Software Requirements Specification Final Report For Student Life

Version 1.2

Prepared By Group 4

June 19, 2020

Table of Contents

1. Introduction

1.1 Purpose

This document intends to describe the requirements of the Student Life application. This application is designed with the goal of assisting students to plan and manage their social, personal, and academic life.

1.2 Document Conventions

The functional requirements for this document are organized around the use cases within each user class.

1.3 Project Vision and Product Scope

1.3.1 Project Vision

Students who intend to use the Student Life application will be able to manage their university life from their smart device. Students shall be able to add and manage events in the calendar feature. Additionally, students shall be able to converse with tutors, peers, lecturers, and clubs/groups through a messaging functionality. Furthermore, students can also borrow/request access to resources, both from the university's library system and other students.

1.3.2 Product Scope

Student life is an application that provides Adelaide University students with access to a range of Adelaide university related tools and platforms. The application will integrate with existing university systems in order to make it easier for students to find and access these services and systems. The application will also have the functionalities of calendar management, messaging capabilities, and requesting/borrowing equipment from fellow students.

1.4 References

Existing websites and mobile systems used as inspiration for design choices with respect to the system's functionality include:

www.mystudylife.com
m.harvard.edu/
www.latrobe.edu.au/students/study-resources/study-tools/mylatrobe-app
www.uow.edu.au/student/life/myuow
ask.unimelb.edu.au/app
www.uwa.edu.au/students/campus-life/uwa-app

2. Overall Description

2.1 User Classes and Their Use Cases

2.1.1 Anonymous User

An anonymous user is an individual who has not presented the system with credentials. They are only able to access the login screen and option to register an account

UC – AU – 1: Login

UC – AU – 2: Register

2.1.2 Logged-in Student

A student is any individual who is registered as a student at the university and has verified this when they create their account. While logged in they can create personal challenges, receive and accept event invitations, and use the instant messaging system.

UC – S – 1 : Create a challenge

UC – S – 2 : Receive and accept event invites

UC – S – 3 : Access and use instant messaging

UC – S – 4 : Create a group chat

UC – S – 5 : Search for other users

UC – S – 6 : Set event reminder time

UC – S – 7 : Submit an assignment

UC – S – 8 : Download a lecture

UC – S – 9 : Lend an item

UC – S – 10: Borrow an item

UC – S – 11: Form a group calendar

2.1.3 Logged-in Coordinator User

A Coordinator User has the same privileges as a student, in addition to access to the event systems, allowing them to create and manage events.

UC – CU - 1: Create an event

UC – CU - 2: Delete an event

2.2 Operating Environment

Student life shall be compatible with all standard smartphones (Apple, Samsung, Google, Huawei and LG). The system shall be supported on devices running iOS 11 or Android OS 9 or later.

2.3 Design and Implementation Constraints

CO-1: Commercially available systems shall be used for the messaging and calendar features of the system, limiting the amount of unique code that needs to be created.

CO-2: The commercially available systems must be able to integrate with existing university systems.

2.4 User Documentation

The developer shall provide the necessary user documentation on how to use the system, system features and solutions to common problems for users through the Help menu tab.

2.5 Assumptions and Dependencies

Multiple functions of the system are dependent upon the integration of other software systems such as the calendar system and university assignment submission system.

3. System Features

3.1 Use cases for Anonymous Users

UC – AU – 1: Login

Description: An anonymous user will be able to login to the client.

System.login - (Priority=H)

Upon accessing the client, the system shall prompt the anonymous user for login credentials. Prior to the login, the anonymous user shall have no access to the rest of the system.

System.register - (Priority=H)

Upon accessing the client, the system shall display an option to register an account. This will prompt the user for an email and password. Completing this action will result in the user being added to the system, and granted access to their account.

3.2 Use cases for Logged-in Students

UC – S – 1: Create a challenge

Description: A logged-in student shall be able to create a challenge for themselves from the system. These challenges exist as a list of goals that the user can tick off as they are completed.

Challenge.create - (Priority=L)

When prompted, the system shall allow the logged-in student to create challenges using a variety of templates, and customization tools. These templates will allow for a range of challenge options, such as daily life goals, or even year long targets.

Challenge.info: - (Priority=L)
During an active challenge, the system shall display detailed information about the progress of the challenge from the challenge's page.

UC – S – 2: Receive and accept event invites
Description: A logged-in student shall be able to receive and accept event invites from other users.

Event.invite.notify: - (Priority=H)
When a user has been invited to an event, the system shall send a notification to the user regarding the event. Each user shall have the option to request more information and to accept or decline.

Event.info: - (Priority=M)
At all times, the system invitation shall provide a link in the invitation, to a page with the event information. This should include information about other staff members, the time and date, the coordinator, and general information about the event.

Event.accept: - (Priority=H)
If the user selects the accept option, the system shall create an entry in their calendar, and add the user to the allocated group chat, giving access to further event information.

Event.decline - (Priority = H)
If the user selects the decline option, the system shall delete the invitation.

UC – S – 3: Access and use instant messaging
Description: A logged-in student shall be able to access and use the instant messaging system.

User.message: - (Priority=L)
While logged in, the system shall allow each user to directly message other users.

UC – S – 4: Create a group chat
Description: A logged-in student shall be able to create a group chat with other users.

Group.create: - (Priority=H)
While logged in, the system shall allow each user to create group chats with class groups, clubs, teams, and fellow peers.

UC – S – 5: Search for other users
Description: A logged-in student shall be able to search for other users.

User.search: - (Priority=M)
While logged in, the system shall allow each user to search for other students and staff using their full names or ID numbers. If the other student or staff member is not found, the system should display “User not found”.

User.info: - (Priority=L)
When a user is selected from the search, the system shall display the profile of the chosen user in a popup tab. The profile shall show the user’s profile picture, current degree and year of study, and any clubs/teams they are a part of.

UC – S – 6: Set event reminder time
Description: A logged-in student shall be able to set a reminder for an upcoming event.

Event.set.reminder: - (Priority=L)
While invited to an event, the system should allow each user to select when they would like a reminder for an upcoming event. The reminder time should be able to be set to up to 24 hours before an event begins.

Event.notify: - (Priority=L)
If a reminder time has been set, the system shall notify the user via a device notification at the set time of the reminder.

UC – S – 7: Submit an assignment
Description: A logged-in student shall be able to submit an assignment to the university from the system.

Assignment.info: - (Priority=M)
If the assignment has been posted, the system shall provide a link to the assignment page with the assignment information, from the assignments list page.

Assignment.submit: - (Priority=M)
From the assignment page, the system shall allow each user within the assignment’s course to submit files for the assignment. The maximum file size and deadline for submissions will be set by the course coordinator.

UC – S – 8: Download a lecture
Description: A logged-in student shall be able to download available lectures from their classes.

Lecture.download - (Priority=M)
If the lecture video has been uploaded and the user's device has more than 1 GB of available storage, the system shall allow each user to download lectures onto their device.

Lecture.info: - (Priority=M)
When the download has been prompted, the system shall display information to the user about the lecture. This should include the file size, file name and the date the lecture was recorded.

UC – S – 9: Lend an item
Description: A logged-in student shall be able to lend items to other users.

Item.post: - (Priority=L)
From a designated page, the system shall allow each user to publicly post any items they are willing to lend to other students.

Item.lend: - (Priority=L)
While viewing borrow requests, the system shall allow each user to select a student to lend an item to. Once selected, the user who requested the item will receive a request to have the item be lent to them.

Item.info: - (Priority=L)
While an item is being borrowed, the system shall display its status to both parties in the transaction for the duration of the lending/borrowing. The lending/borrowing tracking shall be ended when the item is manually marked as returned by the lender.

UC – S – 10: Borrow an item
Description: A logged-in student shall be able to borrow items from other users.

Item.request: - (Priority=L)
From a designated page, the system shall allow each user to publicly request any items they want to borrow.

Item.borrow: - (Priority=L)
Once a student has selected a user to lend an item to, the system shall notify the user requesting the item. If the borrower accepts the request, the item should be listed as "borrowed" in the borrower's "items" view and "lent" in the lender's "items" view.

UC – S – 11: Form a group calendar
Description: A logged-in student shall be able to form a group calendar that tracks the events of the users in the group.

Group.calendar.create: - (Priority=M)
While a user is in a group, the system shall provide an option for a group calendar to be created in which students from that group can add events to.

Group.calendar.info: - (Priority=M)
If a group calendar has been created, the system shall allow each user in the group to see the information that the calendar contains.

