

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 1

2 Software Architecture: Introduction 2

3 Software Architecture: Back-End 4

4 Software Architecture: Front-End 6

5 Initial Software Design 7

# Concept Introduction

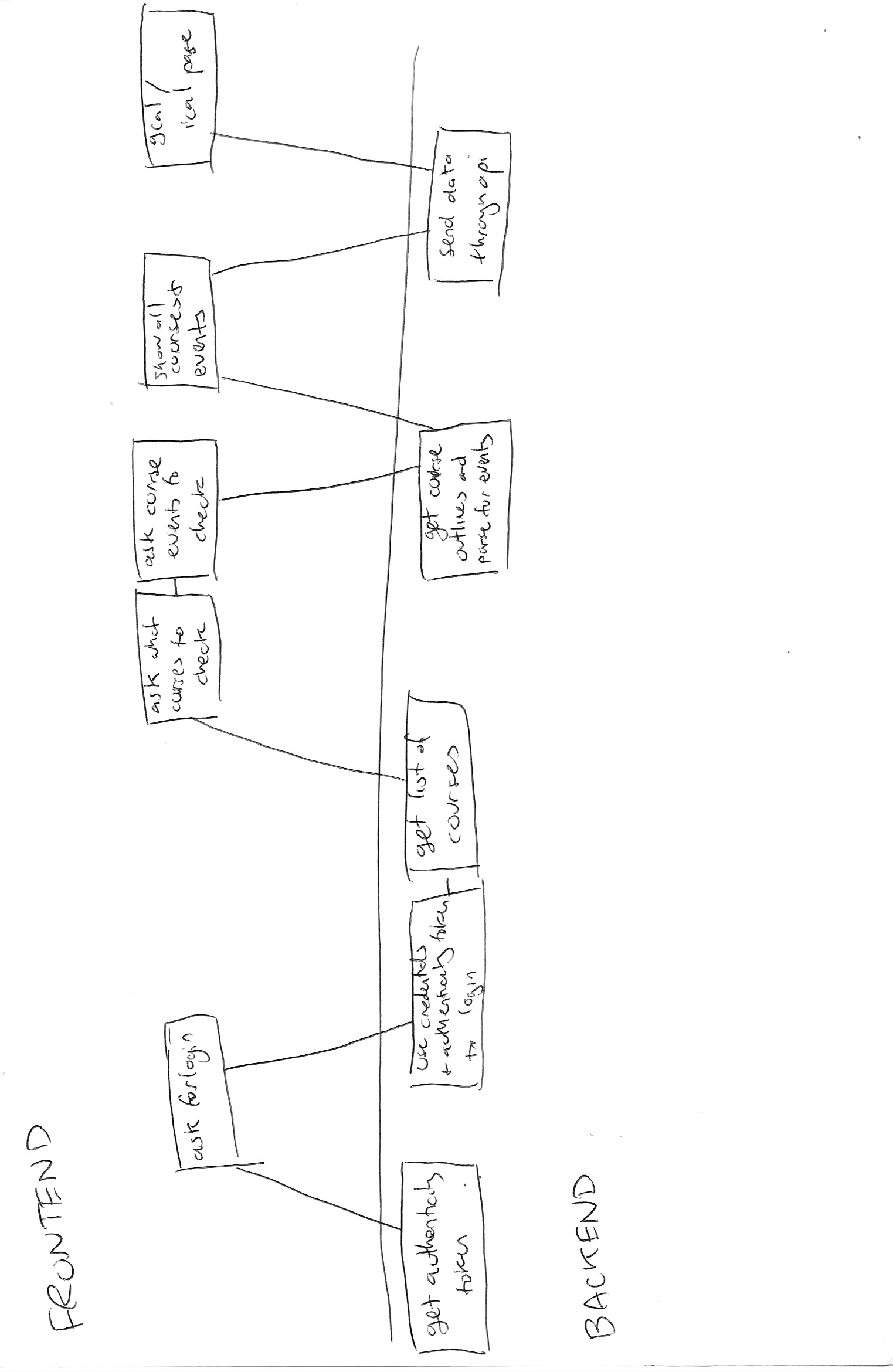
This project is centralised upon addressing the concerns of a typical UNSW WebCSM3 student whose time is now limited with the introduction of a tri-semester calendar system, prompting courses to accelerate and increase the workload which is necessary each week. Specifically with the initial weeks of a term going straight into course content, this provides a student little time to scope their courses requirements, through the course outline. Consequently this project intends to create an application in which a UNSW WebCMS3 student course commitments can be accumulated and exported to a digital calendar. This would then solve the issue of a typical student having work/family/university commitments providing more time for recreational activities or study.

