RMIT UNIVERSITY

Technical Solution Design
for
< Custom Study Plan Generator >

Version: V1.0
Date: <16 November 2015>
Sponsor: <Santha Sumanasekara>
Author: <Michael Baggott, Jennifer Dousset, Bret Thomas, Duane McMhaon>

Document Control

Distribution

Version	Issued	Recipient	Position
V <1.0>	<23 November 2015>	<Dr Santha Sumanasekara >	<Deputy Head (Learning & Teaching) of the RMIT School of CS & IT>

Amendment History

Section	Page	Version	Comment
<1 - 11>	<1-22>	<1.0 >	<Initial document creation>

.

Staff or Entities Consulted

NAME	Position / Organization
<Dr Santha Sumanasekara>	<Deputy Head (Learning & Teaching) of the RMIT School of CS & IT>

Related Documents

Name	Author	Description
Website Address	Whole Team	https://ec2-52-11-206-174.us-west-2.compute.amazonaws.com/

Preface

The purpose of this document is to detail the technical specifications, functional specifications, design and architecture of the Custom Study Plan Generator project, and will serve as a technical brief for the team, the sponsor, and the project manager.

Known risks and issues have been detailed, along with instruction for deployment.

Table of Contents

1 Introduction

The Custom Study Plan Generator project is a web based interface that will allow a staff member at RMIT to create and manage study plans for students enrolling at RMIT. The generator caters specifically for full time study at 4 units per semester, and has been tested with four supplied courses obtained from the project sponsor, but is extensible for any course up to a maximum of 16 semesters.

The technical environment for the project consists of a .NET application, with an external Microsoft SQL Server database, and includes the use of Javascript and jQuery. The estimated level of complexity is moderate to high, with use of a front end and a backend for extensibility purposes.The benefit of the web application is to save a lot of manual work for the selection officer in creating a custom study plan for a student, and creates a fast way to achieve the task, with a user friendly, graphical UI, and a simple backend form maintaining the database.

2 Technical Environment

Technologies include:

Microsoft .NET/C# - chosen for its developer friendly and mature approach to enterprise web design, with good inbuilt security, extensive support for external APIs, and good support by Microsoft for the IDE Visual Studio, which is a high end developer environment.

Javascript/jQuery - Used to support the Microsoft front end cshtml pages, makes client side operations a lot more manageable and through the use of Ajax is able to communicate with the back end to provide an advanced user interface

Microsoft SQL Server - Chosen as an excellent complimentary database to .NET, with good access by the language (C#) through the use of Entity Framework, enables a database first approach to creating a model for C# to work with.

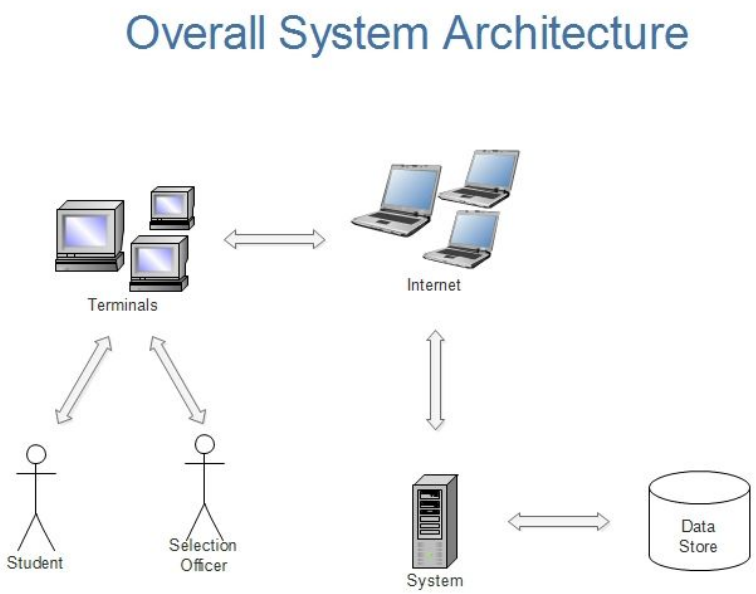
BitBucket Version Control - Used by the team for version control, has allowed the creation of separate development and production repositories, has enabled the team to work simultaneously on the project, and contains a complete history of all changes/updates of the repository.

