UNSW
AUSTRALIA

School of Computer Science and Engineering

The University of New South Wales

SENG2021 – Requirements and Design Workshop

Term 1 – 2019

Raisin

Date of Submission: 29/03/2019

Submitted by: Richard Liu (z5165455)

Rory Madden (z5208738)

Yorke Li (z5207298)

Faiz Ather (z5170340)

Group Info:

Name: digital_invention (formerly NoName3)

Mentor: Christopher Joy

Meeting Day: Tuesday

Meeting Time: 1:20PM – 1:40PM

CONTENTS

1 CONCEPT INTRODUCTION

This project aims to address the concerns of a typical UNSW student, who is enrolled in courses that use WebCMS3. Under UNSW's new trimester system, courses are accelerated and due dates can arrive quickly, and sometimes surprisingly. Consequently, little time is left available to prepare and students now instead are prompted to learn course content before knowledge of the course requirements, in order to keep up to the pace of the course content.

Therefore our project intends to create a simple web application for students to view a list of all their WebCMS3 due dates (assignments, labs, project milestones and exams, etc) ahead of time, with the option of being able to export them to their Google or Apple calendar/iCal for viewing later. Ultimately this aims to assist students in being prepared for their courses and provide a swift service so more time can be dedicated towards study.

2 SOFTWARE ARCHITECTURE: INTRODUCTION

Since our application will be website based, it will require a graphical user interface (front-end), and a data processing server (back-end). The necessary interactions between our application's front-end and back-end can be seen in Figure 2.1. Thus, the application will use a client-server architecture, which will allocate the workload accordingly through a centralised server as clients request services. The entire software architecture of our application can be seen in Figure 2.2.

Primarily our application is targeted towards students wanting to summarise their WebCMS3 course due dates ahead of time. Although many unique users would be served, the application provides a single-use service for each user: requiring no persistence or permanent storage of data. Thus data will be temporarily stored during a session, and once finished, it will be handed over to a respective calendar API and then deleted from our system.

Currently the web application is intended to be executed using the local computer address on a computer, hence the main hardware is targeted towards a computer that can use a Linux machine. However so long as the computer has access to the software Java and Python version 3.7+ then any system could use the application.