Specifically our web application will provide the user with the ability to view all of their courses’ weighted material (Assignments, Class Tests, Laboratory Exercises, etc) and their respective deadlines through an automated process. After displaying a summary of all of the courses the user will then have the option to export the assessment tasks onto a digital calendar (Google Calendar or iCal/Apple’s Calendar app) at their specific due dates. Ultimately our web application aspires to provide a rapid and simple user experience.

# 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). This interaction between the front end and back end for when a user uses our application is conceptualised in Figure 2.1. Consequently the website application will embody a client server architecture, which will partition the workload accordingly through a centralised server as a client requests services. This entire software architecture that we will implement into our application is visually represented in Figure 2.2. Furthermore since typically a UNSW WebCMS3 student would only get their courses assessments then export it into a digital calendar once (as course outlines rarely change once published), our application has no need for persistence. Hence our software architecture will not use any permanent database and instead will temporarily store any necessary data during runtime.

Since the web application will be run using the local computer address on a computer, 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

Straight Edges = External Application

Rounded Edges = Python Modules

Circle = Templates

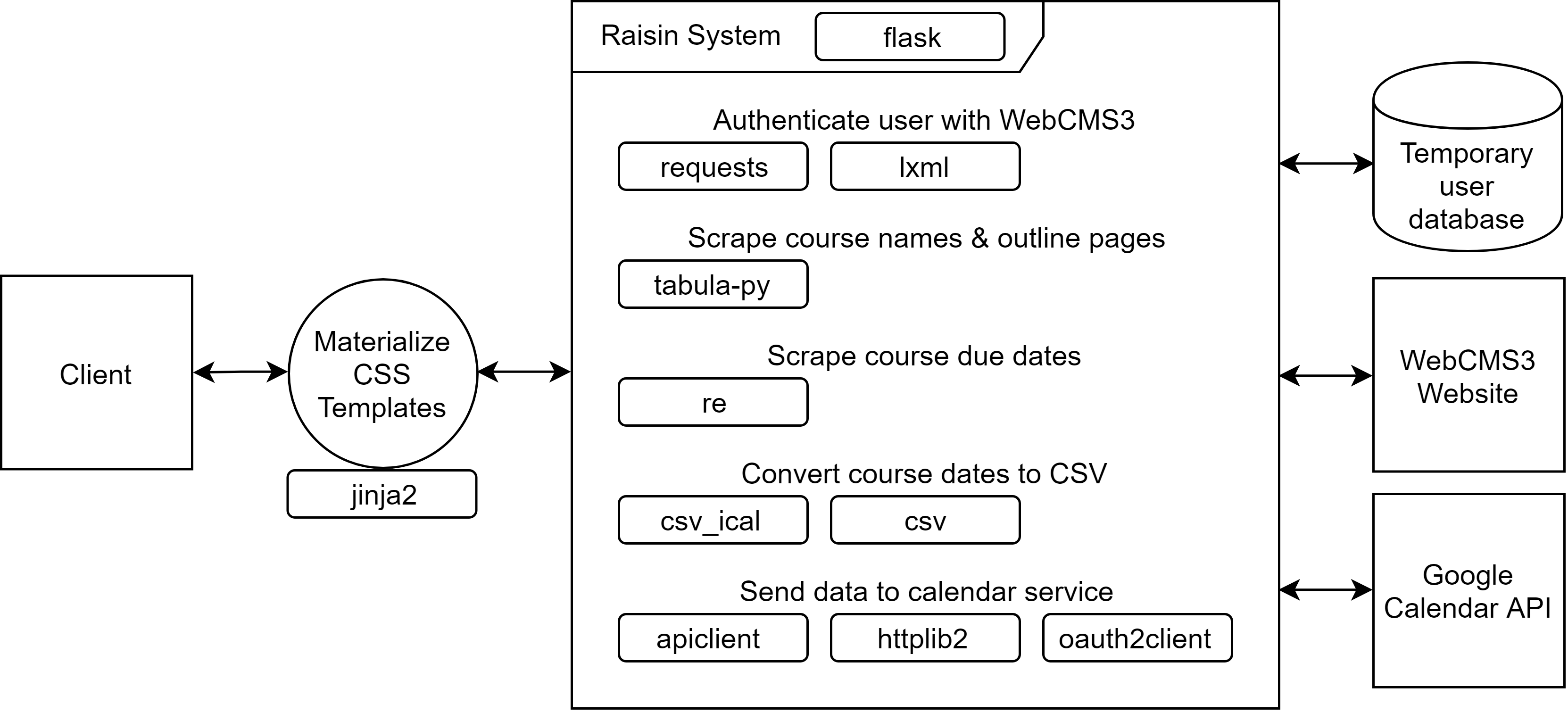


Figure 2.2: System Architecture Diagram of Raisin

# 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 of which 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 application programming interface. The application will need to connect to a UNSW student’s currently enrolled courses which can be found through WebCMS3’s data concerning a course and it’s course outline. Similarly in order to export a course’s contents into a calendar form, the application will require access to Google Calendar’s API in order to connect to a UNSW student’s Google account in order to create the event. This then automatically imports then imports the event directly into the user’s Google Calendar after allowing permission. Subsequently both processes will require the user to provide their login credentials (as certain courses on WebCMS3 do not allow public access to a course’s details) which will not be stored by the application. Conversely, when offering the course assessments for the iCalendar/Apple’s Calendar, our application will provide a download of an “.ics” file which will be created internally the application’s server.