Owin Security - Used in CAS authentication

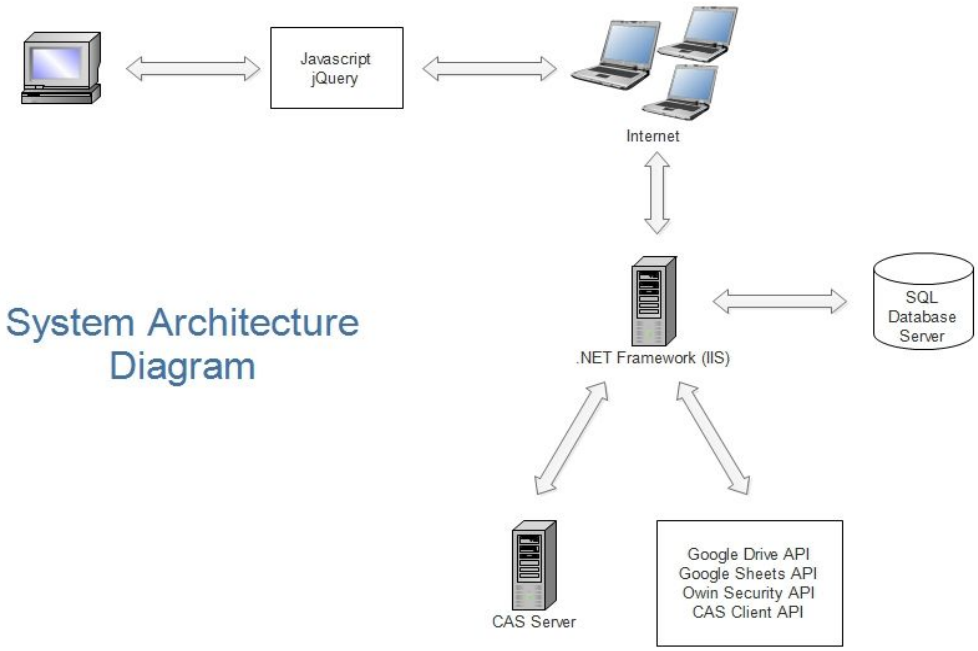
Java Enterprise Edition CAS Server - Client access provided in the source code, but not operational in the final product.

.NET CAS Client - Used to give the client access to the CAS server for authentication

3 Overall Architecture



4 System Architecture



System Architecture Diagram

.NET Framework (IIS) version 4.5.1 - The Internet Information Server where the .NET application is stored. This is Microsoft Server running .NET framework version 4.5.1.

CAS Server (Central Authentication Server) This is served on a Tomcat server.

SQL Database Server - SQL Server Express 2012 was installed on the MS Server 2012 instance.

API stack - Each .NET external API, loaded as external packages into the .NET Application. Google Drive provides the upload/share feature to Google Drive, Google Sheets provides the ability to format the uploaded plan, Owin Security and CAS Client both provide the ability for the user to log on to a CAS server.

Internet - The application is only accessible through the Internet (Once project is completed).

Javascript/jQuery - Used extensively for client side operations such as validation, the User Interface, Error messages.

4.1 FUNCTIONALITIES/FEATURES

4.1.1 System Administrator logs into system using a non-RMIT Google account, and sets up Courses, Units, and Prerequisites as required for RMIT courses.

4.1.2 System Administrator then configures a “Default Plan” for each course

4.1.3 Selection officer logs into system using a non-RMIT Google account.

4.1.4 Selection officer adds student to the system, then creates a new plan, by entering a student number and selecting Create Plan.

4.1.5 There will be a warning if a plan currently exist for the student, and if it does, the Create Plan process may be cancelled, and the current plan edited in the “Edit” mode.

4.1.6 Selection officer selects a course, and signifies if the course is a mid-year intake.

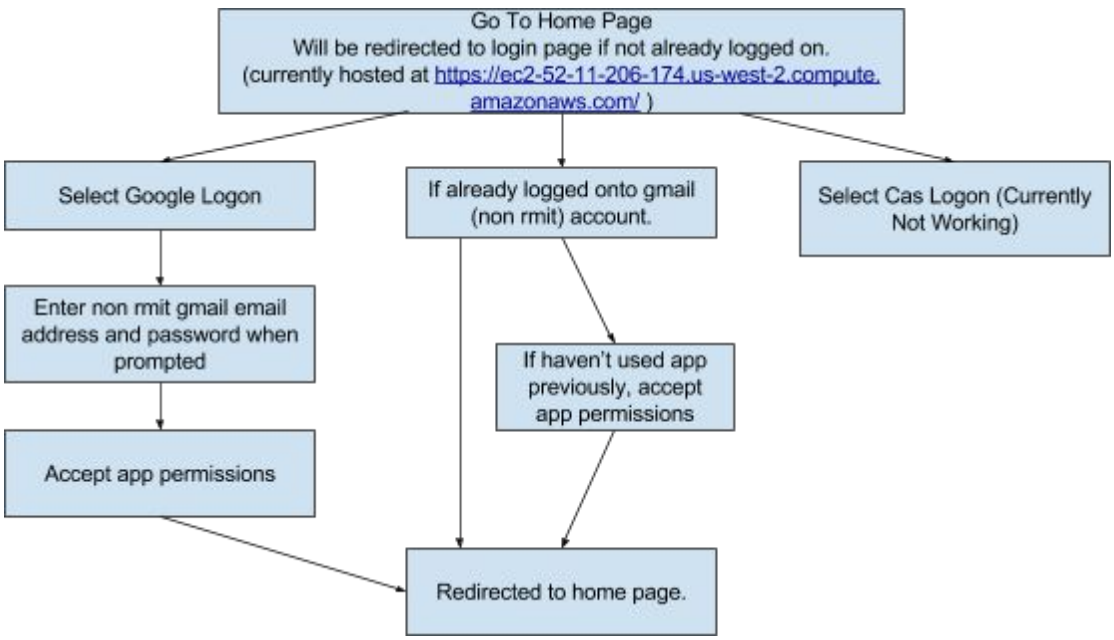
4.1.7 Selection officer denotes exemptions, then continues. this decision may be reversed using the back button in the application.

4.1.8 The system runs its algorithm, reshuffling the units to a best fit, which then may be manually modified by the officer. It is also possible to check for any prerequisite violations at this stage. Note that units that have been exempted will not violate prerequisite requirements.