3.3 Use cases for Logged-in Coordinator Users

UC – CU – 1: Create an event
Description: A logged-in coordinator user shall be able to create an event, set the details of the event and invite others to it.

Event.create: - (Priority=H)
While logged in, the system shall provide each coordinator user the ability to create an event by inputting the required details of the event in a designated page.

Event.invite: - (Priority=H)
While logged in, the system shall provide each coordinator user an option to invite other users by searching for them, from the event's page.

UC – CU – 2: Delete an event
Description: A logged-in coordinator user can delete events which they have created. This shall remove the event from the system, and notify all other staff members who accepted the invitation of the deletion.

Event.delete - (Priority=H)
Once an event has been created, the system shall provide an option for the creator of the event to remove it from the system. Prior to deletion, a second prompt shall ask for confirmation of the deletion. If the confirmation is accepted, the system shall purge the event from the Coordinator User's account, and from the staff members' accounts. The system shall also delete any pending invites for the deleted event.

4. External Interface Requirements

4.1 User Interfaces

UI-1: Fields with known maximum length, such as the 4 characters for setting calendar reminder time, shall not be wider than the expected entries.

UI-2: At the bottom of each tab, the application shall display the copyright notice: Copyright ©[current year] The University Of Adelaide.

UI-3: From any page of the application, tabs to other key pages shall be visible in a bookmark bar located at the bottom of the screen.

UI-4: When an assignment submission has been successfully completed the message “submitted successfully” will be displayed.

4.2 Software Interfaces

External links will exist from StudentLife to the following sites:

4.2.1 *myuni.adelaide.edu.au*
4.2.2 *access.adelaide.edu.au*
4.2.3 *mail.google.com*
4.2.4 *adelaide.edu.au*
4.2.5 *librarysearch.adelaide.edu.au*

Links are expected to come into StudentLife from the following sites:

4.2.6 *calendar.google.com*
4.2.7 *www.icloud.com*

4.3 Communications Interfaces

There are no specific communications interfaces besides URL links.

5. Quality Attributes

5.1 Performance Requirements

PER-1: 90% of the application’s pages shall fully render in an average of 1 second or less and take no longer than 2 seconds, from the time the user requests the page, over a 4 megabytes/second internet network connection

PER-2: The system shall accommodate an average of 25,000 users and a maximum of 30,000 during the time window 9:00 AM to 5:00 PM UTC+9:30, when peak usage is expected.

5.2 Security

SEC-1: The application shall adhere to Adelaide University’s security protocols when transferring any private information regarding users.

SEC-2: Students shall not be able to access other students’ calendars without being granted access beforehand.

SEC-3: The application shall log user activity data to help determine the cause of a breach if one is to occur.

SEC-4: The system shall only allow an account to be created if the signup email is associated with the University of Adelaide, i.e. @adelaide.edu.au, and after a verification email has been sent and confirmed by the user.

5.3 Extensibility

EXT-1: The application shall be designed to permit alterations and the integration of other software systems, without altering more than 10% of the code for any given function in the system already.

5.4 Usability

USA-1: Students shall be able to navigate the application and complete tasks correctly without requiring any tutorial at a 95% success rate.

5.5 Availability

AVA-1: The system shall be available at least 97% of the time between 7:00 AM and 1:00 AM UTC+9:30, and at least 90% of the time between 1:00 AM and 7:00 AM UTC+9:30, excluding system downtime scheduled for maintenance.

6. Business rules

BR-1: Before submitting an assignment, a student shall be required to indicate that the work they are submitting is their own, original work.

Appendix A: Use Cases

UC ID and Name:	UC-1: Submit an Assignment		
Created By:	Spencer Edwards	Date Created:	May 2, 2020
Primary Actor:	Student	Secondary Actors:	N/A
Trigger:	Student clicks the Submit Assignment button		
Description:	A student accesses the assignment webpage from the student life mobile application system (SLMAS), clicks the Submit Assignment button, selects file/s to be submitted and clicks a final submit button.		
Preconditions:	PRE-1: Student is logged into the SLMAS. PRE-2: It is not past the assignment due date.		
Postconditions:	POST-1: The submitted file/s is uploaded and stored into the SLMAS. POST-2: Status of assignment is updated to 'Submitted'		
Normal Flow:	1.0: Submitting an assignment 1. Student selects and enters the assignment webpage 2. SLMAS displays a menu of available interactions that the student can use (Download, email, submit etc.) 3. Student selects the submit option 4. SLMAS displays an add file button along with a submit button 5. Student clicks add file and selects a file from their device 6. SLMAS displays the file name under a files tab 7. Student checks a confirmation box and clicks submit 8. SLMAS confirms acceptance of the submitting 9. SLMAS stores file and updates the assignment status to 'Submitted'		

Alternative Flows:	N/A - submission will only be available from the specified assignments webpage
Exceptions:	1.0.E1: Submission request is after specified deadline 1. SLMAS informs student that it is past the deadline and it is too late to submit the assignment. 2. SLMAS cancels the use case 1.0.E2: The selected file to submit is too large 1. SLMAS informs student that the chosen file is too large to be submitted 2a. If student exits the page, then SLMAS cancels the use case 2b. Else, return to step 4 of normal flow
Priority:	Medium
Frequency of Use:	Average of 3000 uses per day. Peak usage load for this use case is between 6:00 P.M. and 12:00 A.M UTC+09:30
Business Rules:	N/A
Other Information:	1. Student shall be able submit multiple times, overriding the latest submission 2. Student shall be able to view and download the latest submission
Assumptions:	The assignment is to be submitted through online methods

UC ID and Name:	UC-2: Download a Lecture		
Created By:	Spencer Edwards	Date Created:	May 2, 2020
Primary Actor:	Student	Secondary Actors:	N/A
Trigger:	Student clicks the Download Lecture button		
Description:	A student accesses the list of available lectures page from the SLMAS, selects the lecture to be downloaded and selects a Download Lecture button.		
Preconditions:	1. Student is logged into the SLMAS. 2. The selected lecture has been uploaded to the SLMAS		

Postconditions:	1. The selected lecture is downloaded onto the student's Downloaded Lectures folder.
Normal Flow:	2.0: Downloading a lecture from the Lectures webpage 1. Student selects and enters the class's lectures webpage 2. SLMAS displays a list of lectures, including upcoming lectures, and an options button at the side of each listed lecture 3. Student clicks the options button and selects the Download Lecture button 4. SLMAS displays information about the lecture download, including file size, and prompts student to confirm 5. Student confirms the download 6. SLMAS displays progress bar of the download and expected time to complete 7. SLMAS sends a notification when download is complete
Alternative Flows:	2.1: Downloading a lecture from the Lecture video player 1. From an individual lecture's video player, an options button is available near the other video controls 2. Return to step 3 of normal flow
Exceptions:	2.0.E1: There is not enough storage space for the downloaded lecture 1. SLMAS informs student that they do not have enough storage space for this download 2. SLMAS cancels the use case
Priority:	Medium
Frequency of Use:	Average of 1500 uses per day. Peak usage load for this use case is between 10:00 A.M. and 6:00 P.M UTC+09:30
Business Rules:	N/A
Other Information:	1. Student shall be able cancel the download at any time before the download is completed 2. Student shall be able to delete downloaded lectures 3. SLMAS shall not allow a lecture to be downloaded if it is already present in the Downloaded Lectures folder 4. SLMAS will allow up to 3 concurrent downloads 5. SLMAS will limit the Downloaded Lectures folder size depending on the device's storage capabilities
Assumptions:	The lecture was recorded and is going to be uploaded

UC ID and Name:	UC-3 Notify calendar event		
Created By:	Sarah Telford	Date Created:	01.05.20
Primary Actor:	Calendar feature	Secondary Actors:	User
Trigger:	An event scheduled on the user's calendar is set to begin in an hour's time.		
Description:	This feature ensures that students are notified an hour before a scheduled event is about to begin.		
Preconditions:	PRE-1 User has created an account PRE-2 Event has been added by user to their calendar with a set date and time PRE-3 Notifications have been enabled by user		
Postconditions:	POST- 1 Notification will appear on user's device and hour before event		
Normal Flow:	3.1 Event automatically set reminder time to 1 hour before beginning event 3.2 System automatically identifies scheduled events beginning in the next hour on current date of calendar 3.3 System will notify user if there is an event beginning in an hour's time		
Alternative Flows:	3.1.1 User has modified the set reminder time (i.e. notify day of event) 3.2.1 System automatically identifies scheduled events beginning in set reminder time on current date of calendar 3.3.1 System will notify user if there is an event beginning at set time		
Exceptions:	3.0.E1 User does not have set time for reminder System will not notify user about upcoming event 3.0.E2.1 System identifies multiple events occurring within set reminder time System will display a reminder for each event occurring 3.>0.E1 User has different set reminder times for each event System will display event reminder dependent on reminder set time of each event		
Priority:	Medium		
Frequency of Use:	use dependent on number of events scheduled		
Business Rules:	N/A		
Other Information:	Forming intuitive individual and group calendars on the app is desirable to help the students to follow-up with their plans		
Assumptions:	The calendar allows user to alter reminder times		