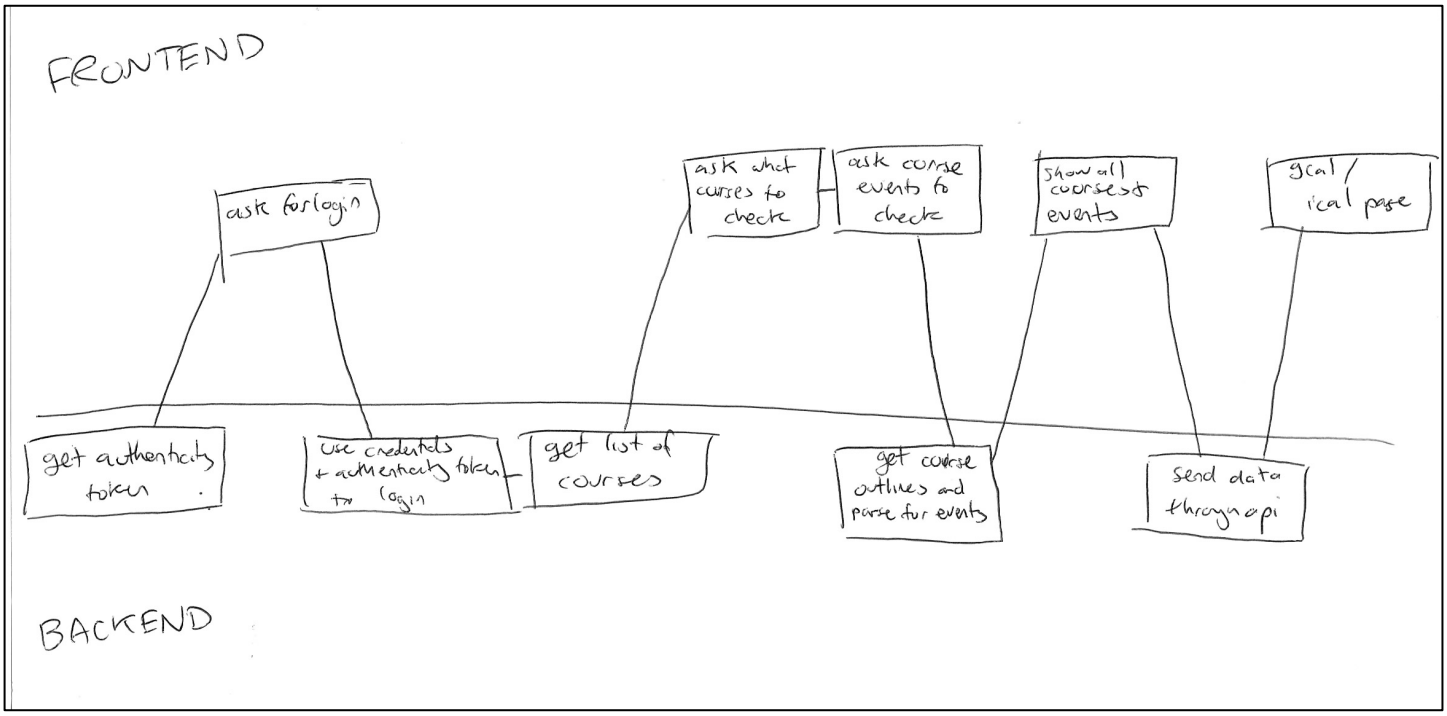

Figure 2.1: The front end & back end process of using the web application

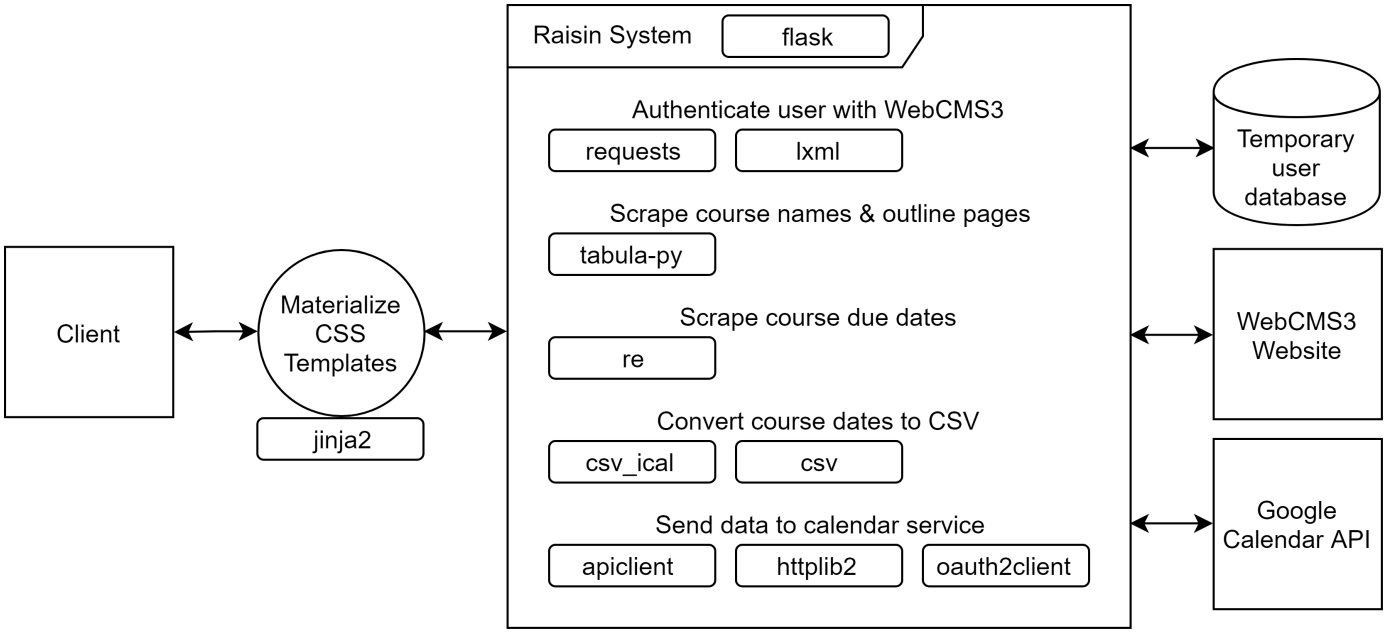

Figure 2.2: System Architecture Diagram of Raisin.

Legend:

Straight Edges = External Application/Entities

Rounded Edges = Python Modules

Circle = Web Interfaces/Templates

Cylinder = Database

3 SOFTWARE ARCHITECTURE: BACK-END

The server for our application will be written entirely in Python as its countless powerful packages and frameworks are suited to efficiently create our simple website model. Python also suits our group as we all are experienced with the language and its simplicity and easy readability facilitates a cohesive group working environment. Additionally, the classes that we will use to run our server will incorporate the structural software design pattern "*façade*", so our server maintains high cohesion and low coupling. These principles will allow the server to simply its subsystem and make it less dependent upon them. Specifically, for our server, it will streamline the process of using the objects that represent an assessment and simplifies the other features that use the object. Consequently, Python will be used to construct a back end for our application so the tools and data sources can be used effectively.

The main data sources our project will use are WebCMS3 and Google Calendar's API:

- The application uses "web scraping" to obtain a student's current courses, as well as their respective course outline documents, which are found on WebCMS3. Accessing the course outlines requires a student's zID and password, which are sent directly to WebCMS3 and are not withheld by our application.
- In order to export a course's due dates into a calendar form, the application will require access to Google Calendar's API, where a student can log into their Google account and through the use of the Python "oauth2client" library, the calendar automatically requests & updates.
- Alternatively, if a student would prefer to use Apple's calendar, our application will then convert the current data into an ".csv" file then into an ".ics" file, which can then be manually added to a calendar.

However, the process of extracting due dates from a course outline document is no trivial task and is essentially the core functionality of our application. This will require our group to develop a specialised web scraper & PDF scraper in order to extract the assessments and their due dates from a course's outline. The web scraper will use the Python "Beautiful Soup" library to extract data from webpages while the PDF scraper will use the Python "tabula-py" module. Overall intended process to achieve this is, follows:

(Bolded portions represent front-end interactions on Raisin's web browser)

1. Server scrapes the CSRF authenticity token from the WebCMS3 login form (this is necessary for login)
2. **User manually inputs their zID and password**
3. User's login details are passed to WebCMS3 for authentication, along with the CSRF authenticity token from before
4. WebCMS3's top navigation bar (which contains currently enrolled courses) is scraped for course names and URLs
5. **User is asked what courses and event types – assignments, labs, project milestones and exams – they want**
6. Course outline documents are scraped for each course selected
    - If the course outline is a PDF, the PDF scraper is initiated, extracting all tables from the PDF
7. Points of interest are extracted from the course outlines, namely table rows and bullet points. These are each searched for instances of "Assignment X" or "X due" or "X exam", etc.
8. If any of these terms are found, their immediate surroundings are searched for text such as "Week X" or "Exam Period", as well as a "% weighting". If found, these values are added to a dictionary
9. **Due dates for each course are formatted and presented to the user**

4 SOFTWARE ARCHITECTURE: FRONT-END

The graphical user interface will use Flask, Jinja2 and Materialize in order to provide a simplistic and easy-to-use experience.