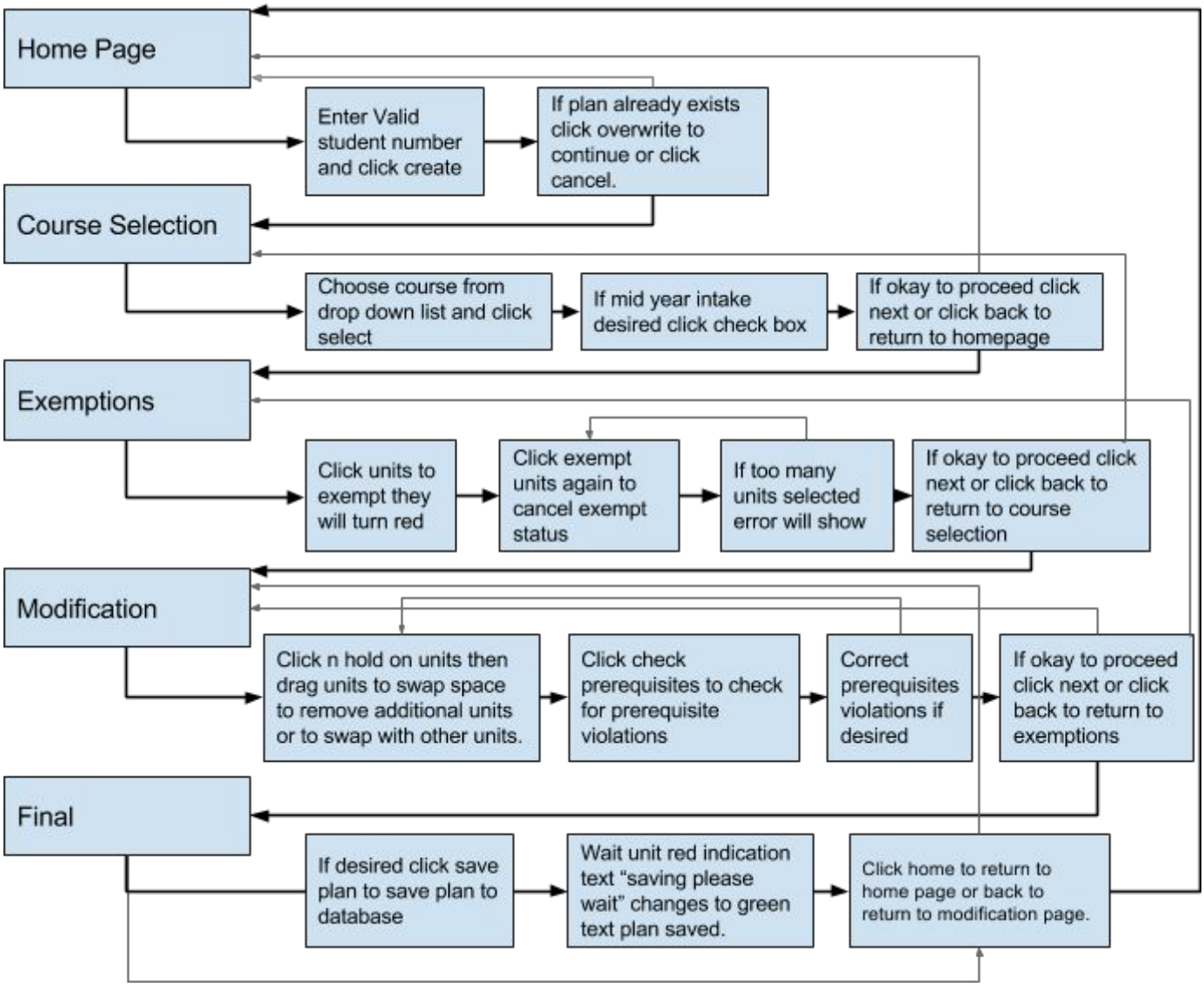
4.1.9 Office proceeds to the final page where the final plan is presented. Again the officer may use the back buttons to reverse any of the decisions so far and make adjustments. When the officer is satisfied with the plan, the “Save Plan” button is pressed, and once the plan is saved it may be uploaded to Google Drive for the student using the Upload To Drive button in the top navigation menu. Once completed, the user may return home.

4.1.10 The most recent saved plan may be accessed from the Home menu using the “Edit” button. Once the plan is uploaded, the officer may access it for viewing in their Google Drive, under the Custom Study Plan folder. Note due to Google Sheets limitations, the plan is accessible on the second “Tab” of the Google Sheet, labelled “Study Plan”