UC ID and Name:	UC-4 Forming group chats		
Created By:	Sarah Telford	Date Created:	01.05.20
Primary Actor:	User	Secondary Actors:	Other user accounts, communication channel
Trigger:	User pressed create groups tab on communication page of student life system		
Description:	This allows users to create group for text-based communication by adding users to a group based on their first and last name.		
Preconditions:	PRE-1 User has created an account		
Postconditions:	POST-1 The system creates a group chat containing members added by user		
Normal Flow:	4.1 User presses create group tab on communication's page 4.2 System displays 'creating group' page of student life 4.3 User searches for full name of other users in search bar 4.4 System validates input 4.5 System displays list of students with matching name 4.6 User selects desired name 4.7 System adds names to list which is displayed to user 4.8 User presses 'DONE' button when finished selecting desired users' 4.9 The system creates a group chat containing all selected users		
Alternative Flows:	4.1.1 User joins/or is added to a group (i.e. class group, robotics club) System automatically adds user to chat with other members of group		
Exceptions:	4.0.E4 User input not validated by system or name inputted is not found in system System displays message "User not found" 4.0.E6 User selects name of user unintended to be in chat System allows user to deselect users in selected list 4.0.E8 User does not select 'DONE' after completing selection of users System does not create group and all selections are erased when communications page is closed		
Priority:	Low		
Frequency of Use:	Once per week per user		
Business Rules:	N/A		
Other Information:	System could enable the students to form groups for their common classes based on their interests and availabilities		
Assumptions:	Everyone at Adelaide university has an account User knows full name of another user		

UC ID and Name:	UC-S-5: Lending/borrowing item		
Created By:	Hubert Au	Date created:	2/5/20
Primary Actor:	Student	Secondary actors:	Borrower
Trigger:	The student selecting “lend an item”		
Description:	The system shall track when an item has been lent/borrowed between two students. The lent/borrowed status shall be visible to both parties in the transaction. The lending/borrowing shall be ended when the item is manually marked as returned by the lender.		
Preconditions:	1. The device is connected to the internet		
Postconditions:	1. The student who lent the item will have that item shown in their “lent items” tab 2. The borrower will have the item shown in their “borrowed items” tab		
Normal Flow:	5.0 1. The student navigates to the lending/borrowing page and chooses to lend an item, and the system opens a prompt that asks for who they wish to lend to and what they are lending. 2. The student sends the information that is asked for, and the borrower receives a request to accept the borrowing of the item. 3. If the borrowing of the item is accepted, the item is listed as “borrowed” or “lent” for the borrower and student respectively. 4. When the borrowed item is returned, the student can mark it as such, and the item is removed from their list of lent items.		
Alternative Flows:	nil		

Exceptions:	5.0.E1 – if the internet connection is disrupted when the request is being sent, the request will not go through. The student will be notified of this, and be prompted to check their internet connection. 5.0.E2 – if the borrower does not accept the request, the lender will be notified of this, and the item will not be shown as borrowed.
Priority:	Medium
Frequency of Use:	Student-dependent; anywhere from 0-20+ times per semester
Business Rules:	none
Other Information:	For ease of use, the system is conceptualised as one-way i.e. the lender must initiate the process. This should be of little issue, as both parties should have access to the system on their devices and be physically present for an exchange of an item, and thus it should not matter who initiates the procedure.
Assumptions:	A system already exists for interconnectivity between student users.

UC ID and Name:	UC-S-6: Forming group calendar		
Created By:	Hubert Au	Date created:	2/5/20
Primary Actor:	Student	Secondary actors:	Group members of the relevant group
Trigger:	The student selecting “form a group calendar”		
Description:	The system shall provide students with the ability to create separate calendars that tracks events for a given group. The separate calendar shall only be visible to members of the group the calendar was created for.		
Preconditions:	1. The device is connected to the internet 2. The student is in a group that they wish to form a calendar for		

Postconditions:	1. A separate calendar containing group-specific events will be available to group members
Normal Flow:	6.0 1.The student is in the view for all of their calendars, and selects the “form a group calendar” option; they are then presented with customisation options for the calendar. 2. The student confirms their calendar settings, and the system creates the calendar, which will then be visible to all group members.
Alternative Flows:	6.1 1.Student is in the main view for the group, and selects the “form a group calendar” option (wording of the option is purposefully the same as in the normal flow, for consistency); they are then presented with customisation options for the calendar. 2. Re-joins the normal flow at this point.
Exceptions:	6.0.E1 – if the internet connection is disrupted when the request is being sent, the request will not go through. The student will be notified of this, and be prompted to check their internet connection.
Priority:	Medium
Frequency of Use:	Student-dependent; anywhere from 0-4+ times per semester
Business Rules:	none
Other Information:	It would be possible to automatically create a group calendar upon the formation of a group; however, this is not what is asked for in the system description, and thus a manual option has been selected. Multiple group calendars are allowed as it may be useful to some users.
Assumptions:	A calendar system is already in place; one possible implementation of the personal calendar feature would be to create a general group calendar function, and have the personal calendar be only available to the student in question; thus, the base calendar function can cover both desired functionalities.

UC ID and Name:	UC-7. Creating Challenges		
Created By:	James March	Date Created:	3/5/2020
Primary Actor:	Challenged User	Secondary Actors:	N/A
Trigger:	Challenged User Indicates that they want to create a challenge		
Description:	The Challenged User creates a challenge they wish to fulfil. They can choose to have this challenge be re-occurring or a once off.		
Preconditions:	1. User is logged into the system and authenticated		
Postconditions:	1. Challenge is created and logged by the system 2. System tracks data related to the challenge		
Normal Flow:	7.0 Creating the Challenge 1. System prompts for title and description (see 7.0.E1) 2. System prompts Challenged user for if the challenge is a once-off or re-occurring (see 7.1) 3. System prompts Challenged User for data to track or offers manual check off 4. System prompts for confirmation of creation and displays the challenge 5. Challenged User selects (6a) “Confirm” or (7a) “Cancel” a) Challenge is created b) Challenge is added to system main screen c) Challenged User is returned to main screen d) Challenge is deleted e) Challenged User is returned to main screen		
Alternative Flows:	7.1 Re-occurring Challenge 1. Challenged User is prompted for if the challenge is daily/weekly/monthly 2. Challenged User is prompted for end conditions 3. Challenged User is returned to normal flow 7.0, 4		
Exceptions:	7.0.E1 Challenge title already exists		

	1. System compares title to all other challenges. If it already exists it informs the user and restarts flow 7.0. If not, returns to 7.0, 3.
Priority:	Low
Frequency of Use:	User is expected to create a challenge once a week.
Business Rules:	N/A
Other Information:	If the system crashes during creation, return to the main screen and display an error message. Challenges should be created on the cloud, however, also should be saved locally if the cloud is not reachable. When connected to a network, upload challenges and local data. If the application hangs or some other issue transpires, challenge can be discarded.
Assumptions:	User's device is capable of tracking data, and the application has permission to do so.

UC ID and Name:	UC-8. Creating Event		
Created By:	James March	Date Created:	3/5/2020
Primary Actor:	Event Coordinator	Secondary Actors:	Event Staff, Calendar
Trigger:	Event Coordinator indicates they want to create an event		
Description:	The Event Coordinator creates an event. They set a date, time, venue, expected attendance, staff list, run time. They can choose for the event to be private. Sends invitations to Event Staff members, creates a group chat for Staff and Coordinator, and creates a calendar event for the Coordinator.		
Preconditions:	1. User is logged into system and authenticated 2. User's account is authorised to create events		
Postconditions:	1. Event is created on system 2. Invitations are sent to staff members 3. Event appears in Coordinator's calendar 4. System creates a group chat for the Event		

Normal Flow:	8.0 Creating an Event 1. System prompts for Event name, venue, expected attendance, date, time, staff list, run time. 2. System creates event 3. System adds event to Event Coordinator's calendar 4. System sends invitations to indicated staff 5. System creates chatroom for Event Coordinator and over time adds the Staff members as they accept the invitation
Alternative Flows:	N/A
Exceptions:	N/A
Priority:	High
Frequency of Use:	Low. Expect few users to actively use this feature.
Business Rules:	N/A
Other Information:	If the system crashes during creation, return to the main screen and display an error message. Events should be created on the cloud. If the application does not have network access, creation of events should be not accessible. No limit to event creation as authority to create should be limited.
Assumptions:	Users are authorised to create events Staff have access to the application System does not care about other events in the calendar and will schedule events that conflict.