Flask

Flask is a powerful, yet relatively straightforward web server, with enough features to cover all the needs of our application. Due to the limited timeframe, this was chosen as all members of our group are experienced with its use.

Jinja2

Jinja2 allows for seamless integration between a client and server in the form of modular HTML templates. These allow for data to be presented effectively and efficiently, in order to serve multiple clients at once. Again, this has also been chosen due to our group's prior experience.

Materialize

These templates are styled using Materialize CSS, allowing for simplicity and ease of use, all while maintaining a modern aesthetic. While no group members have had any prior experience with it, the documentation is simple enough to understand.

Through the use of the above libraries, our web application should achieve a high-quality user experience.

5 INITIAL SOFTWARE DESIGN

In order to fulfil the purpose of our application, the features that our application will need to provide have been deconstructed into user stories. The relationships between the entities within these user stories will be handled treated by our application as Figure 5.1 indicates, while realistically on the back-end, the connections between these entities will operate as viewed in Figure 5.2.

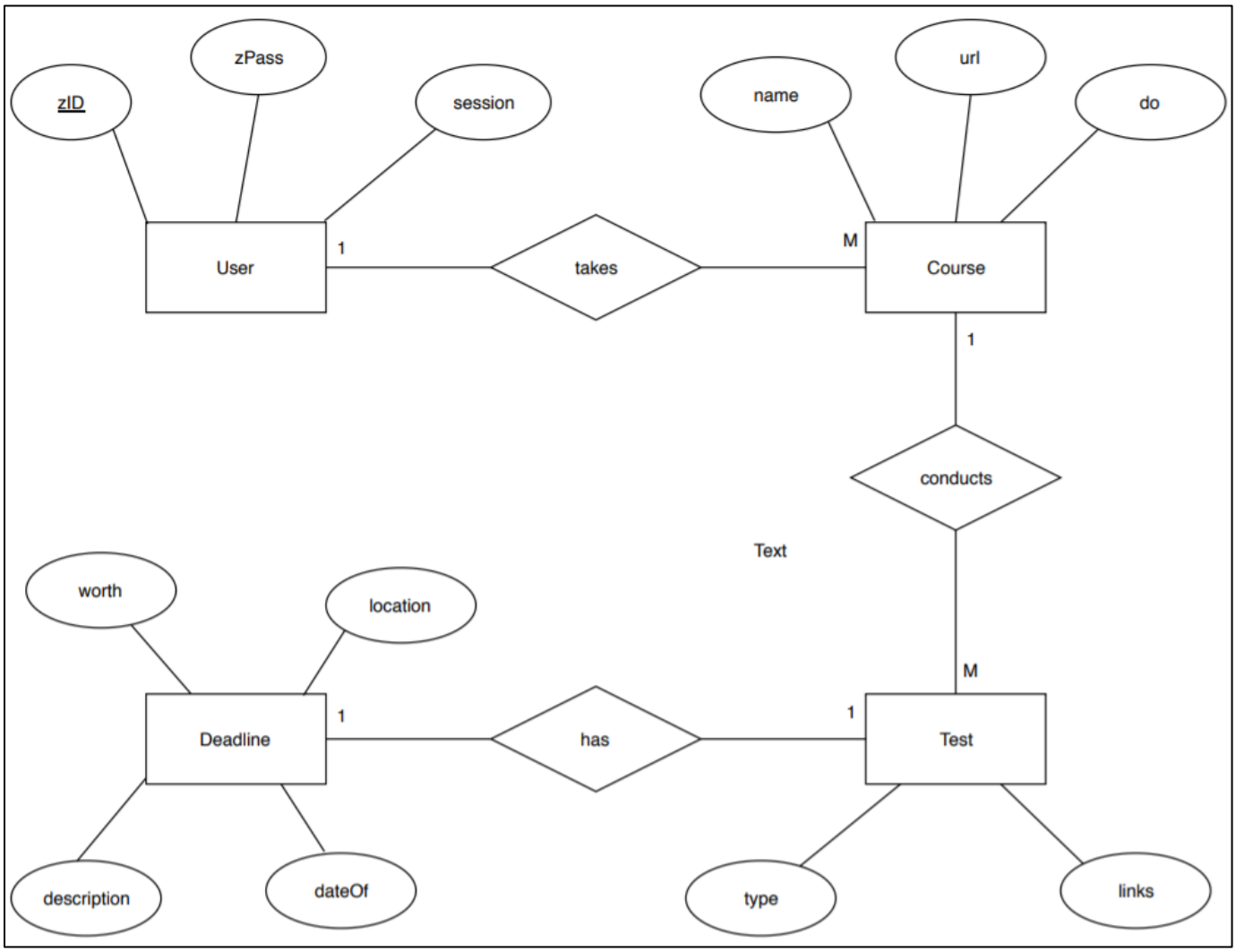

Figure 5.1: System Entity Relationship Diagram

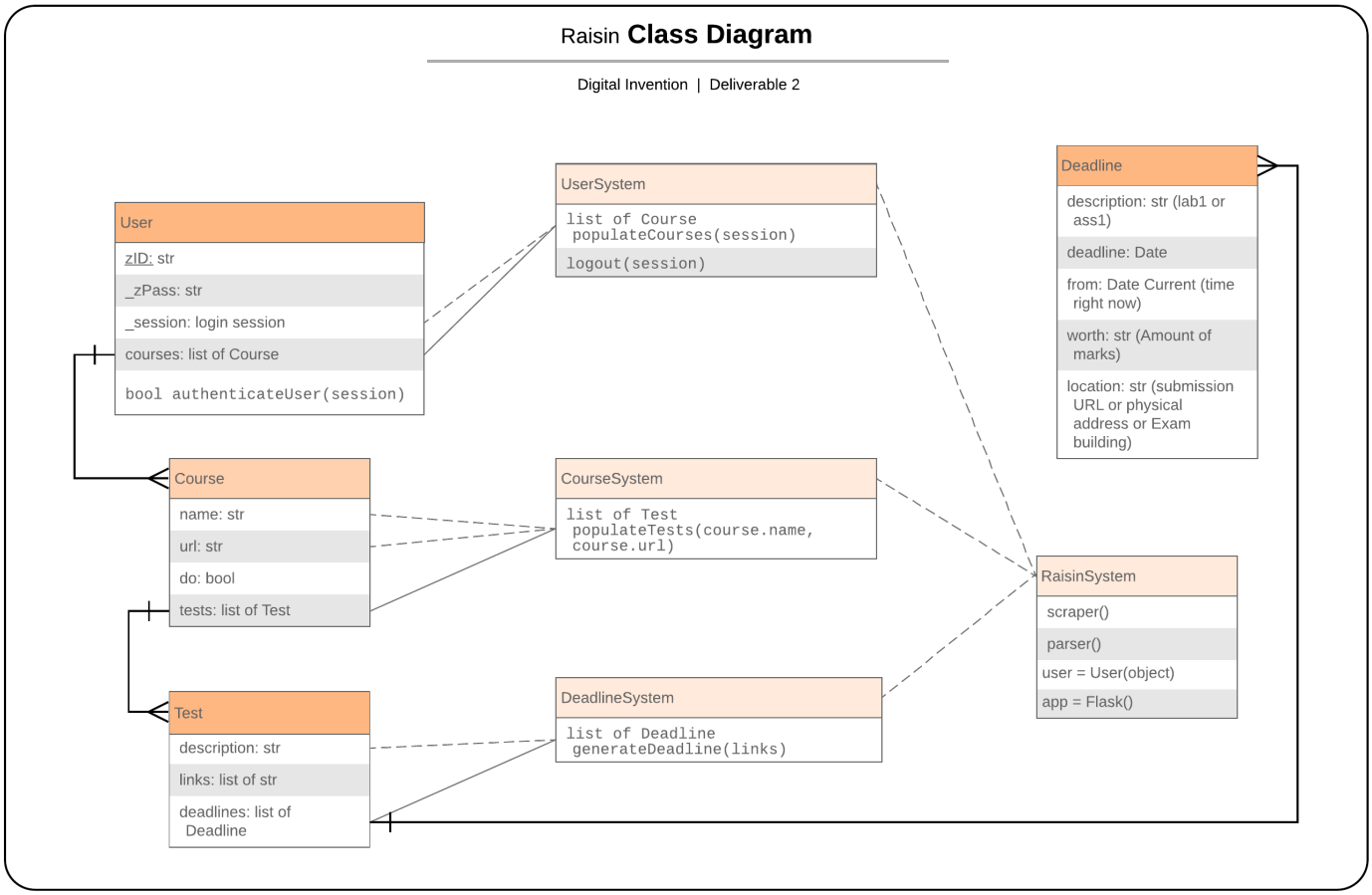

Figure 5.2: Class Diagram of Raisin's System

All of these features have been assessed and prioritised accordingly while also embracing SMART principles so that our project can be realistically visualised to achieve success. Hence the user stories that our application must implement are delineated below.

The figure below explains the priority scale for our project

Priority Level	**Description of feature**	**Urgency**
Priority 1	Essential for the application to function.	Must be completed first.
Priority 2	Improves the quality of the application's necessary functions.	Should be completed directly after all priority 1 tasks are completed.
Priority 3	Improves the quality of the application by providing more useful features.	Can be implemented if time permits, only if all priority 2 tasks are completed.

Figure 5.3: Feature Priority Table

Priority Level: 1
Feature
Feature: To login and access all currently enrolled courses **As A** UNSW WebCMS3 student **I Want To** be able to log into my WebCSM3 account **So That** I can use the web application; Raisin
Scenario
Scenario: I just launched the Raisin web application and would like to access its functions/features **GIVEN** I am on Raisin's Login Page **WHEN** I enter my WebCSM3 login details **AND** my account has been authenticated **THEN** I will be redirected to Raisin's features/functions; the Course Assessment Selection page
Estimated Time Required to Integrate Feature: 5 Hours