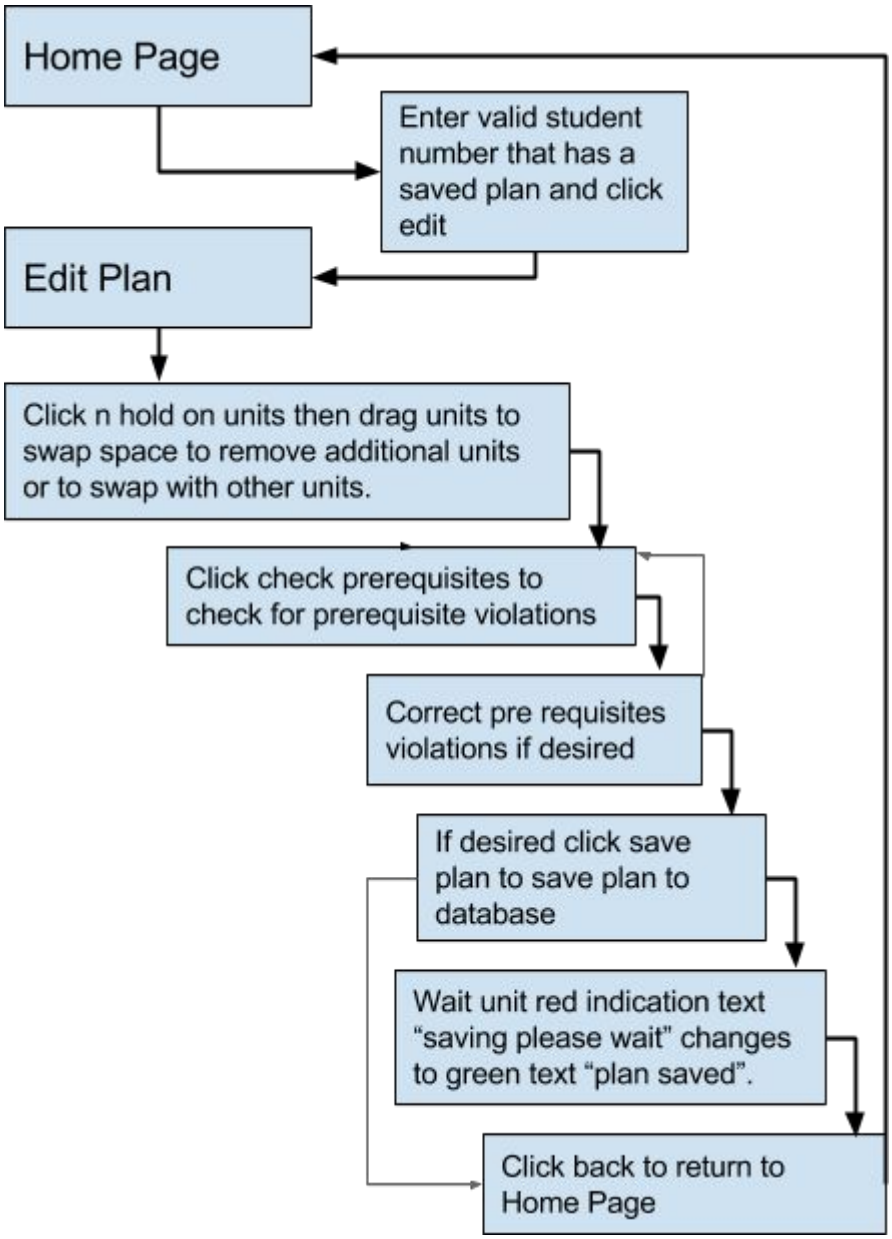
4.2.1 Logon To System Flow Diagram



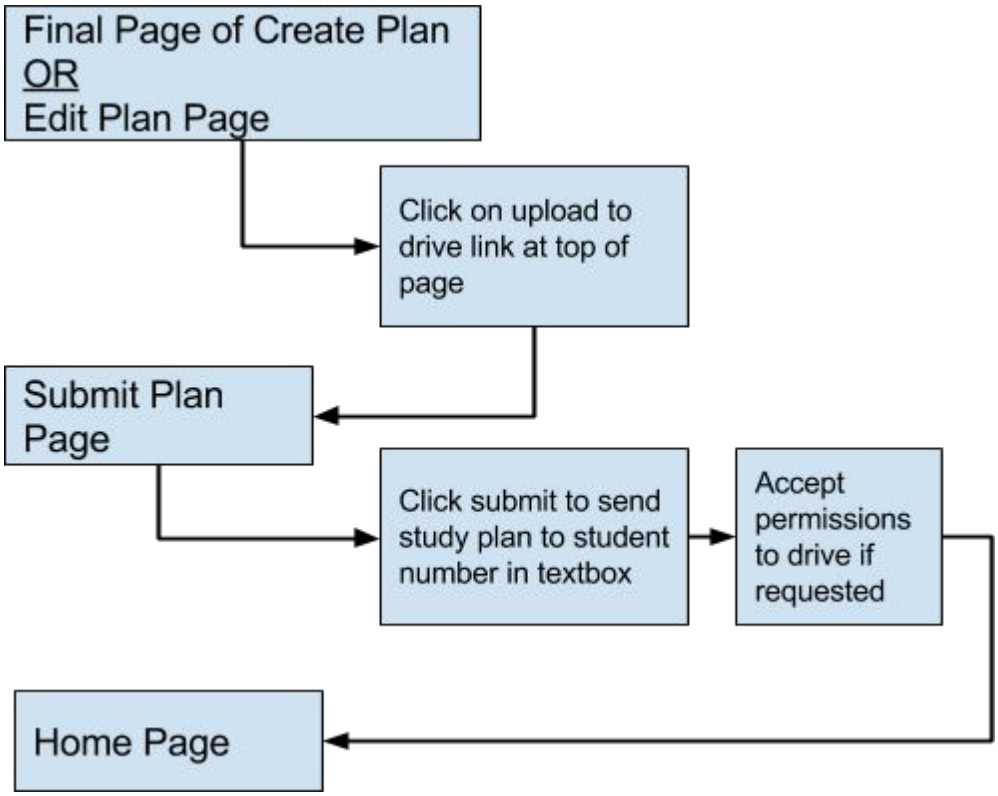
4.2.2 Create Study Plan Flow Diagram



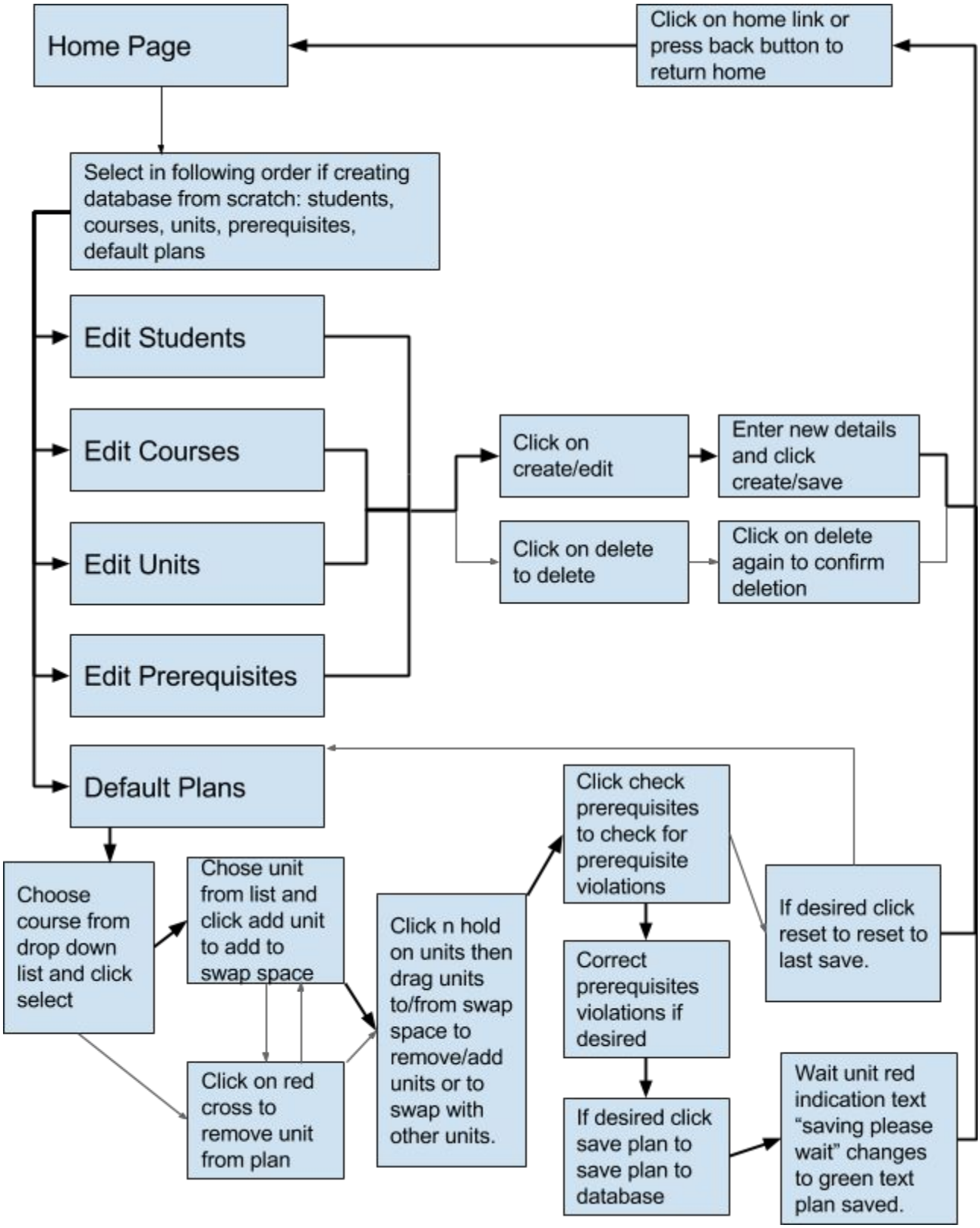
4.2.3 Edit Study Plan Flow Diagram



4.2.4 Upload Study Plan to Student, Flow Diagram

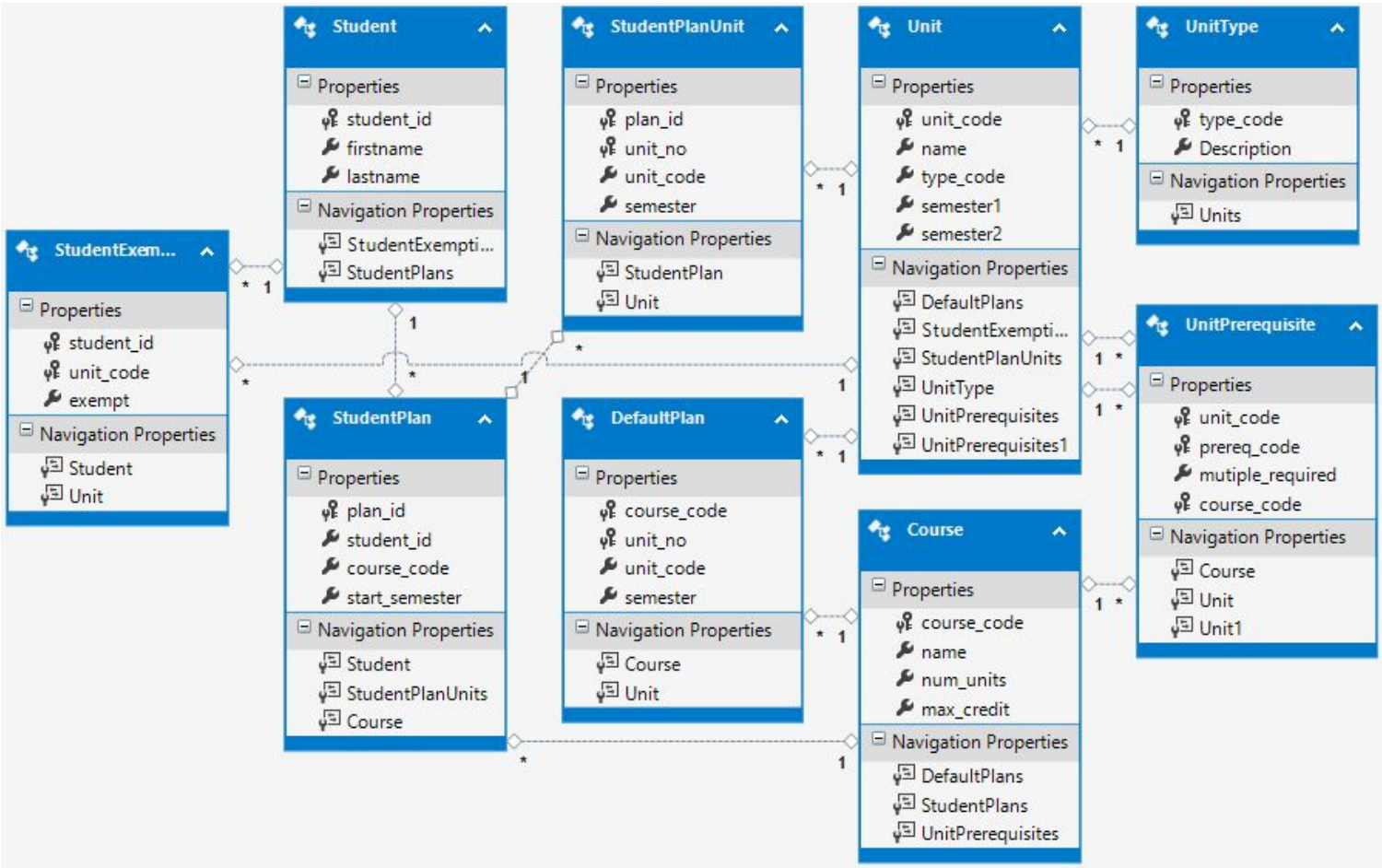


4.2.5 Data Entry for Students, Courses, Units, Prerequisites, and Default Plans Flow Diagram



5 Database Architecture

5.1 ENTITY RELATIONSHIP DIAGRAM



5.2 DATABASE DESIGN

The structure of the database is primarily designed around Courses and Units. As there are multiple courses, and Units can be part of more than one Course, these items are stored as completely separate entities. Tables for Study Plans then associate Courses with their respective units, with foreign keys to link back to these tables.

Course information includes the course code, which is used for the primary key and as a foreign key to other tables, along with the course name and the number of units in the course. There is also a value max_credit which is to indicate the maximum number of exemptions a student can be given for the course. This value is used in the application for validation on the Exemptions page, so that the user cannot select more exemptions that the amount allowed. Whilst the standard limit