In order to find a UNSW WebCMS3 student’s courses and their respective details, our group will need to develop a PDF scraper and a web scraper. The web scraper will allow us to scan over the courses and the text in their course outlines to extract the assessments. A PDF scraper will must also be implemented in the case that a course outline is in a PDF format; SENG2021’s course outline. The process that our scrapers will undertake is outlined below:

1. Server scrapes the authenticity token from the login form
2. User manually inputs login details for WebCMS3
3. User’s login details and token are passed to WebCMS3 for authentication
4. WebCMS3’s course navigation bar is scraped for course names and URLs (which contains currently enrolled courses)
5. On Raisin, the user is asked what courses and event types they want
6. After selections have been made, the course outline documents are scraped for each course selected

* If the course outline is a PDF, Python module tabula-py extracts all tables

1. Specifically the 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.
2. If any of these are found, their immediate surroundings are searched for “Week X” or “Exam Period”, as well as a “% weighting” - If found, these values are added to a dictionary
3. Once all courses are searched, the course assessments and their due dates are formatted then presented to the user on Raisin

# Software Architecture: Front-End

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

**Flask**

Flask will serve as a straightforward micro framework of our web application which is beneficial so everyone our group collective understands the code so we can efficiently create a website with our limited project timeframe.

**Jinja2**

Each individual HTML webpage will be created using in Jinja2, as its flexibility and ahead time complication will ensure our web application can achieve all of its functions and features so a seamlessly rapid user experience is provided.

**Materialize**

These webpages will then be refashioned using the CSS, Materialize, so that a user can effortlessly navigate and operate our web application.

Together through the user of various Python libraries and frameworks will guarantee our web application achieves a high quality user experience.

# Initial Software Design

Introduction

Each user story of each respective feature will be presented together such that their priority can be explicitly represented. Each user story will embrace SMART principles so that our project can be realistically visualised to achieve success.