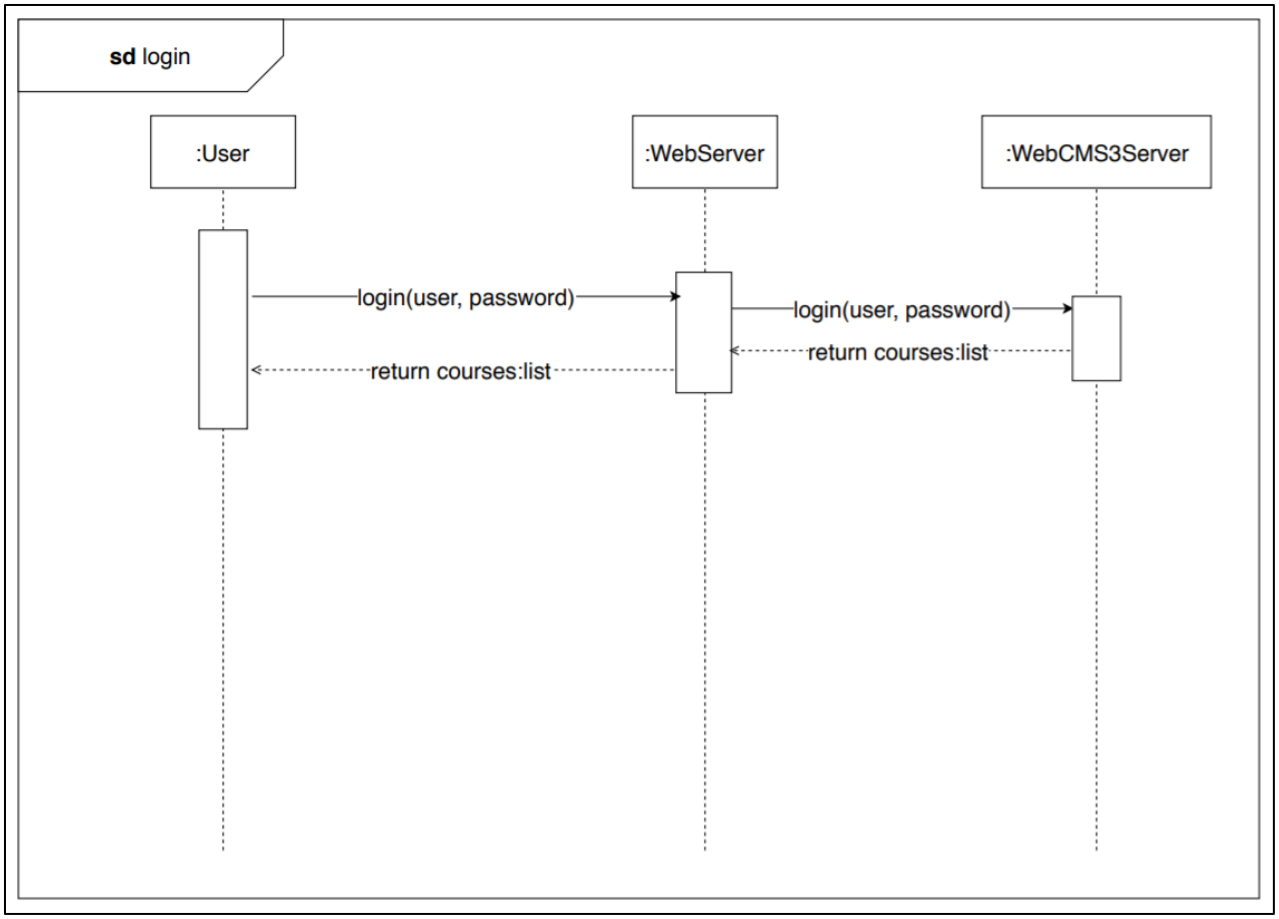

Figure 5.4: Sequence Diagram for Raisin's Login Process

Priority Level: 1
Feature
Feature: View all course commitments **As A** UNSW WebCMS3 student **I Want To** be able to view all my course assessments together on a single page **So That** I am prepared for my entire term
Scenario
Scenario: Course outlines have just been released and I want to find all the assessable tasks & when they are due. **GIVEN** I am on Raisin's Course Selection page **WHEN** I have checked all the boxes for my enrolled courses or clicked the "Select all" button **AND** clicked the "Next" button **THEN** I will be navigated to a Course Assessment Selection page **WHEN** I have checked all the boxes for the course assessments or clicked the "Select all" button **AND** clicked the "Next" button **THEN** I should be navigated to an Assessment Synopsis/"Due Dates" page displaying a summary of all my course assessments and their deadlines
Estimated Time Required to Integrate Feature: 3 Weeks