is 16 units for a 24 unit/3 year course and 24 units for a 32 unit/4 year course, it was opted to include this value as part of the course data rather than hard coding it in the application. This is to allow for greater flexibility and ease of updating to meet any future changes to the exemptions/credit transfer rules, or to allow for any other differing requirements for other courses.

Unit data contains the unit code as the primary key, along with the unit name, and two booleans representing its availability for Semester 1 and Semester 2. There is also a type code to represent the type of unit (e.g. Core, IT Elective, Minor, etc), which has a link to another table with the definition of the code. The Unit Type did not end up getting used in the final implementation of the algorithm, however this structure has been left in place in the database for future use if desired.

The Unit Prerequisite table links Units to their prerequisite Units. The algorithm checks this data when creating a custom plan to ensure units with prerequisites are scheduled in the correct order and all requirements are met. The table allows for one unit to have multiple prerequisites as some units have more than one requirement, along with a boolean to indicate whether all the listed prerequisites are mandatory or whether it only needs to be at least one of them met. A Course is also associated with each entry in this table as part of its primary key, because some unit prerequisites are different depending on the course. E.g. “COSC2413 Web Programming” has different prerequisites when in the Bachelor of Computer Science course than it does in the Bachelor of Information Technology. So the database has been designed to allow for these differences.

To get the best possible performance for units that span multiple sessions, it is recommended to create Multiple units with similar names, eg Project Management (Part 1), Project Management (Part 2) … etc, with codes such as COSC1111-1, COSC111-2 … etc. Then assign prerequisites for the block to the Part 1 unit, and assign Part 1 as a prerequisite of Part 2, and so on. Note it is possible for these units to be split by the algorithm, so some manual intervention may be required when creating the plan.

The Default Plan table contains the standard study plan or path for each course in the system, along with the units in that course and the semester that each unit is normally scheduled for in the standard path. This data forms the basis for the system and is used by the Custom Study Plan Generator algorithm to generate a custom plan based on the standard units required to complete each course.