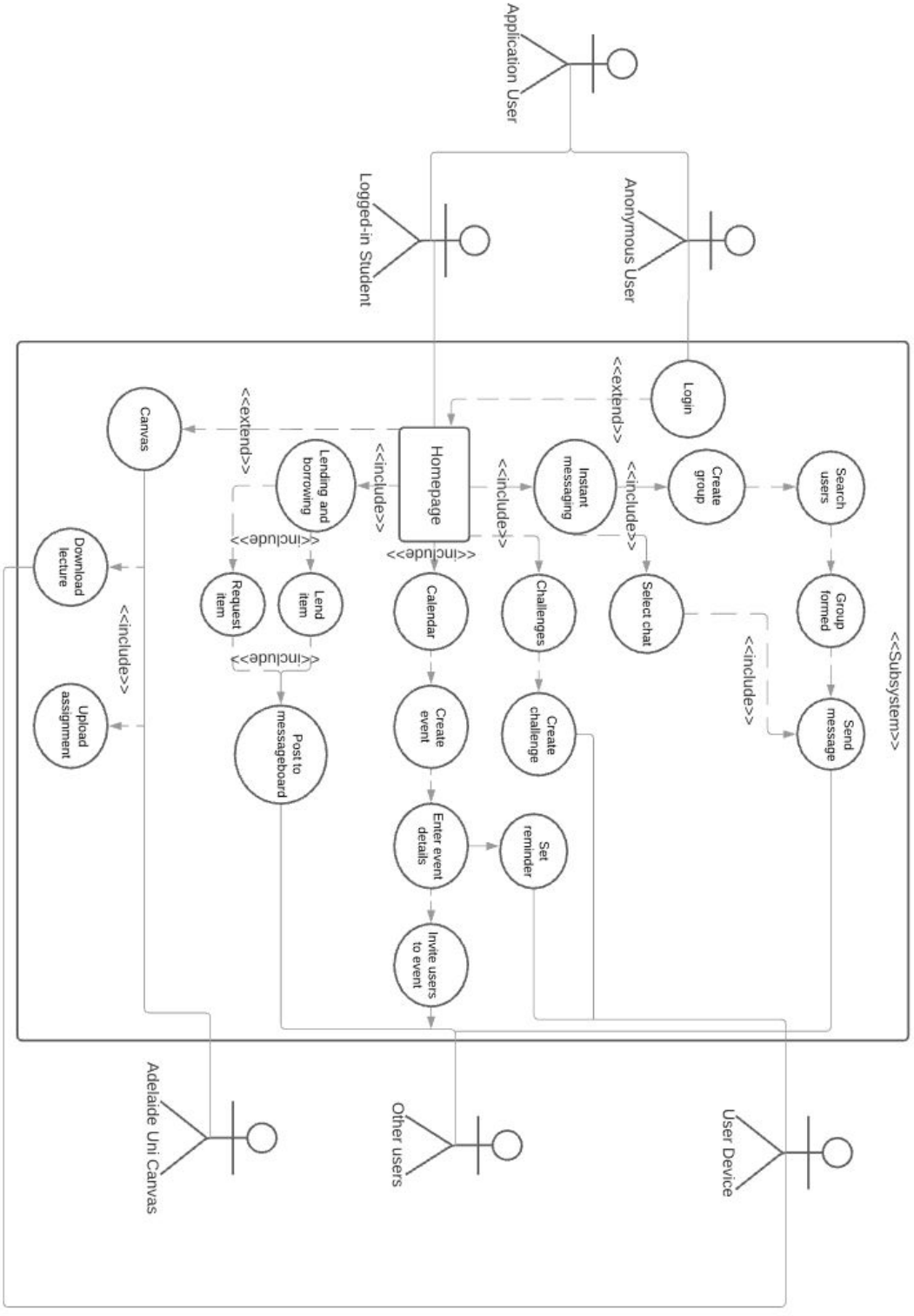
UC ID and Name:	UC-S-19: Login		
Created By:	Damon Evans	Date Created:	1/5/2020
Primary Actor:	The application User	Secondary Actors:	-
Trigger:	The system event that will trigger the on-launch login is the application starting up.		

Description:	On start up of the application the user will be prompted with a login box asking for their username (id) and their password. This is in place to ensure the security of the system and also provides multiple users to be able to login on the same device if necessary.
Preconditions:	1. The user's device must be connected to the internet 2. The user must have an account prior to logging in
Postconditions:	1. The user shall be logged in if username and password conditions are met 2. The system shall grant the user access to the system
Normal Flow:	1. The user opens the application 2. The user is met with a login prompt screen asking for a username and password 3. The user enters their username and password 4. System validates the inputs 5. If correct the user shall be logged into the system
Alternative Flows:	1. The user opens the application 2. The user is met with a login prompt screen asking for a username and password 3. The user enters their username and password 4. System validates the inputs 5. If incorrect the user will be prompted with a message saying as such and still be on the login screen
Exceptions:	9.0.E1: If the user is not connected to the internet on the login screen a message will be prompted telling them as such while stating that internet access is required. A large number of users might be trying to login which might slow down the authentication process.
Priority:	High priority
Frequency of Use:	0-5 times a day per user
Business Rules:	N/A
Other Information:	If the user does not close the application on their device, it will not ask for them to login when they go back into the application unless it or the device has lost connection to the internet.
Assumptions:	That there is a database that has all the students (or other users) usernames passwords stored.

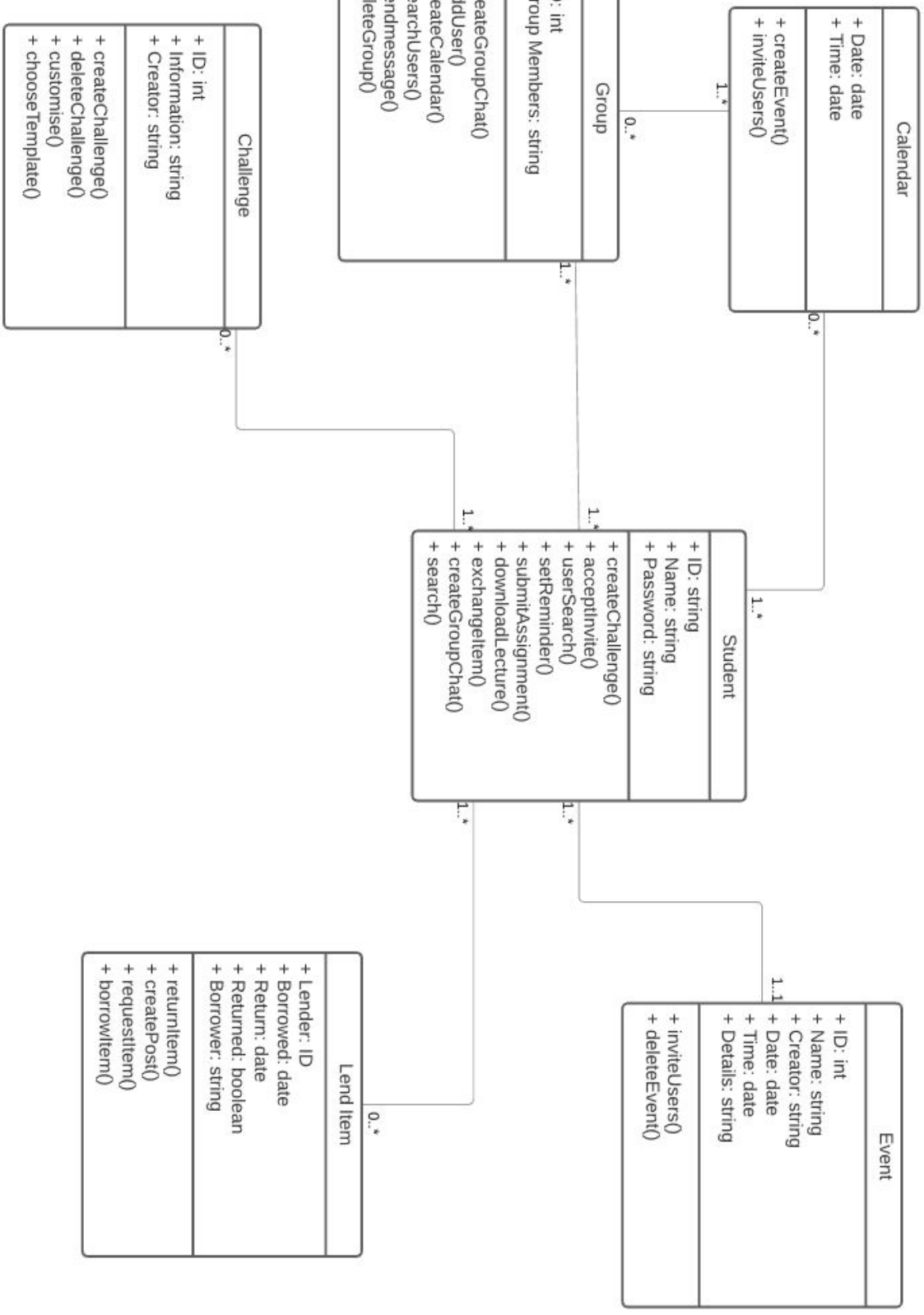
UC ID and Name:	UC-S-10: Join Event		
Created By:	Damon Evans	Date Created:	1/5/20
Primary Actor:	The application user	Secondary Actors:	Other user(s) of the application
Trigger:	When the student hits the button to join any available event.		
Description:	Students can plan events on the application so that other students can join if they are interested so it is necessary for the system to provide this feature.		
Preconditions:	1. The user is logged into the application 2. The user has hit the button to an available activity that they would like to join.		
Postconditions:	1. The user is accepted to join the activity 2. The event has been updated to register the student has joined in the database		
Normal Flow:	1. The user opens an event they are interested in that another user has created 2. The event opens displaying the appropriate information and a button to join (if applicable) 3. The user hits the button to join the event 4. The system updates placing the user into the event 5. The event page updates with the join event button now becoming a leave event button		
Alternative Flows:	-		
Exceptions:	10.0.E1: There could be a large number of users trying to access the particular event and hitting the join button around a similar time that could cause the request not to go through.		
Priority:	Medium Priority		
Frequency of Use:	0-2 times a day per user		
Business Rules:	N/A		
Other Information:	If the event is not available to be joined no button will be able to the user this can occur in events that have already happened, are full or are unavailable to that student.		
Assumptions:	That the user is connected to the internet throughout this entire process. There is a database for the system to store all the various events and users who have joined.		