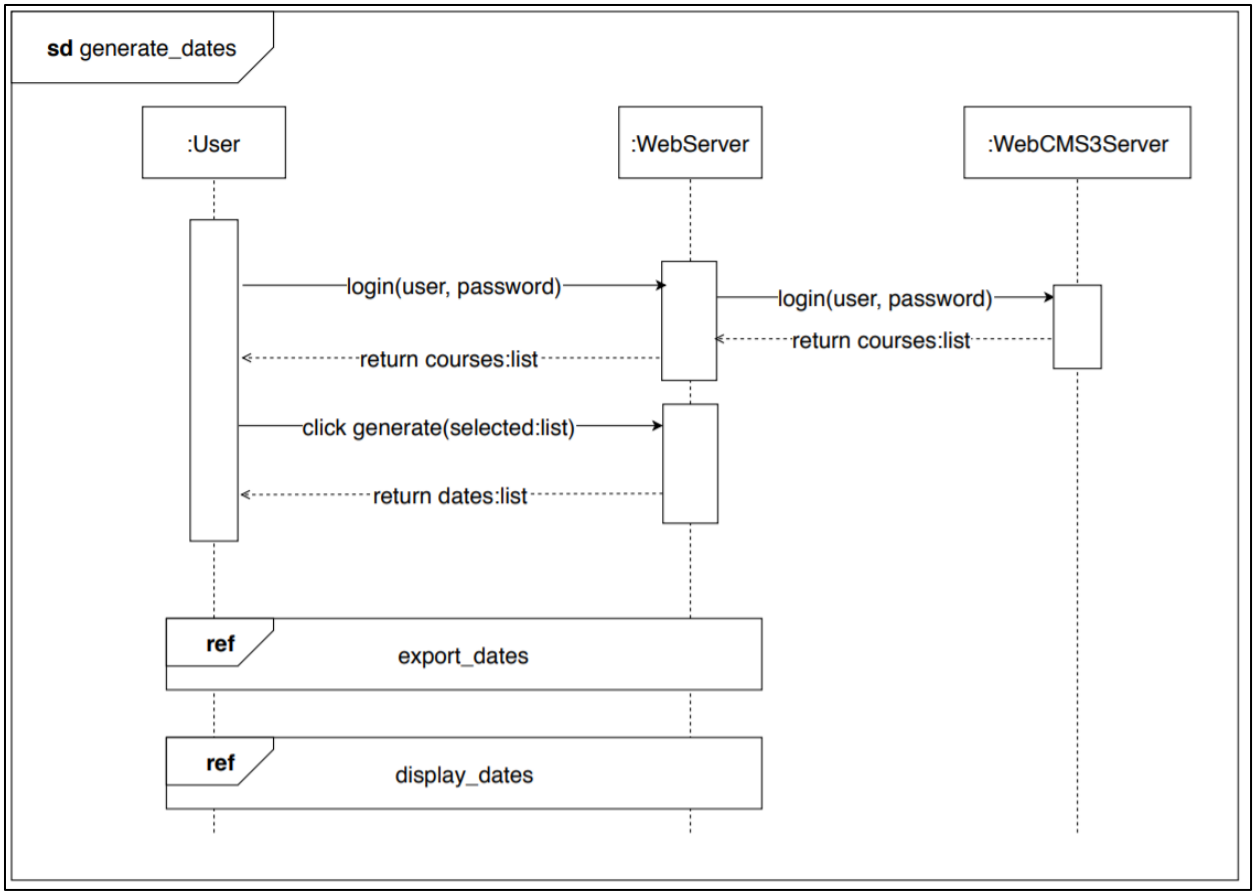

Figure 5.5: Sequence Diagram for Viewing Course Commitments

Priority Level: 2
Feature
Feature: All course assessments can be transferred onto a digital calendar **As A** UNSW WebCMS3 student **I Want To** be able to export all my course assessments to a digital calendar; Google Calendar **So That** I can complete my assessments on time
Scenario
Scenario: Raisin has just completed gathering all of the course assessments and I wish to export them into my Google Calendar **GIVEN** I am on Raisin's Assessment Synopsis/"Due Dates" page **WHEN** I click on the "Google Calendar" **AND** I log into my Google account **THEN** I will be asked to allow permission **AND** a pop up will appear verifying that the assessments have successfully been imported into my Google Calendar **WHEN** I check my Google Calendar, the assessments will be there
Estimated Time Required to Integrate Feature: 1 Week