Student Plan table is used to store a record of each student plan that gets generated by the application and saved by the user. It contains a primary key called Plan_ID and associates a Student number to the Course they are having a plan created for, with foreign keys linking to the Student table and Course table. Another table Student Plan Units is used to record the Units in the custom study plan and the semester the application has scheduled them for. It has a primary key of the Plan_ID and unit’s position in the Study Plan. This separates out the Plan from its Unit data to prevent repetition of values such as the Course code and the Student number.

5.3 DATABASE SCALABILITY

The database has been designed to be very scalable, with the possibility of future expansion taken into consideration in the design process.

Table design for fields pertaining to Courses and Units has been kept very generic, such that the fields are not specific to just the School of Computer Science and Information Technology and its courses. This allows for the potential of expansion to use in other courses from other departments within the university easily.

Microsoft SQL Server was chosen for the database as it is a high-performance enterprise product with very high capacity. This database infrastructure is very scalable and will have the capacity to fully support any future expansion of the project.

6 Implementation Instructions

Required: Microsoft based web hosting to host the application (Microsoft IIS) running .NET Framework 4.5.1. Microsoft SQL Server hosting. The SQL server setup scripts will be supplied in the Visual Studio project and may be used to create a fresh instance of a SQL database. The SQL server connection string will need to be adjusted in the Visual Studio project, and is accessible in the web.config file under the name “CPT331”

For CAS access, a CAS server running Java Enterprise Edition, that the application can connect to. This is not required as this part of the solution has not been implemented, but the project includes code and an API that will allow connection to a CAS server.

Google Developers’ Console Application entitled “custom-study-plan-generator”, having the following project-specific API’s enabled:

- Drive API (with scopes added for spreadsheet feeds and accessing spreadsheets in Drive)
- Google Apps Script Execution API
- Google+ API

A Google Apps Script was written and bound to the above application to style the Google Spreadsheet.

The following .NET Clients were incorporated into the application to interface with Google services:

- Drive, Script, GData

The free tier products available at Amazon Web Services were used to provide the server-side infrastructure. The following servers were used:

- Ubuntu Server 14.04 (Apache)
- Microsoft Server 2012 (IIS)

Although a single server could have been used there was not enough storage space available on the free Microsoft product to host the CAS server. Many features deployed in this project at this tier of the architecture are not suitable for production and are acknowledged as being suitable for development purposes only, including:

- SSL certificates are self-signed and unverified,
- Region used (Oregon, West-Coast) on AWS, for example, would realistically be located elsewhere (Sydney),
- Redundancy has not been built in to the design.

The project has been designed with a backend that allows the creation of new Courses, Units, Students, and Prerequisites, and as such, no data migration is required. The Units have a property called Unit Type, which has been included for future extensibility and possible refinement of the algorithm, but this property is not required for the web application to function, and as such, no backend access has been given to set up new Unit Type information at this stage.

7 Functional Specifications

- System must allow creation, and editing of Students, Courses, Units, Unit Prerequisites, and Default Plans (Courses), and deleting of these entities when they have not been used elsewhere in the web application (ie. if an error is made upon creation).
- System must allow creation of a customised study plan.
- System must allow study plan to be saved to the database, and uploaded to Google Drive and shared with the student in Create mode.
- System must allow most recent study plan to be edited, then saved and re-uploaded and shared with the student (overwriting previous study plan in both cases).
- System runs a complex algorithm on the study plan during the Create Plan process to determine the best fit plan for the student based on exemptions, unit prerequisites, and intake time.

8 Non-functional Specifications

- System to have high usability and user to interact with the system via a simple clear concise interface.
- Clear simple student plans that show all information necessary for student to understand their study.
- Source code can be easily understood and maintained by others.
- System can be extended to include other courses and or units.
- System to have good reliability and crash free operation.
- System to be reasonably quick to respond to user input.

9 Summary of Test Results

Test ID	Test Case Name	Test Description	Steps	Expected Results
HS01	Home – Student – Valid	Selecting a valid Student ID allows user to proceed to Create Plan or Edit Plan	1. Login. 2. Enter Student ID of a valid student already existing in the system. 3. Click “Create”.	User is successfully taken to the first page of the Create Plan process.
HS02	Home – Student – Invalid	Selecting an invalid Student ID does not allow the user through to Create Plan or Edit Plan.	1. Login. 2. Enter Student ID that does not exist in the system. 3. Click “Create”.	User receives an error message “Student ID does not exist”.
HS03	Home – Create Plan – Overwrite	Selecting a Student ID with an existing plan in the system and clicking Create Plan.	1. Login. 2. Enter Student ID with an existing study plan. 3. Click “Create”.	User receives a popup notification prompting them to confirm if they wish to overwrite the existing plan.
HS04	Home – Edit – Valid Plan	Selecting a Student ID with a valid plan proceeds to the Edit screen.	1. Login. 2. Enter Student ID with an existing study plan. 3. Click “Edit”.	User is successfully take to the Edit Plan page and can make modifications to the existing study plan.
HS05	Home – Edit – No Plan	Selecting a Student ID with no plans and clicking Edit Plan.	1. Login. 2. Enter Student ID with no existing study plan. 3. Click “Edit”.	User receives an error “Plan for this student does not exist”.
CP01	Create – Select – Valid	Selecting a valid course in the first Create Plan screen allows the user to continue to second page.	1. Enter Student ID. 2. Click “Create”. 3. Select a Course and click “Select”. 4. Click “Next”.	User successfully proceeds to Create Plan second page, Exemptions.
CP02	Create – Select – Invalid	Attempting to proceed to second page of Create Plan without a course selected.	1. Enter Student ID. 2. Click “Create”. 3. Click “Next”.	User is directed away from the second page until they select a valid course.
CP03	Create – Exemptions – Valid	Selecting a valid number of Exemptions allows the user to proceed to the next page in Create Plan.	1. Enter Student ID. 2. Click “Create”. 3. Select a Course and click “Select”. 4. Click “Next”. 5. Choose 4 Units for Exemptions. 6. Click “Next”.	Exemptions are recorded and algorithm runs on the remaining 20 units. User is directed to the Modify page displaying the generated Plan.