Appendix B: Visual Models

Use Case Diagram



Class Diagram



Appendix C: Requirements Review Report

Damon Evans:

Completeness

The main requirements that the customer states are activities for the student to participate in such as lectures, exercises and group work, also having lunches and spending time at the Union House or Central Hub. The customers' outline also states the application may have different features to facilitate student life such as study-related activities or leisure-related activities. Where students can form groups based upon common interests/classes.

The requirements specification addresses the main needs of the client by implementing features which make those which the client wants available. For example, in the system, the user shall be able to make a group in which other students can join. However, while social features have been stated academic lectures such as the ability to participate in lectures and exercises have not been referred to. Overall, the majority of the main features in which the customer has outlined are said to be implemented bar a few academic activities.

The documentation has all the necessary sections and subsections for a proper software requirement specification. Comparing the document to that of the Pearls From Sand there are no notable exclusions that detriments the completeness. However, while the structure is correct and the information is mostly there, a notable oversight in the documentation can be found. This is as no error conditions are anticipated. There has been external software outlined in which the system shall be using. These are both named and properly listed under the correct sub-section.

Correctness

Overall, the requirements that are outlined in the specification do not conflict with one another as they are mostly separate actions which occur or can be completed by interacting with the system. However, some requirements are somewhat similar due to their use in the system such as creating a task or event. Nonetheless, they are both important requirements for the system and do achieve different outcomes.

All the requirements that have been specified in the document use unambiguous, grammatically correct language whilst still being clear and concise. The requirements written are also undoubtedly requirements for the potential system and are not design or implementation restraints as the outline what shall be implemented in the system in such a manner. However, while the specified requirements hold these attributes, they do not contain any error messages. These shall have to be implemented.

Quality Attributes

Proper performance and usability requirements are stated with the performance requirements outlining the exact performance qualities the system shall adhere to while the usability requirement gives a percentage of how well the students shall be able to use the said system. Security requirements have also been outlined with the system operating per Adelaide University's security protocols whilst outlining other requirements that the system will adhere to. However, any safety requirements have not been outlined. This is an important sub-section to be implemented as such safety features such as failure modes and functional safety should be outlined for any safe system.

The majority of quality requirements are measurable with numbers and percentages being given with such requirements from performance, usability and availability. However, extensibility, for example, has not been given a measurable trait. Such a trait that could measure this is how many other systems this one should be able to handle. Along with being measurable the functions that adhere to certain time criteria, in this case, performance has been outlined with specific timing criteria for it. Trade-offs for the quality attributes have also been taken into account with performance, usability and availability all having certain trade-offs. However, extensibility is again lacking with no outlined trade-offs.

Sarah Telford

We have formulated a range of requirements that support the main needs of the customer:

Mobile application

- "Student life shall be compatible with all standard smartphones (Apple, Samsung, Google, Huawei and LG)"

Formation of intuitive individual and group calendars/ Help with event organisation

- A logged-in student shall be able to set a reminder for an upcoming event.
- A logged-in student shall be able to form a group calendar that tracks the events of the users in the group.
- A logged-in student shall be able to receive and accept event invites from other users.
- A logged-in coordinator user shall be able to create an event, set the details of the event and invite others to it.
- A logged-in coordinator user can delete events which they have created. This will remove the event from the system and notify all other staff members who accepted the invitation of the deletion.

Formation of intuitive individual and group communities

- A logged-in student shall be able to access and use the instant messaging system.
- A logged-in student shall be able to create a group chat with other users.

Facilitate for lending/ borrowing study items

- A logged-in student shall be able to lend and borrow items from other users.

Creating challenges

- A logged-in student shall be able to create a challenge for themselves from the system.

Supporting educational activities

- A logged-in student shall be able to submit an assignment to the university from the system.
- A logged-in student shall be able to download available lectures from their classes.

Integrate with the existing systems

- The commercially available systems must be able to integrate with existing university systems.
- Commercially available systems shall be used for the messaging and calendar features of the system, limiting the amount of unique code that needs to be created.
- All anticipated software interfacing systems can be found in section 4.2

However, we have not specified all behaviours for anticipated error conditions. For instance, it has not specified what will happen if a logged-in student tries to submit an assignment to the university from the system which is larger than that allowed file size. In order to solve this issue, we will need to add a use case under the assignment submission requirement which states: "if the file uploaded is larger than the allowable file size, the system shall display a message saying, "file size too large, please try again"". Another instance where errors have not been identified is if the user does not complete the creation of challenges, events, group chats, or lending/borrowing submissions. These errors could be mitigated by specifying: "if the user has left a submission or field incomplete for a period of 24 hours, clear all entered data form the submission".

None of the requirements that have been created are duplicates, nor are any of them contradictory to each other. I believe that most of the requirements are written in clear, concise, unambiguous, grammatically correct language. However, there are a few requirements that are not written in this manner.

i) "A logged-in student shall be able to create a challenge for themselves from the system" - this requirement if ambiguous and unclear. The requirement could be rewritten as such "A logged-in student shall be able to create a challenge by entering the details of the challenge into the system"
ii) "A logged-in student shall be able to set a reminder for an upcoming event." – this requirement is ambiguous, due to the unspecified manner of the reminder. As such it could be rewritten as "A logged-in student shall be able to set a visual or audio reminder for an upcoming event."
iii) "A logged-in student shall be able to create a group chat with other users." - this requirement is ambiguous, due to the unspecified manner of the group chat. As such it could be rewritten as "A logged-in student shall be able to create an instant messaging group chat with other users".

All specified error messages, for instance, "user not found" if a user is not found when creating a group chat, are clear and concise with the displayed message. This ensure that any error is quickly and easily conveyed to the user. The only requirements that could be considered a solution and not actually requirements are those related to the creation of instant messaging groups. This is because the customer only specifies a need for the formation of intuitive individual and group communities, meaning this system does not necessarily have to be an instant messaging system. These requirements could be altered as such: A logged-in

student shall be able to access and community channels, A logged-in student shall be able to create a community channel with other users.