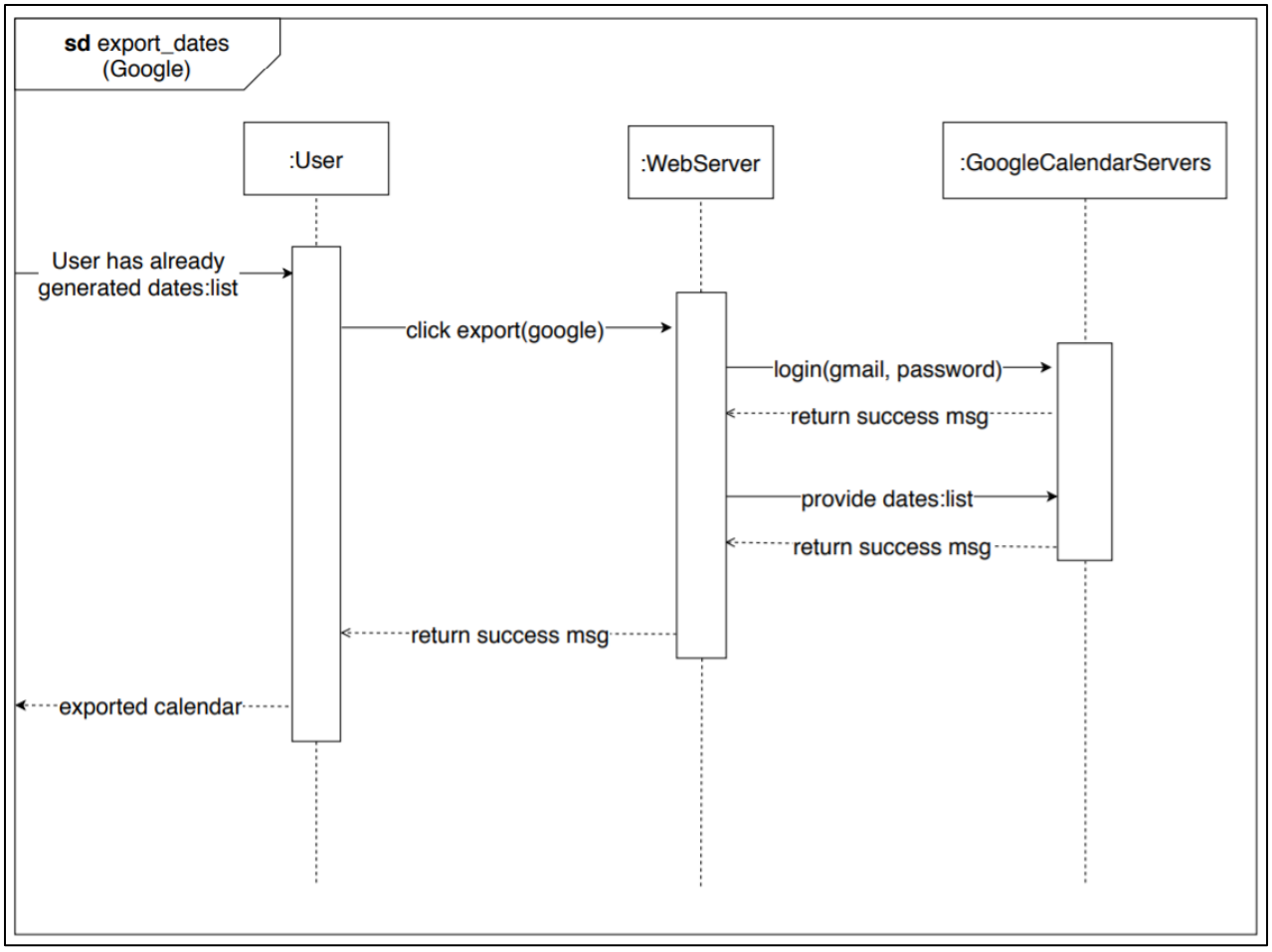

Figure 5.6: Sequence Diagram for Exporting to Google Calendar

Priority Level: 2
Feature
Feature: All course assessments can be transferred onto a digital calendar **As A** UNSW WebCMS3 student **I Want To** be able to export all my course assessments to a digital calendar; Apple's Calendar application/iCal **So That** I can complete my assessments on time
Scenario
Scenario: Raisin has just completed gathering all of the course assessments and I wish to export them into my iCalendar/Apple Calendar **GIVEN** I am on Raisin's Assessment Synopsis/"Due Dates" page **WHEN** I click on the "iCal" **AND** my course assessments have been compiled and converted into an ".ics" file **THEN** a downloadable file of my course assessments will appear in the "Downloads" of my web browser
Estimated Time Required to Integrate Feature: 1 Week

