CS355 FINAL ASSIGNMENT

EDUCATION

Sami Bedrani & Patrick Brennan
17363121 & 17473946

Table of Contents

Eduline: A remote Education software

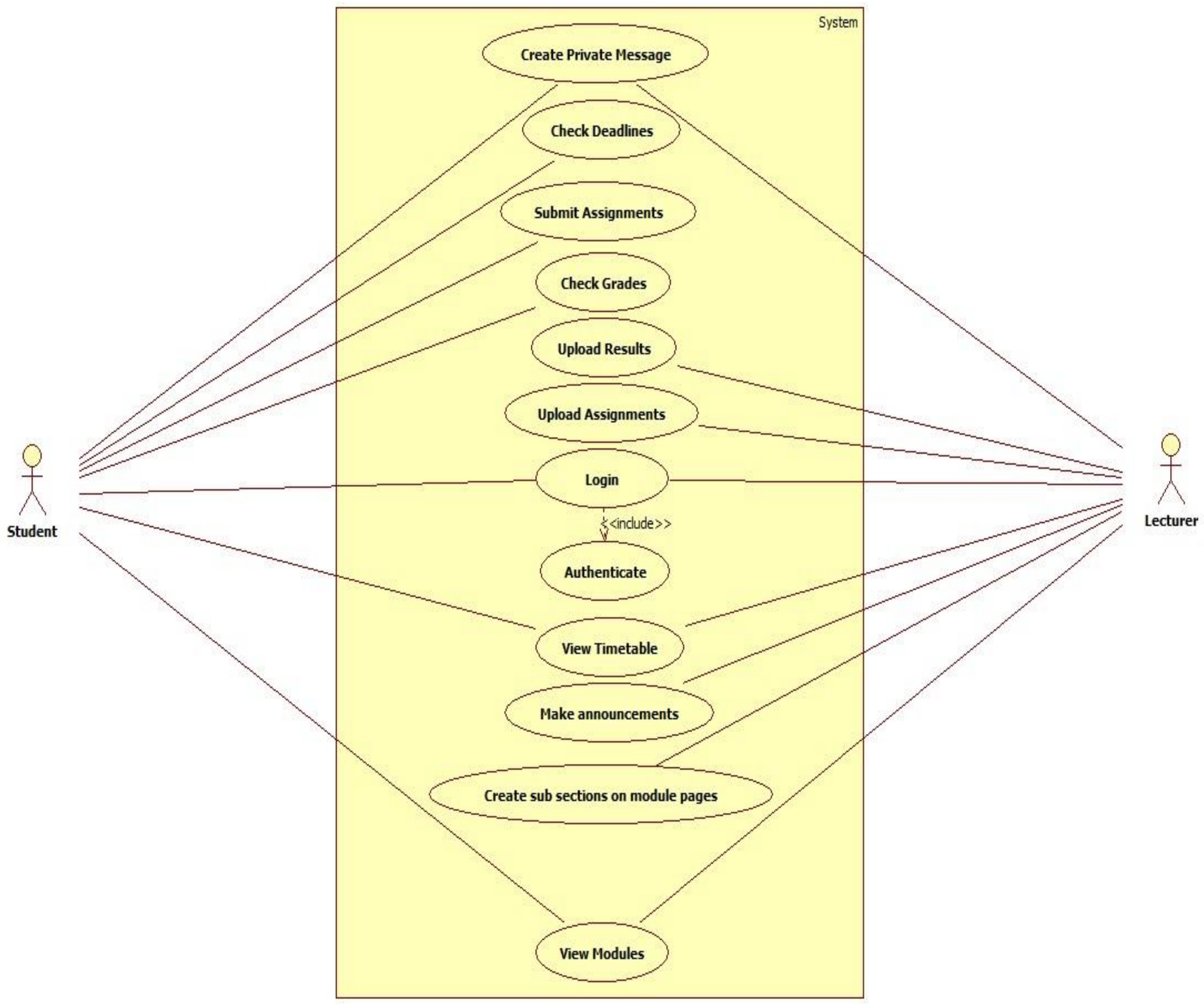
Due to the current events we decided upon the education sector as we now have personal experience with online education allowing us to better identify areas of potential improvement in this sector.