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 5.1: 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 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 |

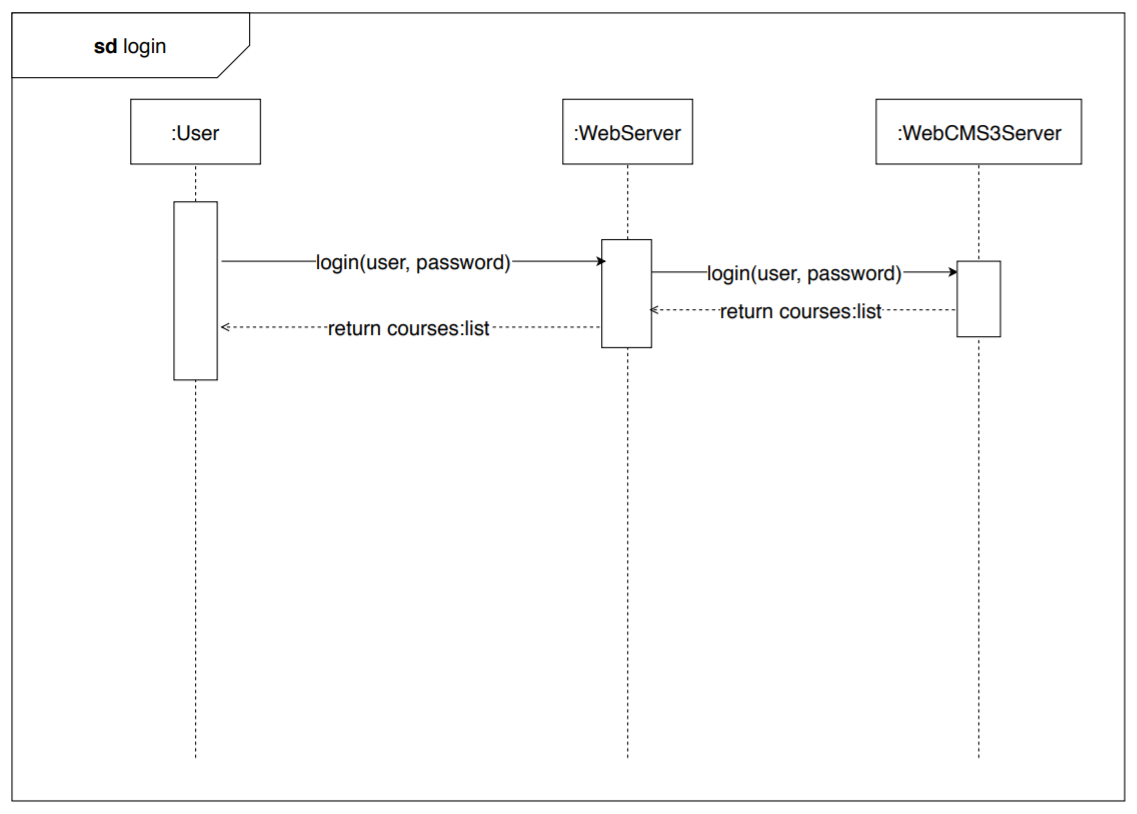


Figure 5.2: Sequence Diagram for Raisin’s Login Process

