrapper can be designed as an independent application that provides RESTful APIs to be used by our CUSP Crusher application.

7.4 Sample Question Features

Possible future works in the sample question component would be:

1. **Question discussion board.**

   For each question in public database, there will be a discussion board for it. People can publish their opinions in this area and give thumbs up to the sample answer, and one user can only give one from either thumb up. And future users can take the thumb up number as a reference to see whether the sample answer is good enough.

2. **Display comments by the thumbs up number order.**

   The comments for a question will be listed under the question by the order of thumb up number.

3. **Display comments by the time order.**

   The comments for a question will be listed under the question by the order of create time.

8. Conclusion

In this proposal, the demands and feasibility of our project CUSP Crusher were discussed and the schedule, implementation plans, and component distribution were elaborated. Some proper quality management schemes were applied in the illustration of mock-up, data model and system architecture.

To conclude, this proposed project is unique and feasible. The deliverables could help to achieve our aims. It could be suggested that our group (Group 24) should continue this project with the proposed plans.

9. Reference

[1] https://studentvip.com.au/usyd/subjects