User Stories

1. As a student, I want a 24/7 server uptime so that I can access my course material easily from different time zones
2. As a student, I want to be able to upload files so that I can submit my assignments no matter the file type.
3. As a lecturer. I want to be able to upload a question sheet so that students can view their assignments on the module page.
4. As a lecturer, I want to be able to upload student grades privately so that I can leave individual comments and feedback for students.
5. As a student I want to be able to view all my current modules in the same place so that I can easily navigate through my course.
6. As a student, I want to be able to view all my upcoming deadlines in a chronological list so that I can better manage my workload.
7. As a student, I want to be able to view my course timetable so that I can better plan my academic day.
8. As a student, I want to view my continuous assessment grades on each module course page so that I can track my progress.
9. As a student I want to be able to privately message lecturers so that I can interact with lecturers confidentially.
10. As a student, I want to be able to view course work from modules I've previously taken in order to revise previous topics.

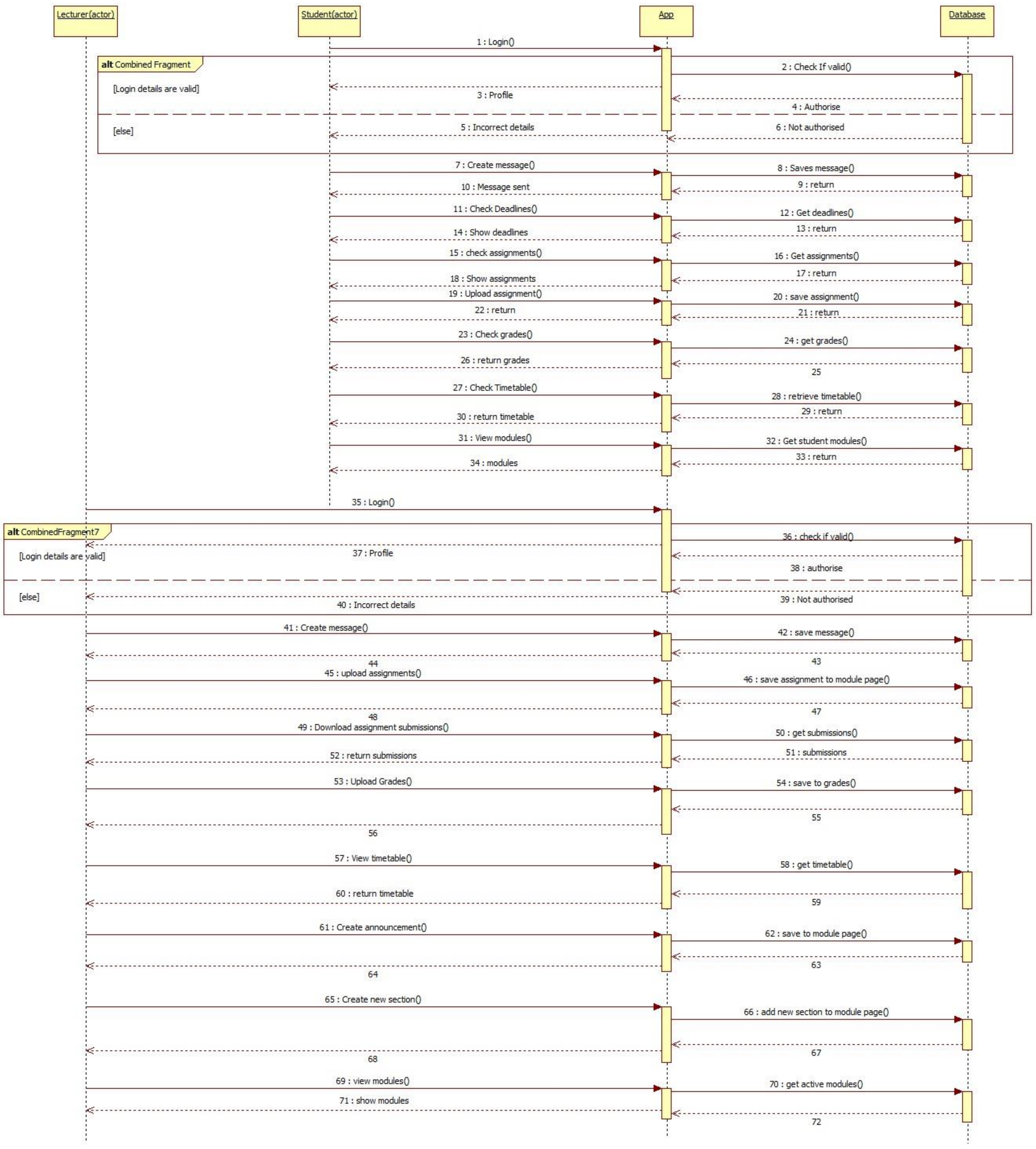
Use Case Diagram

This UML diagram displays the use cases for our app. We have two actors, the student and the lecturer, then we have the system containing each use case. There are many use cases for either actor with a few use cases in common such as the ability to log in, create a private message, view modules and check the timetable. There are also unique use cases that apply to each actor. The student will have the ability to submit assignments, check deadlines and check their grades. The lecture will have the ability create sub sections on module pages, upload assignments, upload grades and make announcements.



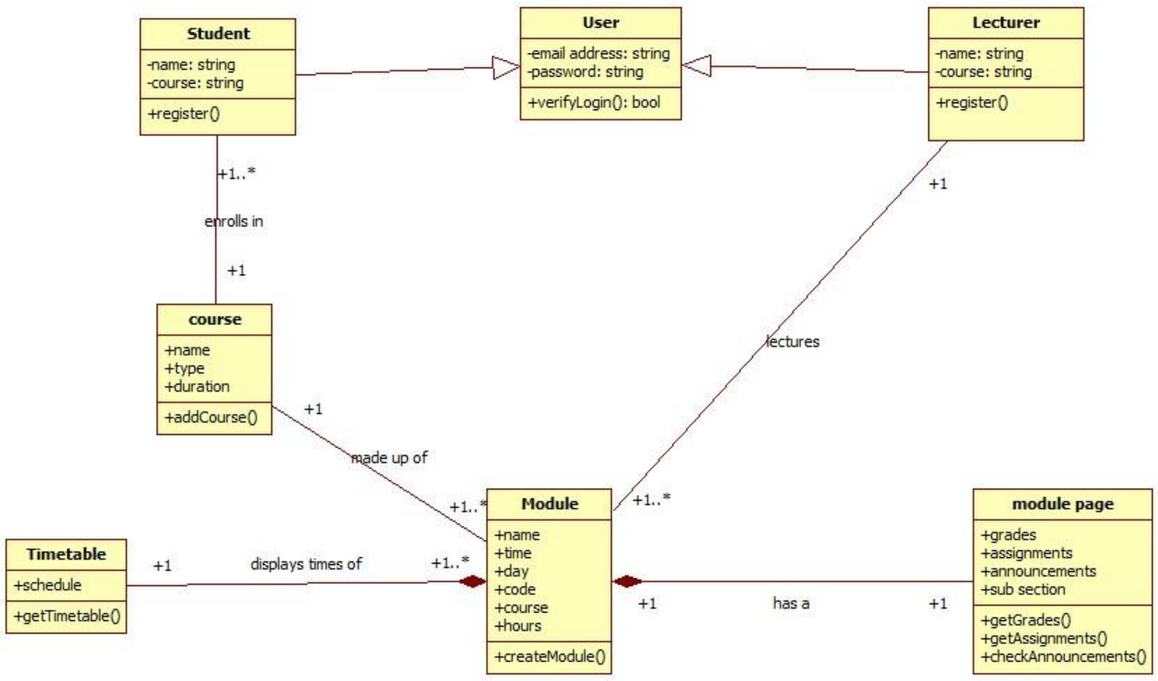
Sequence Diagram

This UML diagram shoes the order in which our ap would perform its functions using two actors, Student and lecturer, the app itself and the database will contain all the data.