|  |
| --- |
| **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 |

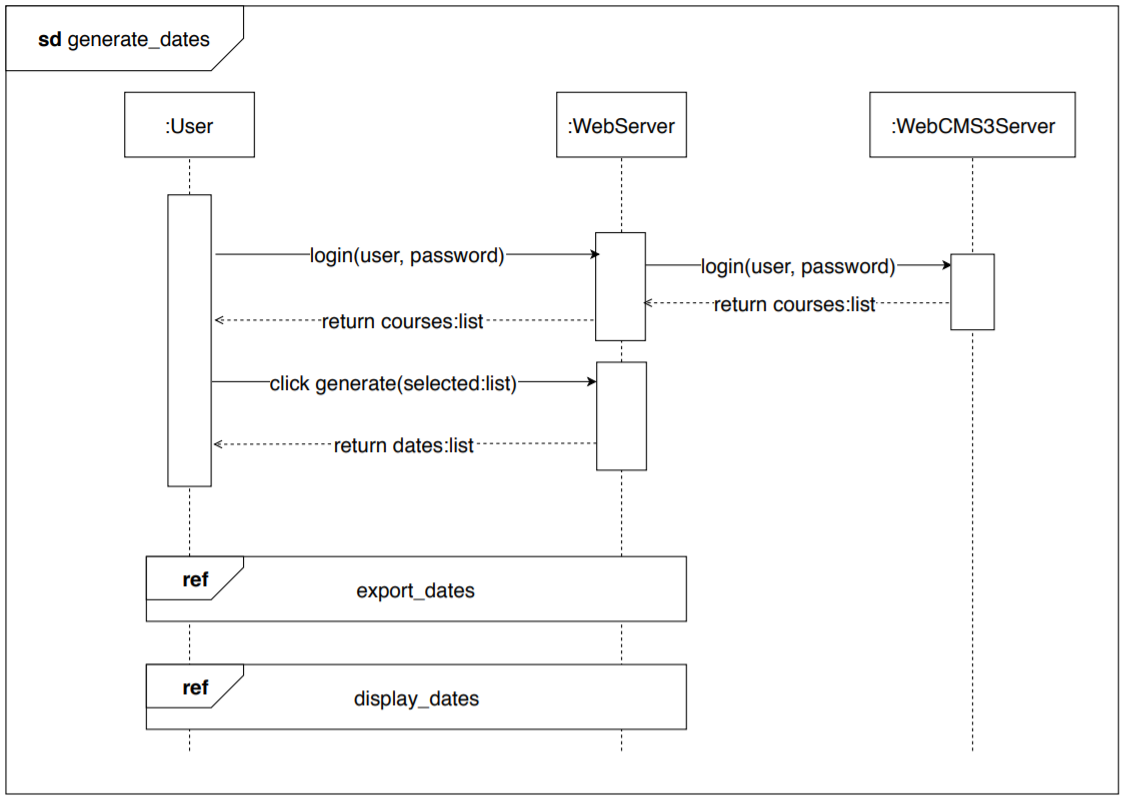


Figure 5.3: Sequence Diagram for Viewing Course Commitments

|  |
| --- |
| **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 |