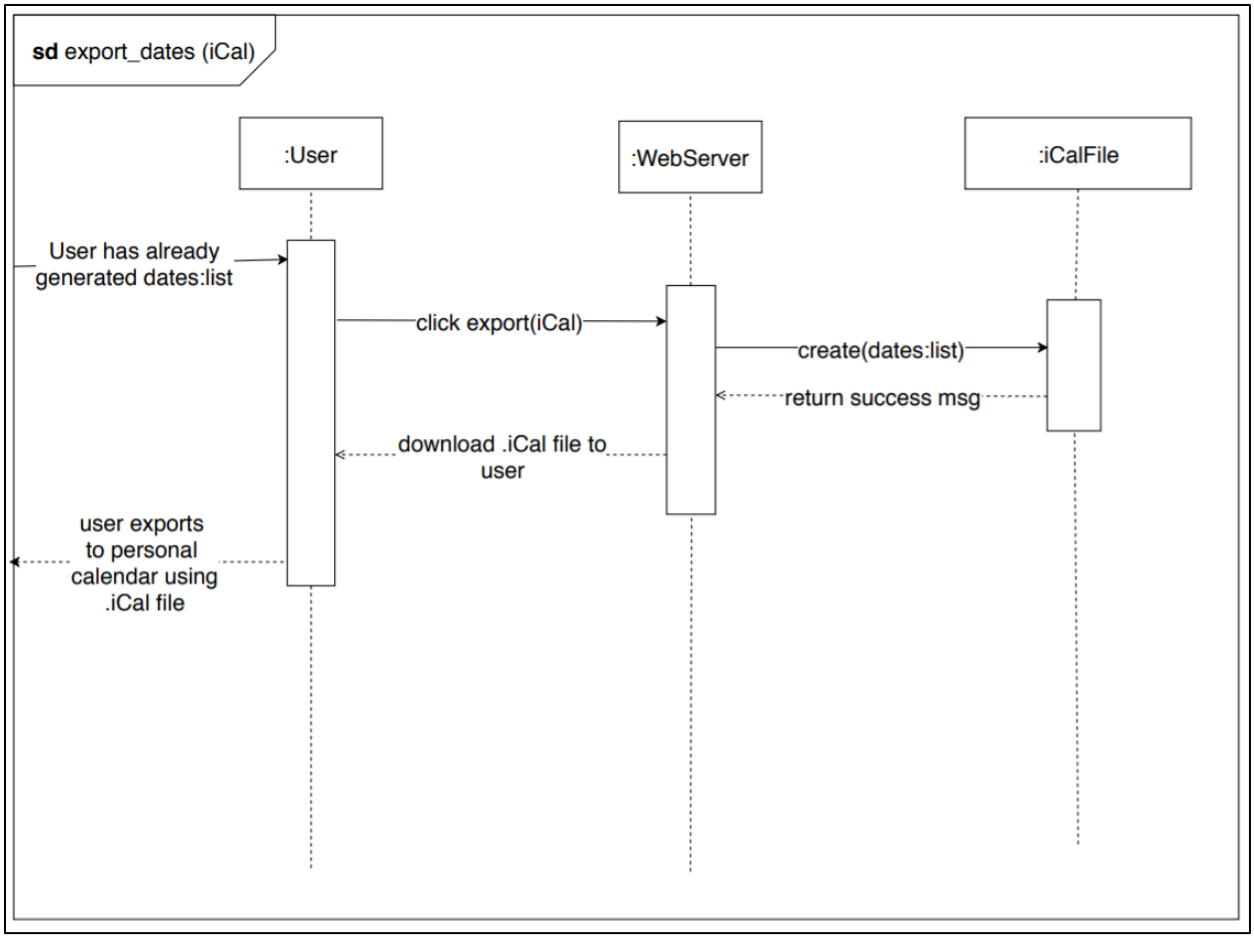

Figure 5.7: Sequence Diagram for Exporting to iCal

Priority Level: 3
Feature
Feature: To alternate how I can view the organisation of my course assessments **As A** UNSW WebCMS3 student **I Want To** have the ability to toggle how my course assessments are displayed **So That** I can organise my commitments each week or to each course accordingly
Scenario
Scenario: Raisin has just completed gathering all of the course assessments, and I want to see a summary of all my assessments tasks by course and/or by week **GIVEN** I am on Raisin's Assessment Synopsis/"Due Dates" page **WHEN** I click on a button called "Courses" **THEN** the course assessments should be arranged into separate sections for each individual course, which is organised by each week in these sections **WHEN** I click on a button called "Week" **THEN** all course assessments should be organised together into their respective weeks
Estimated Time Required to Integrate Feature: 1 Day

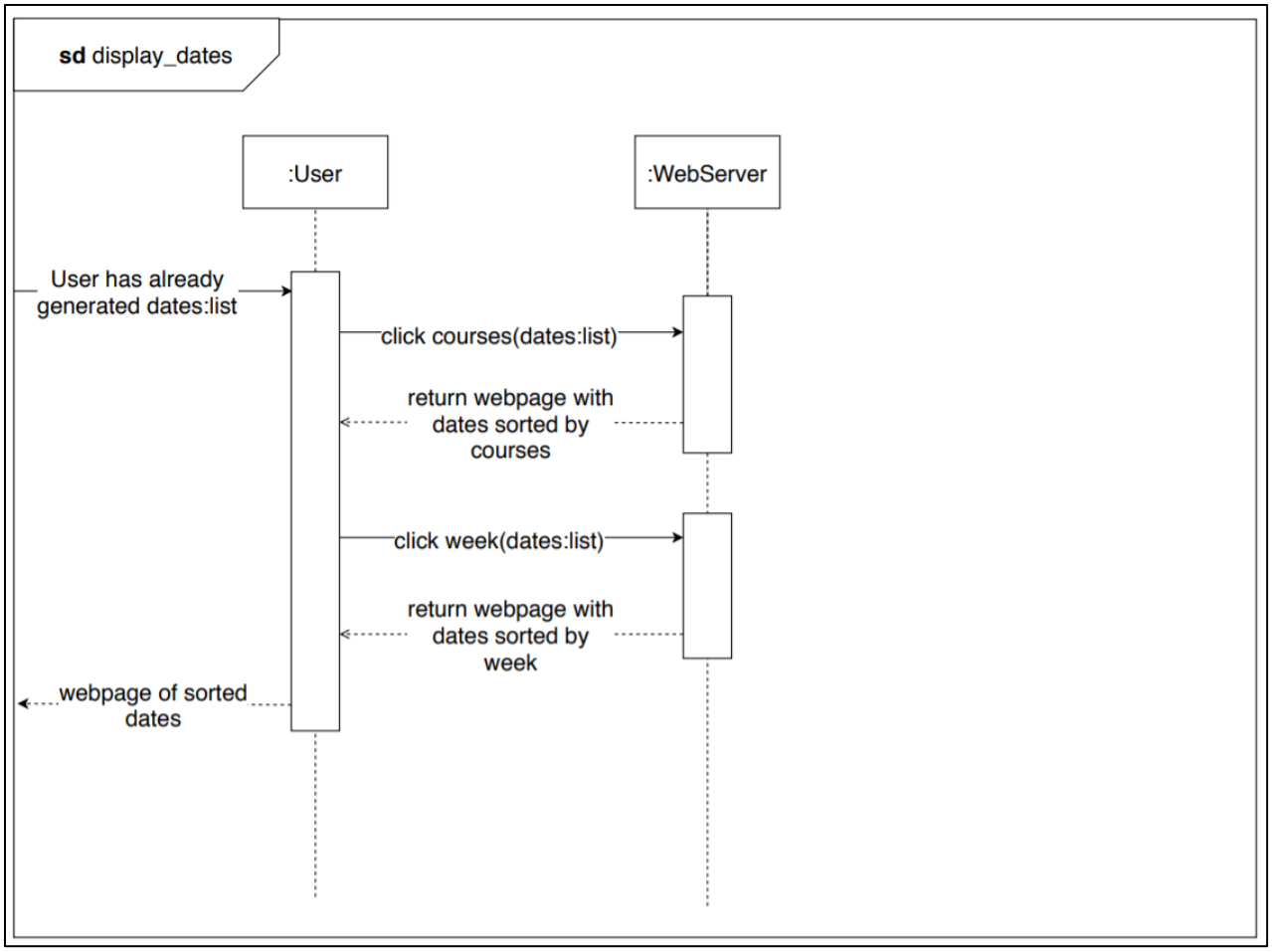

Figure 5.8: Sequence Diagram for Viewing Course Assessments Tasks by Group