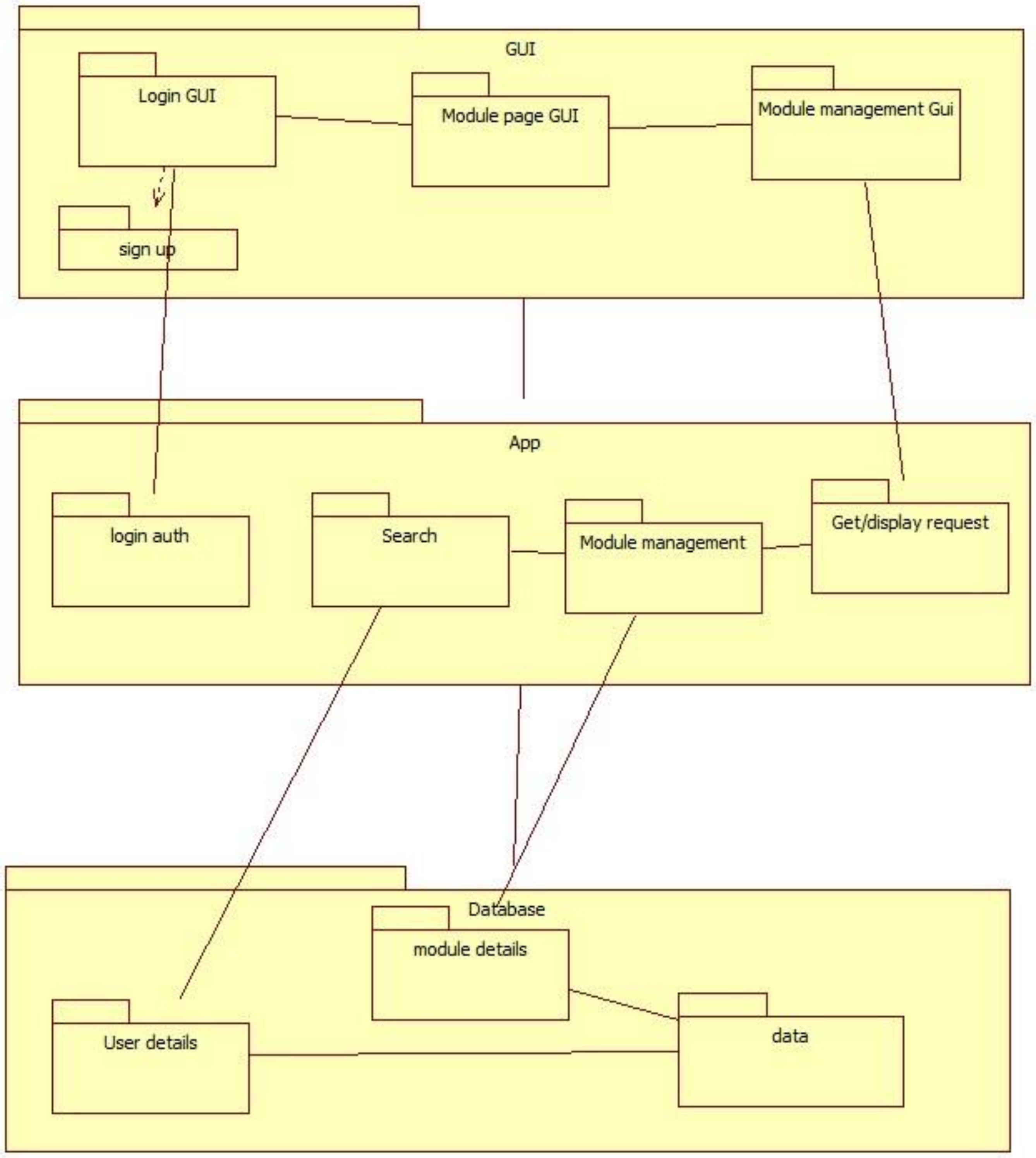
Class Diagram

This is the UML diagram depicting the potential classes that our app would consist of. With a User class being the parent so the Student and Lecturer classes that are used to create users, a course class that a student enrols in and is made up of modules, a modules class that a lecturer lectures in that also contains the details for each module and then the two classes, timetable and module pages contain the extra module information required.