Each of the quality attributes have been thoroughly analysed and specified. These attributes have been specified through establishing system requirements related to the attribute. For example, security - the application shall adhere to Adelaide University's security protocols when transferring any private information regarding users. Within our project we have also added availability and extensibility quality attributes, which cover functionalities related to future development of the application and when the system should be available for use. All time critical functions have been identified and time criteria have been established. Time criteria are mainly evident in the requirement for performance and availability attributes. Each of the quality attributes are testable and measurable, for instance security can be tested by trying to create an account with a non-university related email or by checking that breaches are recorded.

Spencer Edwards:

Completeness

The main features included features that could help students follow-up with their plans effectively, and activities such as forming student communities, organizing events, lending/borrowing items and defining challenges. The requirements reflect this desire for facilitating student life through the following features: challenge creation, event creation, instant messaging, group chats, setting reminders, loan/borrow system, group calendars, assignment submission, lectures and timetables.

The requirements were written to try avoiding specifying how they should be implemented, to limit how the requirements affect design decisions. However, extra information was specified when it was necessary, to clarify and outline important details. The challenge creation requirement did not, however, include specific information on what the challenges could be. External software interfaces used in the system are listed and defined in the SRS document.

Correctness

All requirements are unique and do not conflict with each other, but where creation was involved (group calendar, groups, challenges etc.), deletion was usually not specified as a requirement. It was not written, but it is assumed that deletion privileges came along with the creation privileges. The requirements are each written clearly, concisely, unambiguously and grammatically correct. This was achieved by following Rupp's template for writing requirements when it was best to, and through multiple drafts. The *"User not found"* error message clearly and concisely displays the cause of error. No other error messages were specified, as the errors that could occur are general and are not specific to the system. It could be addressed during implementation. All requirements are requirements, but some included information about the design and implementation constraints when necessary. Also, the lend/borrow item requirement is actually two requirements and should be split as such.

Quality Attributes

The usability requirement was vague in what is considered a successful operation. The performance requirements clearly specified what the system should be capable of. The

security requirements considered various angles that are not immediately obvious, stating the protocols that should be followed and preventative measures. Safety was considered as part of security in this document.

Extensibility was documented to address the possibility of integrating other software systems over time. However, it was vague and did not include any acceptable trade-offs. The performance requirements specified the expected loading time at certain internet speeds. Event/deadline notifications were not discussed or identified as time-critical functions. This is a main feature of the system and should be considered due to their time-critical role. Not all the quality requirements are measurable, mainly due to their nature. However, the requirements that could be measured were not given a clear specification on how.

Hubert Au

Review Checklist

Completeness

☑ *Do the requirements address the main features that the customer needs?*

☐ *Is any needed information missing? If so, it is identified as TBD?*

Referring to the StudentLife specification document, the main desired functions appear to be forming groups and forming group calendars, both of which have been specified in the design. The desire for integration with pre-existing systems has also been noted. Although the ability to add new features based on user demand is not explicitly written, the modular structure of the system should allow for easy integration of new features.

☑ *Are the external software interfaces defined?*

The external software interfaces have been defined to the extent of our ability at this stage in the development process. In future, it may be decided that the use of further external software is desirable or necessary, and the design may be updated to reflect as such.

☐ *Is the expected behaviour documented for all anticipated error conditions?*

Detailed use cases following the use case templates have not been generated for all of the requirements, as that was outside of scope for this report. Thus, the expected behaviour for all anticipated error conditions has not been documented in full. It is expected that as the life cycle of the project continues, these use cases will be fully generated, and this condition will be met.

Correctness

☐ *Do any requirements conflict with or duplicate other requirements?*

None of the requirements conflict with or duplicate the other requirements. Although the co-ordinator user class has the same privileges as a regular student, along with additional abilities, the requirements are not duplicated, and the above is instead noted.

☐ *Is each requirement written in clear, concise, unambiguous, grammatically correct language?*

UC-S-1: “variety of template and customisation tools” is perhaps a little vague. Further specification regarding these templates and tools would be desirable.
“Detailed information”; it would be more exact to specify what information is being taken from the challenge’s page.
UC-S-7: Legality of “will be set by the course co-ordinator” could probably be increased to at least “should”.

☑ *Are any specified error messages clear and meaningful?*
All specified error messages have been made as clear as possible; for example, in UC-S-5, the error message that is given if the searched-for user is not found is “User not found”. This is very unambiguous, and would be difficult to misinterpret.

☑ *Are all requirements actually, requirements, not solutions or design or implementation constraints?*
All of the requirements in the requirements section are requirements. The descriptions for each feature are slightly looser in wording, but they are not the formal requirements.

Quality Attributes
☑ *Are all usability, performance, security, and safety objectives properly specified?*
All usability, performance, security and safety objections have been specified. Particular focus was given to the amount of users that the system should be designed for; student enrolment data for the university was consulted before an adequate conclusion could be reached.

☑ *Are other quality attributes documented and quantified, with the acceptable trade-offs specified?*
Other quality attributes are documented and quantified. The one possible exception to this is SEC-1, which only states the system “shall adhere to Adelaide University’s security protocols when transferring any private information regarding users”. However, it is almost certain that these security protocols are quantifiable, and thus this condition would still be met.

☑ *Are the time-critical functions identified, and timing criteria specified for them?*
The only requirement in the design that is currently time-critical is PER-1, and timing criteria have been specified.

☐ *Are all of the quality requirements measurable?*
The single extensibility requirement (EXT-1) is not specified in a quantifiable way. It would be prudent to specify this in a way that can be measured, so that going forward, the system can be designed to adhere more strictly to this requirement.

Appendix D: Individual Task 4A

James March

Section 4A:

The "Harvard Mobile Application" is a product that closely resembles the system that is outlined in the Student Life Application's description. Both cater to similar audiences, this being University Students, and aim to assist in similar ways. The website for the Harvard Mobile Application states that the application is designed to "improve the mobile experiences of students, faculty, staff, visitors and neighbours". This closely resembles the goal of the Student Life Application. My impression from the Harvard Mobile Application is that it seems to have an excessive number of features. The application seems over saturated with content and as such is difficult to navigate. For our Student Life Application, care needs to be taken in making the user interface intuitive, while also providing the features our clients need.

The "My Study Life" application is designed as a "paper planner and more". This application is designed to help students keep on top of their university course work and manage their time effectively. One of its main selling points is the cross-platform use, where it can be accessed from a mobile phone, tablet or even computer. Our Student Life Application should take note of the intuitive and simplistic design of the My Study Life application, and the ease of use it offers with the cross-platform user experience.

"Canvas" is a system from Infrastructure that is designed to facilitate the connection between students and teachers. It features most of the elements that our My Student Life application intends and has an extremely easy to use interface. Our My Student Life Application should focus on the aspects of Canvas that make it so successful, such as the intuitive user experience, and should take care to not to closely resemble a product like Canvas as it has far more features than our application intends.

Software Requirements, Third Edition by Karl Wiegers and Jo Beatty is one of the most helpful resources that I had access to during our project. It has helped with most aspects of the group project and contains a vast repository of information related to requirements engineering and the process behind delivering a successful product. The textbook was most helpful for understanding the intent behind many of the assignments throughout the semester, and what information to concentrate on which would benefit our project as it progressed. I found it particularly useful during the Requirements Elicitation phase, as it assisted with choosing the most effective method to obtain requirements from stakeholders.

One of the aspects of the project that I struggled with was the UML diagrams. The Tallyfy Article on *All you need to know about UML Diagrams: Types and 5+ Examples* was extremely helpful in clarifying any questions I had during the diagram creation. I found the examples featured to be helpful as I worked on the diagrams with my group, and particularly

appreciated the insight about how the diagrams were supposed to benefit the project, as it gave useful context to what we were creating, and the intent behind it.

Another aspect that I struggled with initially was the Use Case Diagrams. The Medium.com *All you Need to Know About Use Case Modelling* article was an easily digestible extra source of information and understanding during the modelling process. Like with the UML Diagrams, I found the source particularly useful due to its explanation of what the intent was with the diagrams, and how they are supposed to be useful in the future. This was invaluable, as it demonstrates not only how to create a diagram, but why the diagram is designed that way, and how it would be useful in the future.