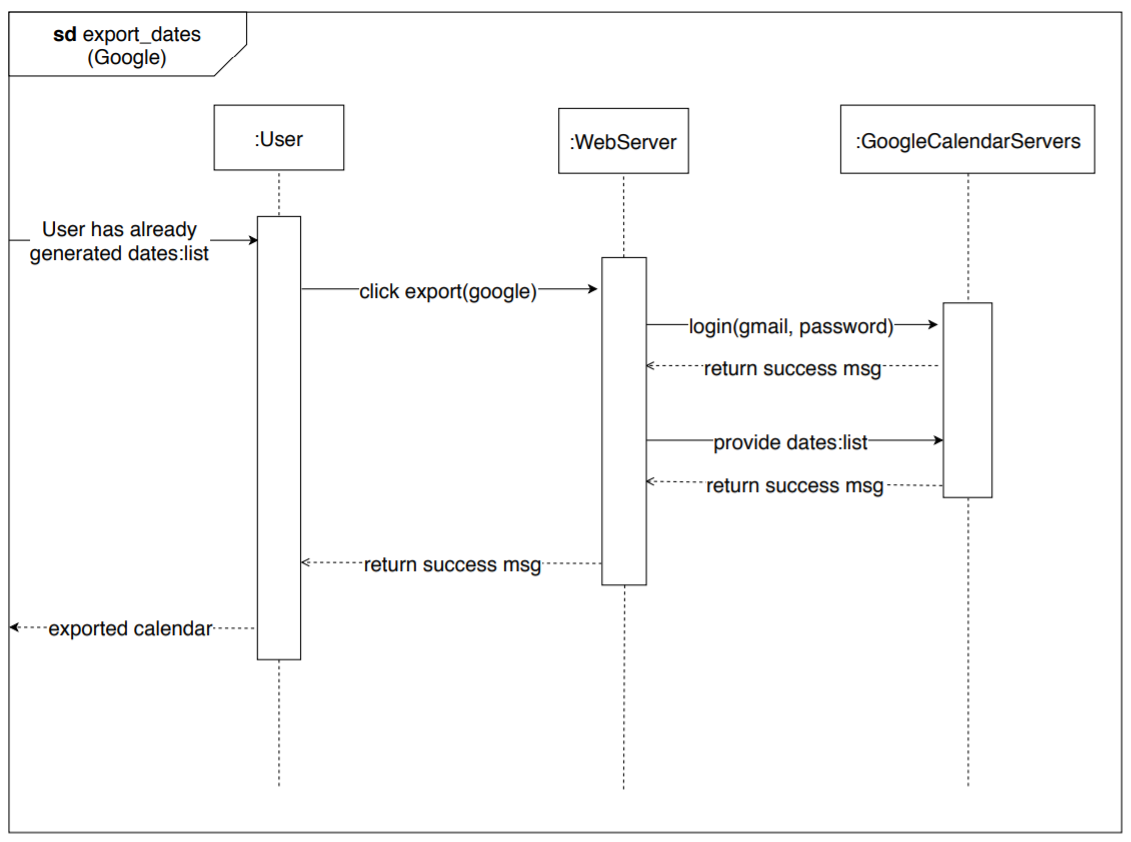


Figure 5.4: Sequence Diagram for Exporting to Google Calendar

|  |
| --- |
| **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 |

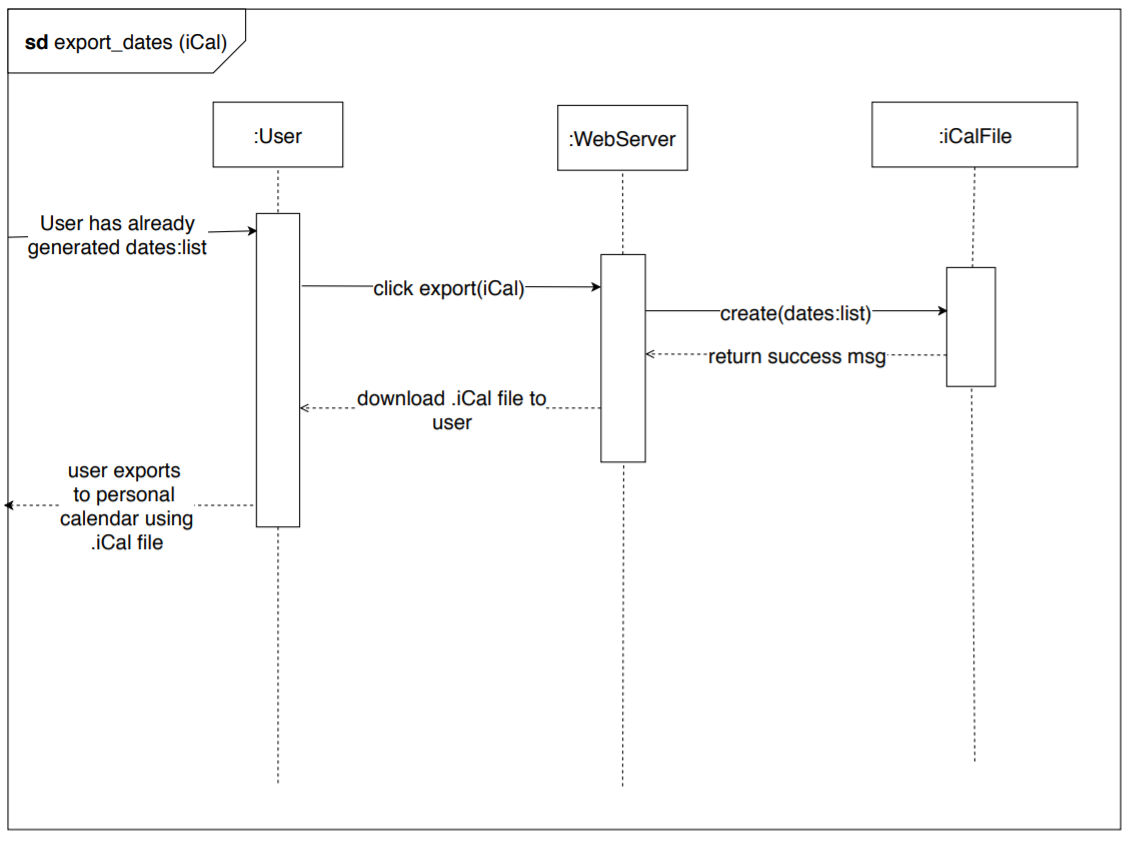