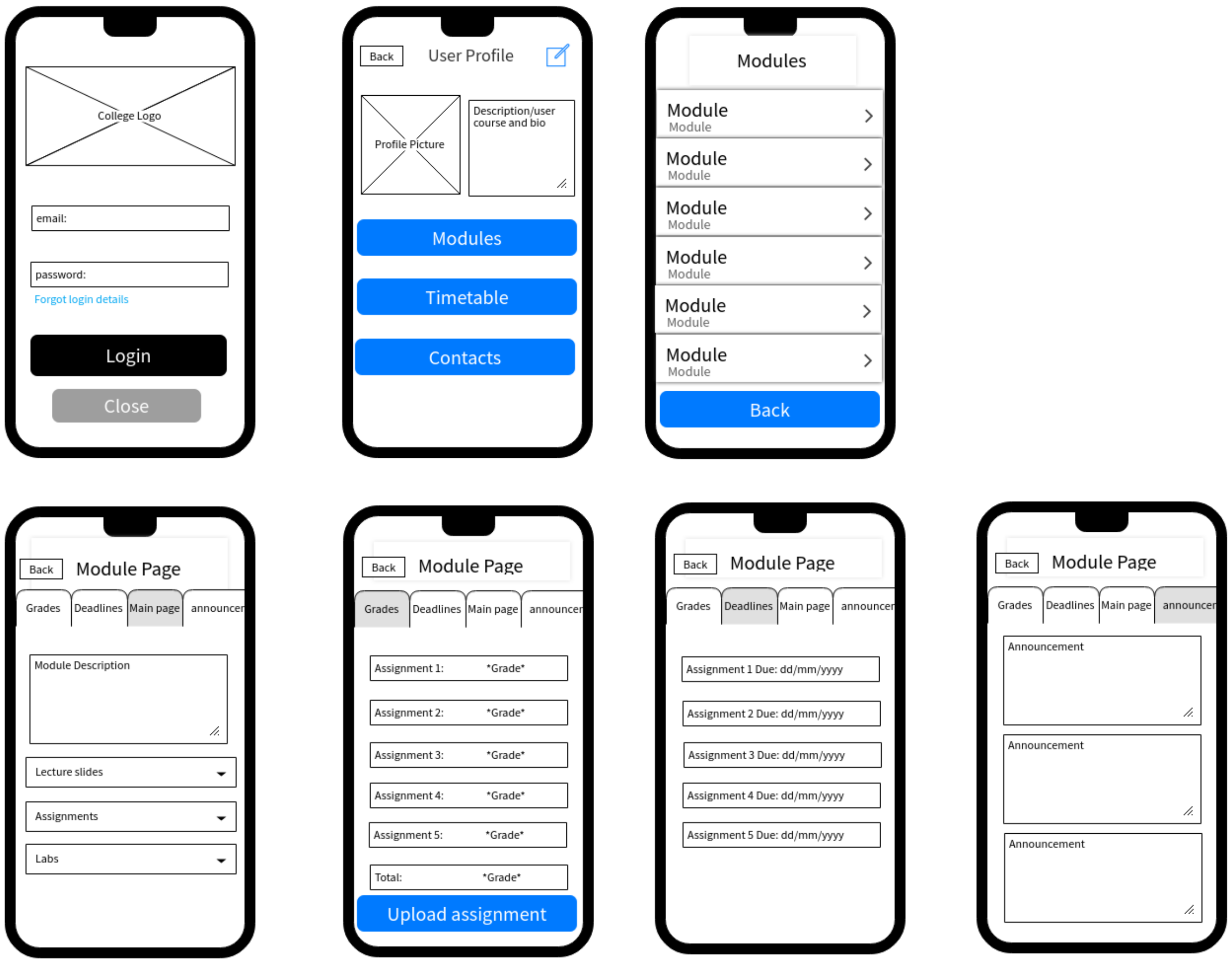
Architecture Diagram

This is the UML diagram depicting the breakdown of our apps architecture. Our app is broken down into GUI, app functions and the database that stores all the data.