Spencer Edwards
Section 4A:
https://www.latrobe.edu.au/students/study-resources/study-tools/mylatrobe-app
This source helped me gain a greater understanding of the student life project domain. It provided me with a variety of features I can expect the system to have. This was useful for the meeting with the client, as it allowed me to better prepare questions that clarify the main features of the system.

https://www.uow.edu.au/student/life/myuow/
My initial thoughts of the initiative of the app was to integrate all systems and activities a student would participate in, in a single place. This source developed my perception of the goal of the app – it should have a greater focus on the student life revolving around the university, not necessarily how the students interact with the university.

https://www.uwa.edu.au/students/campus-life/uwa-app
https://www.mystudylife.com/
These sources show different apps in the same project domain. I used them to study and compare the features that they had with the main features our group identified. This included events, chats, booking and scheduling features, helping me gain an in-depth understanding of the use cases from the others in my group.

https://requirements.com/Content/What-is/what-is-requirements-elicitation
This source along with the course content increased the breadth of my knowledge on requirement elicitation techniques and helped me decide what techniques would be best for the project domain in individual exercise 2. It also assisted in the creation of the requirement elicitation plan for project group task 1.

https://babokpage.wordpress.com/techniques/interface-analysis/
This source outlines the purpose, description, and elements of an interface analysis. Our group chose this technique for our elicitation plan, so I found it advantageous to learn more about it. This knowledge helped in my considerations of the technique throughout the writing of the planned

techniques, schedule and resource estimates, documents and systems needed, expected product and risks of using this technique.

https://reqexperts.com/resources/requirements-articles/articles-what-is-the-difference/#:~:text=A%20Requirement%20Specification%20is%20a,and%20maintenance%20of%20the%20product.
This source helped me greatly when specifying the requirements of the student life system. It outlines the key differences between a requirement and a requirement specification and gave examples of the analysis of a couple of use cases to result in potential requirements.

https://www.perforce.com/blog/alm/how-write-software-requirements-specification-srs-document
This source was helpful for creating the software requirements specification document. It provides a template with information and considerations to make about each section of the document. This specifically helped in the external interface requirements, there are multiple types of interfaces including: user, hardware, software and communications.

Hubert Au

Section 4A:

The most common approach for this kind of application appears to be a proprietary system that is specific to a given institution. Institutions that use this type of system include LaTrobe University[1], The University of Wollongong[2], James Cook University[3], The University of Melbourne[4] and The University of Western Australia[5]. The most common features available in each of these systems are campus maps, class timetables (with time and location), information about events and activities happening on campus, connection with others (typically not student-to-student – the most common forms found were student-to-advisor and student-to-security) and library access. Some apps also offered access to one's university email, the booking of appointments, or multiple profiles depending on one's user class. Integration with a desktop client was a feature that also appeared; notably, the University of Melbourne's mobile app is read-only, with any modifications needing to be done through the associated desktop client. Looking over the reviews on the app stores, scores tend to be middling, with common complaints being issues with the user interface.

An alternative solution is to have a general app for campus life that is not geared towards a specific institution. The only example of this that could be found was My Study Life[6]. Features include timetabling, task lists and notifications for incomplete tasks. Reviews were generally positive, although noted that the app was rather barebones. The main issue with this style of system is the lack of depth; a general system will not be able to have the level of detail an app associated directly with a given institution for events, maps etc.

From the beginning, a survey or questionnaire was thought to be one of the more appropriate forms of requirements elicitation to be used for this project. As the project was to be used by the student body, a large and diverse group, it was thought to be necessary to gather opinions from as many of them as possible in a form that could be mathematically analysed along demographic lines, something that surveys are particularly well suited to[7]. The use of an

online survey was chosen because of its easy implementation, lack of cost and high scalability to accommodate the large student body[8].

Regarding the requirements engineering process in general, factors such as time allocation for the requirements engineering phase and ensuring that knowledge is shared between engineers and stakeholders[9] was not very relevant to this project, due to the lack of an actual product being developed and stakeholders, respectively.

[1] LaTrobe University, 2020. Mylatrobe App. [online] Latrobe.edu.au. Available at: <https://www.latrobe.edu.au/students/study-resources/study-tools/mylatrobe-app> [Accessed 2 April 2020].

[2] Uow.edu.au. 2020. *Myuow - University Of Wollongong – UOW*. [online] Available at: <https://www.uow.edu.au/student/life/myuow/> [Accessed 2 April 2020].

[3] Jcu.edu.au.2020.*JCUApp*.[online]Available at: <https://www.jcu.edu.au/new-students/accept-and-enrol/jcu-app> [Accessed 2 April 2020].

[4] Ask.unimelb.edu.au. 2020. *Ask.Unimelb : My.Unimelb Mobile App Features*. [online] Available at: <https://ask.unimelb.edu.au/app/answers/detail/a_id/6012> [Accessed 2 April 2020].

[5] Uwa.edu.au. 2020. *UWA App*. [online] Available at: <https://www.uwa.edu.au/students/campus-life/uwa-app> [Accessed 2 April 2020].

[6] Au.reachout.com. 2020. *My Study Life*. [online] Available at: <https://au.reachout.com/tools-and-apps/my-study-life> [Accessed 2 April 2020].

[7] Grigsby, E., 2007. *Analyzing Politics*. Belmont, CA, United States: Cengage Learning, Inc, pp.27-31.

[8] QRA Corp. 2019. *Requirement Elicitation Techniques: A Step-By-Step Guide - QRA Corp*. [online] Available at: <https://qracorp.com/requirement-elicitation-techniques/> [Accessed 14 June 2020].

[9] L. Liu, T. Li and F. Peng, "Why Requirements Engineering Fails: A Survey Report from China," *2010 18th IEEE International Requirements Engineering Conference*, Sydney, NSW, 2010, pp. 317-322.

Sarah Telford

Section 4A:

Through analysing the application My Study Life, we gained a better understand the domain requirements of a student life application, as well as, confirmed some of our applications requirements. Many of the features of this application closely relate to the requirements of our own, including the ability to create task/challenges and to create calendars for planning academic and personal life. Thus, we were able to compare our application to My Study Life to confirm requirements, for instance, students shall be able to receive and accept event invites from other users and that students shall be able to set a reminder for an upcoming event. Furthermore, by analysing this application I was able to better understand how different pages/tabs within the application interact and interface with one another. For instance, navigating to the calendar page from the home screen, reminders page, task page, etc. Finally, by analysing this application the following question arose: Does the application need to sync across multiple devices? What is the best way for reminders to be set? Does the calendar sync with external calendar applications?

By analysing another student life application, 'Unilife', we were able to establish a requirement regarding the formation of individual and group communication channels. Unilife allows for students to directly message peers, lecturers, teaching assistants and class groups. This made us consider different methods of communication, such as chatrooms, forums, public blogs, etc. However, we decided that having a direct messaging system would be the best option few a few reasons because some individuals will not feel comfortable posting to public forums, and it will prevent people from being flooded with irrelevant information. Furthermore, private messaging allows for students to talk to tutors, lectures and co-ordinator and other academic staff directly. Some questions that arose from analysing this application are: How will users find each other with the communication channel? How can users form group chats? Do users need to accept group chat invitations?

'Canvas' is another type of student life application that enables a connection between teaching staff and students. The platform allows for teachers to directly contact student groups, i.e. student taking Mathematics 1A, and allows students to manage all their academic life, such as watching lectures, submitting assignments, contacting staff and viewing grades. This platform helped us to develop requirements regarding what features our application will require to meet the academic needs of the students. Some questions that arose are: How can we implement personal life organisation? Do we want to recreate all these features or just interface our application with canvas?

Damon Evans

Section 4A:

Harvard University houses a similar application to that of the one described in the domain description for student life. The application allows users such as students, faculty, staff, visitors and neighbours to interact with Harvard's campus community. Although not entirely based around students some features are similar to that has been described in the domain. For example, the Harvard application includes the feature to add events to your calendar, such as a club or extra-curricular event. This is similar to that has been described in the domain in question, as it states 'The app could be also used for a variety of other activities that take place in the context of student life such as organizing events.

Other applications have been made where the student can be able to have an interactive schedule such as MyStudyLife. In comparison to the domain outlined, MyStudyLife is a watered-down version of what is wanted. It contains features such as being able to edit your timetable/schedule, making a to-do list and also having reminders for upcoming classes and exams. While there are limited apps out there similar to ways into the one described it's clear there are little if not any applications that are very similar to the one described.

Canvas which is used by the University of Adelaide shares some aspects as well that has been outlined in the domain description. The infrastructure includes features which our own

application would highly benefit from including to further enhance the overall user experience and useability from the application.

The main resource that I personally found to help the most was the lecture content as I could easily use it and relate the material covered to the project. Each week's lecture helped me immensely in working through each project task. Another resource that was helpful during the project work was the "PearlsFromSand" example paper. Using this example proved useful when working on the group project for student life.

Appendix E: Individual Task 4B

James March
Section 4B:

Our first group assignment was the Requirements Elicitation Plan. This focused on how to acquire requirements for our project, and the processes that lead up to that stage. Here we had to decide on the appropriate techniques to diagnose the requirements for our system.