Figure 5.5: Sequence Diagram for Exporting to iCal

|  |
| --- |
| **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 |

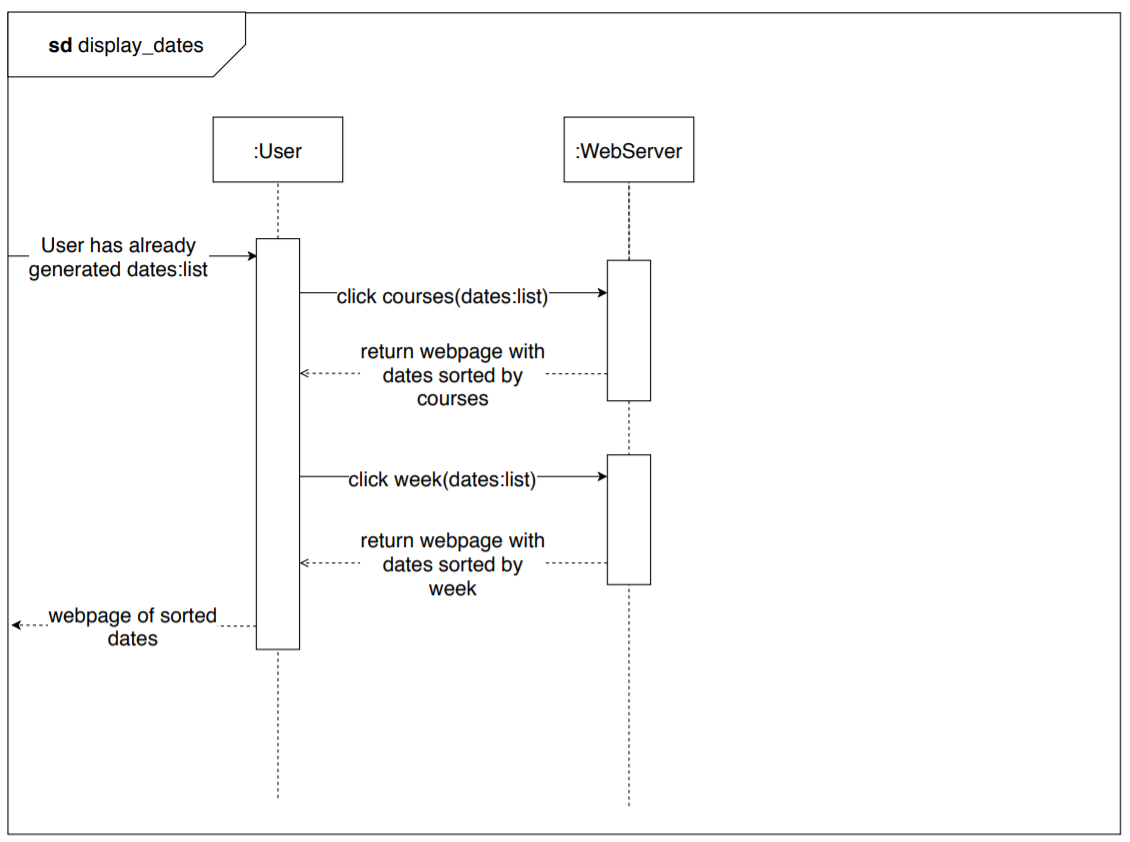


Figure 5.6: Sequence Diagram for Viewing Course Assessments Tasks by Group