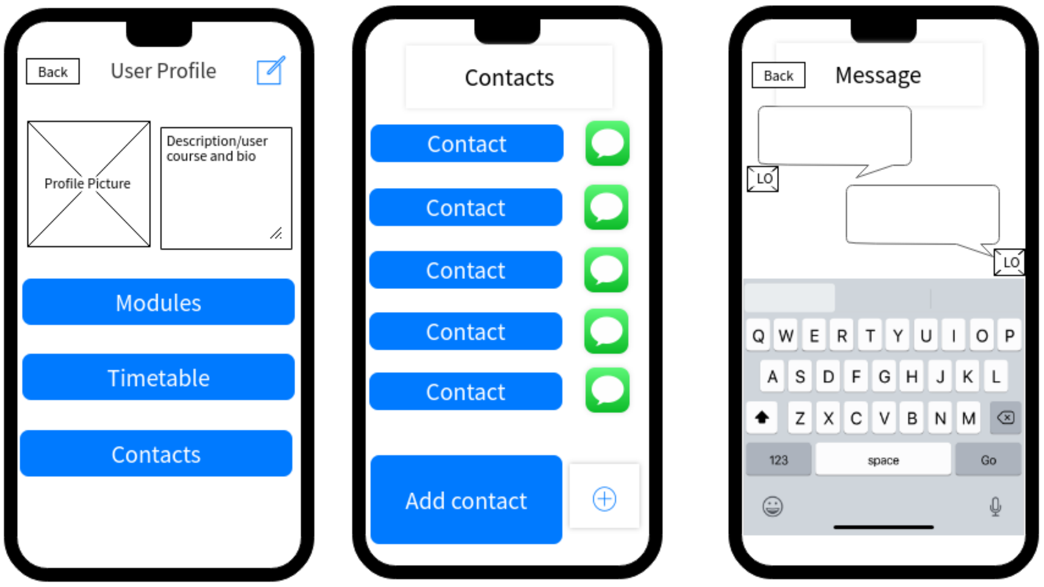


User Interface

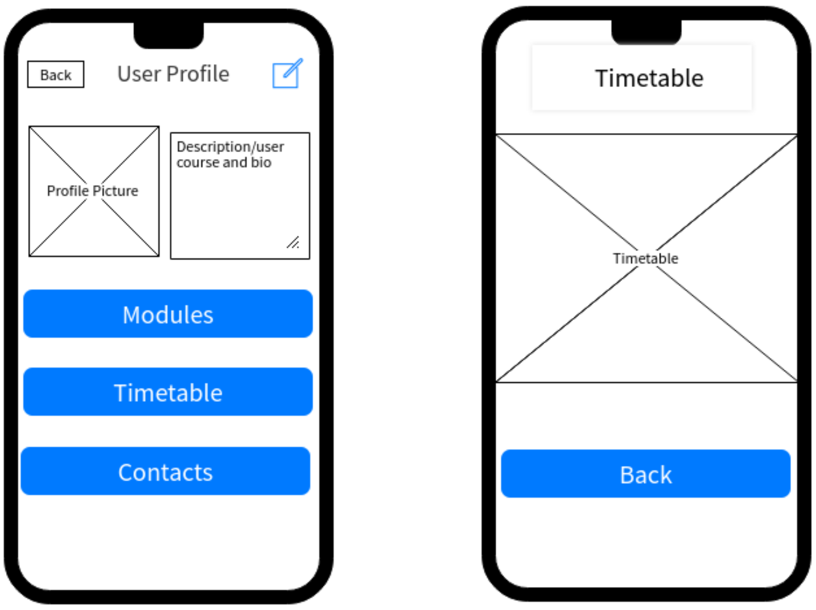
This UI mock-up shows the login screen for our app that is then followed by the user profile on successful login. The user can then select one of 3 options. In this section the user selects modules which brings them to a list of their active modules. The user can then tap into any of these modules to be brought to the main page for that module. Here a description would be displayed alongside lecture slides, assignments and labs. Pressing the tabs at the top of this page would then bring them to either grades, deadline or announcements tab where they can check up on their grades, upcoming deadlines and any lecturer announcements.