The initial content of the course consisted of understanding requirements elicitation techniques, and the types of requirements that we would be finding. As the two topic names would suggest, the group assignment and this aspect of the course were closely related. I found the lecture content to be particularly useful for this section of the course. The lecture discussed the pros and cons of the different methods for requirement elicitation, and their appropriateness as a method depending on the project and its goal. Further reading in the textbook further cemented which techniques would be the most effective, depending on the situation, and I found this information particularly helpful. I feel that it will be of great use during future projects, and an understanding of why particular techniques are more effective will be of great benefit.

The second group assignment, the requirements analysis, closely linked to the individual assignment, where we had to create two use cases for our system. This, of course, tied into the course content that also focused on requirements analysis. The course heavily indicated that this process was important in any project, as it is the process that prevents issues down the line. The process involved taking high-level requirements and breaking them down into appropriate amounts of detail, which could then be turned into functional requirements with appropriate levels of prioritisation. So, our individual assignment where we constructed two use cases, were our high-level requirements, and then the group project task, where we then broke the use cases down, was the requirements analysis process.

The ability to translate from discussion into useful and tangible requirements will be an invaluable skill in the software engineering field as it is what allows an engineer to create a product that isn't merely what the client wants, but more correctly what a client needs.

Part three of our group project concentrated on requirement specification. This process involved taking all previous information, such as quality attributes, functional requirements, external interfaces, and constraints, and then simplifying them into more specific and useful documentation. Translating this raw information into useful and workable material will be an invaluable skill in the future work force as a software engineer.

Tied to this part of our group project is the requirements review individual task, which essentially is the requirements validation. Here we took our document and analysed its appropriateness and usefulness to the potential engineer. I feel that this is an invaluable process that checks for the feasibility of the previous work and potentially catches issues.

The final action before creating our final report was the requirements modelling stage. This process involved taking our requirements and constructing a use case diagram and class diagrams. This aspect of the project aims to further validate requirements and lead into the creating the software. The use case diagram can be used as a potential prototype for the client, and the class diagram a basis for the software design. Use case diagrams and class diagrams are an invaluable skill for software engineers, as they allow for a visual display of aspects of the system that previously were previously obfuscated.

Spencer Edwards

Section 4B:

Requirements Elicitation

This phase of the project included creating a requirements elicitation plan and use cases of the system. Learning about the requirements elicitation techniques at an in-depth level prior to creating the elicitation plan allowed our group to make informed decisions on the techniques that would best suit the project. While eliciting requirements is not what I want to do in my job, having this skill is important in getting a job, as it shows employers that you know how to communicate effectively with clients to accurately articulate what they want. Also, reaching a consensus with stakeholders and gathering a cohesive and complete set of requirements is crucial, especially so in a large project.

Requirements Analysis and Specification

This phase of the project included analysing the use cases to extract functional requirements from them and then specifying these requirements. The requirements analysis process taught in this course was crucial for this phase. The project provided us an opportunity to translate the use cases into functional requirements at varying detail. By learning requirements writing methods like Rupp's template, we were able to write our requirements in a way that minimises the chances of incomplete and distorted requirements. It also allowed our group to work individually to produce requirements that were consistent with the across the document, without the need to consult each other all the time. In a large-scale project, this skill would allow a software engineer/developer to stand out. It could prove to be very helpful to the company as they would not need to specifically hire analysts. Our knowledge on non-functional requirements and quality attributes was put in use

when specifying the requirements and how they will impose on the design and implementation of the system.

Final Report
This phase of the project included addressing the feedback received from previous tasks such as use cases or models. A key element we missed and hence addressed, was giving the quality requirements a measure on their success.

Hubert Au
Section 4B:
Requirements Elicitation Plan

Discussing the various forms of requirements elicitation felt like an extension on what I've already learnt about experimental methods, and thus felt natural. Building a deeper understanding of methods such as surveys and interviews allowed us to take a more thoughtful approach towards the selection of our requirements elicitation techniques. Furthermore, as it is rather fundamental and general, the knowledge obtained here will be easily applicable in future.

Requirements Analysis

The primary form of requirements analysis that was performed during this project was turning the project summary into functional requirements. Much of this is discussed below, under the requirements specification heading; other aspects, such as the implementation priorities, were a minor aspect of the project, due to the lack of actual product being made.

Requirements Specification

Learning about the specifics of the grammar used in requirements specification was interesting. Although it's not a writing style I would employ regularly, exploring how one can strip away extraneous words and structure sentences to attempt to convey meaning as bluntly as possible was a useful look at the way language is used. Although I found some issues with the material provided (odd pronoun usage, for instance, which may have just been a translation quirk) the idea itself was solid. The requirements specification section of the course was also one that received quite a bit of focus, so it feels like it was important. The importance of communication cannot be understated, so I can understand why this topic was given focus.

Requirements Modelling

Requirements modelling was not given particular focus during the course. It comprised only one task, and one section of the report. The two documents given in the task provided a decent starting point, and working as a group we were able to put together what we believed to be an adequate representation of our system, although upon receiving feedback, we could

see that it needed improvement. As requirements modelling is presumably an important part of a project's lifecycle, being introduced to these ideas will be useful for working on actual projects.

Final Project Report

I will use this section to talk about the report writing process as a whole. Of course, in industry, no one works alone, so having this project be a group task makes perfect sense. Furthermore, I basically didn't know anyone in this course, so being put in a group to work with mostly strangers is probably a pretty accurate reflection of starting a new job and being assigned to a team. The lack of face-to-face communication is probably non-standard, and presented some challenges. Text communication will likely never be as fast as talking, so communication as a group was slower. However, we were generally able to overcome communication issues, and we were able to complete all tasks on time, with more-or-less equal contributions from everyone.

Damon Evans

Section 4B:

Requirements Elicitation

For the project, a requirements elicitation plan was made which included the creation of use cases. Having learned about the techniques for requirement elicitation helped the process of when it had to be undertaken. This was done ranging from extracting the requirements out of the domain description of student life and questioning the 'client' about what they expect the system to do and what they would like. Developing an understanding of other techniques on how requirements are elicited was also insightful for future practices.

Requirement Analysis

Analysing requirements properly was another important method learned and implemented in the project. The main objective which involved requirement analysis in the conversion of the project outline into functional and non-functional requirements. Another factor of analysing the requirements that were learnt was the proper language when writing them out so that they are coherent and concise.

Final Report

The final report was a culmination of everything that was learned through the course and then implemented into a professional like report. While the report might not quite be at the level that is found in the industry it is a good first stepping stone in the understanding of how software requirements are written in a professional environment utilising the skills and techniques learned.

Sarah Telford

Section 4B:

Our first task was to develop a requirements elicitation plan. This stage helped me to establish an understanding of the different elicitation techniques and their benefits and limitations. Furthermore, I learnt that a range of elicitation techniques need to be performed in ordered to maximise the identification of both external and internal requirements. I believe that this skill will be helpful in the future because project requirements must be properly and effectively elicited to reduce the project's cost and time spent eliciting requirements. Furthermore, many of the elicitation techniques require leadership, communication and organisation skills, therefore, future employers will be more likely the hire someone elicitation skills as they are well versed.

The requirements analysis and specification are key stages that must be undertaken to limit the occurrence of issues later in the project. Requirements analysis emphasises the importance of breaking high-level requirements down into detailed lower-level requirements. This was done through the use case task, in this task, we had to identify two high-level requirements for our projects, such as the application must have a calendar and creating a priority of the requirements. Requirement specification refers to breaking these requirements down into smaller detailed functional requirement, to ensure that the developers have all the information they need regarding the requirement. Having the knowledge and practice in this field will help me in the workforce because customers do not necessarily know the functional requirements of their product and will often provide developers with a high-level requirement. Therefore, the developer must know how to break these requirements down into smaller workable ones.

Next, we focused on reviewing our requirements. It is important to review the work done so far to ensure that you are meeting the goals and needs of the systems and customer. In this instance, we reviewed the completeness, correctness and quality attributes of each requirement. Being able to critically access and analyse your work is an important skill and without doing so important needs may not be getting met, meaning the overall cost and time of the project will increase.

The next phase focused on requirements modelling. This phase is about creating a visual representation of the interactions between the systems internal functionality and users/external interfaces. In our project, we utilised use case diagrams and class diagrams. These diagrams help to see the interactions and dependencies of the system, for example, the user must be logged in to the application to access the communication channels. Furthermore, these diagrams can be used as a prototype for the system and are a great way of showing customers the progress of the project and the inner working of the system, in a simple but effective manner.