In this mock-up the user selects the contacts option. This brings them to a contacts screen where they will see a list of their contacts with an option to message them. There is also an add contact option. The last page shows how messaging would work if the user wanted to contact someone.



In this mock-up the user selects the timetable option with brings them to a page displaying their personalised timetable.



Software Testing

Before Testing
Before testing our software, we would first ensure that the code runs smoothly and remove and syntax or static semantic errors. If the program was created using java examples of checks we would perform would be ensuring all variables are declared and correctly typed, check variable scopes and whether methods are correctly.

Unit Testing
The first test we would carry out is a unit test. We would launch the application on a compatible device and check that every component of the user interface is working as intended. Some examples of a unit test for our application would be tapping on the module name on the modules menu to ensure that it brings us to the correct course or ….

Integration Testing
The next step would be to perform an integration test. We will use the System interface to test multiple units and ensure they interoperate correctly. This would be especially important in ensuring our messaging between lecturers and students is fully operational??

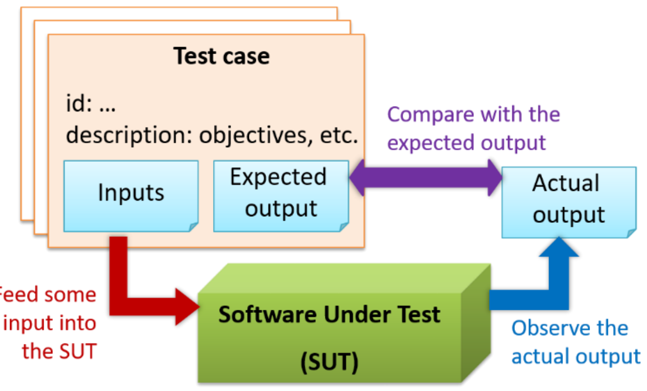
Acceptance Testing
When acceptance testing our application, we will refer to the user stories and ensure that every requirement has been met and that the software functions as intended. This validates the user requirements.

Regression Testing
After we've thoroughly tested our software using the processes above, we must regression test our software to ensure that no new bugs have been created after our software underwent changes related to bugs, we found using previous tests.

White Box Testing
We would only use white box testing when testing our software for a number of reasons.

1. As the development team is only made up of two, we would be able to get much higher coverage faster.
2. Also, because the two of us developed the app together using white box testing would make it very easy to identify the problem as we are already familiar with the code.
3. White box testing would also be helpful when ensuring that our database is retrieving the correct information.
4. Black box testing in my experience is more useful when dealing with inputting specific data rather than testing user interfaces.

To name a few.


Test case
id: …
description: objectives, etc.
Inputs
Expected output
Compare with the expected output
Actual output
Feed some input into the SUT
Software Under Test (SUT)
Observe the actual output

Testing Tools

Upon carrying our research, we decided to use TestNG and Selenium to test our application. We are both already familiar with these tools and know they can easily be integrated into Eclipse IDE.

After Testing

After we've finished testing, we would then analyse the time taken and budget allocated to this testing period. We would tally the money we spent on software testing and compare it with our software budget using Microsoft Excel. This would grant us valuable knowledge that can be used to better predict software testing expenditure in the future.

Contribution

Sami Bedrani

1. Software requirements / User Stories
2. Software testing
3. Explanatory texts

Patrick Brennan

1. UML Diagrams
2. UI Mock-up
3. Explanatory texts

We collaborated throughout the project asking each other's advice and opinions on our sections.