Test ID	Test Case Name	Test Description	Steps	Expected Results
CP04	Create – Exemptions – Zero/Sem 1	Selecting zero Exemptions for Semester 1 intake skips the algorithm.	1. Enter Student ID. 2. Click “Create”. 3. Select a Course and click “Select”. 4. Click “Next”. 5. Click “Next” with no exemptions selected.	Algorithm is skipped, user is directed to the Modify page displaying the standard Default Plan for the course with no changes.
CP05	Create – Exemptions – Zero/Sem 2	Selecting zero Exemptions for Semester 2 intake runs the algorithm on the Default Plan.	1. Enter Student ID. 2. Click “Create”. 3. Select a Course and click “Select”. 4. Tick “Mid-year Intake”. 5. Click “Next”. 6. Click “Next” with no exemptions selected.	Algorithm runs on the Default Plan to account for any scheduling differences, user is directed to the Modify page displaying the generated Plan.
CP06	Create – Exemptions – Invalid	Selecting too many Exemptions prevents the user from continuing to the next step in Create Plan.	1. Enter Student ID. 2. Click “Create”. 3. Select a 24 unit Course and click “Select”. 4. Click “Next”. 5. Choose 20 Units for Exemptions. 6. Click “Next”.	User receives an error message “Exemption limit exceeded” and is prevented from continuing to the next page until a valid number of Exemptions is selected.
CP07	Create – Modify – Next	User is able to reach the Final page of the Create Plan process with valid selections at each stage.	1. Enter Student ID. 2. Click “Create”. 3. Select a Course and click “Select”. 4. Click “Next”. 5. Choose 4 Units for Exemptions. 6. Click “Next”. 7. Click “Next”.	Study Plan is displayed on the Final page of the Create Plan process.
CP08	Create – Modify – Back	Once reaching the Modify page after the algorithm has run, the user is able to go Back to the Exemptions page, view their previous selections and make changes to Exemptions.	1. Enter Student ID. 2. Click “Create”. 3. Select a Course and click “Select”. 4. Click “Next”. 5. Choose 4 Units for Exemptions. 6. Click “Next”. 7. Click “Back”.	User is directed back to the Exemptions page, with the previously selected Exemptions marked, and they are able to make changes to their selections.

Test ID	Test Case Name	Test Description	Steps	Expected Results
CP09	Create – Modify – Back/Next	Algorithm reruns on updated selections if the user reaches Modify page, clicks Back, makes changes to their selected Exemptions and continues again.	1. Enter Student ID. 2. Click “Create”. 3. Select a Course and click “Select”. 4. Click “Next”. 5. Choose 4 Units for Exemptions. 6. Click “Next”. 7. Click “Back”. 8. Select different Exemptions. 9. Click “Next”.	Algorithm reruns when changes are made on the Exemptions page and the user is presented an updated Study Plan on the Modify page.
CP10	Create – Final – Save	User is able to Save the Plan generated by the algorithm at the end of the Create Plan process.	1. Enter Student ID. 2. Click “Create”. 3. Select a Course and click “Select”. 4. Click “Next”. 5. Choose 4 Units for Exemptions. 6. Click “Next”. 7. Click “Next”. 8. Click “Save Plan”.	The algorithm generated Study Plan is saved to the database.
CP11	Create – Final – Back/Next	Final page Plan is updated to reflect changes if the user reaches the page, then clicks Back to Modify and moves units and clicks Next again.	1. Enter Student ID. 2. Click “Create”. 3. Select a Course and click “Select”. 4. Click “Next”. 5. Choose 4 Units for Exemptions. 6. Click “Next”. 7. Click “Next”. 8. Click “Back”. 9. Move some Units to different positions in the Plan. 10. Click “Next”.	Changes made on the Modify page are updated and reflected when clicking Next and visiting the Final page second time around.
EP01	Edit – Save	Editing an existing Study Plan and clicking Save updates the Plan in the database.	1. Enter Student ID with an existing study plan. 2. Click “Edit”. 3. Move Units to different positions in the Plan. 4. Click “Save Plan”.	Changes made to the Plan are successfully updated/saved to the database.

Test ID	Test Case Name	Test Description	Steps	Expected Results
EP02	Edit – Reset	Making changes to an existing Study Plan then clicking “Reset” will revert any unsaved changes.	1. Enter Student ID with an existing study plan. 2. Click “Edit”. 3. Move Units to different positions in the Plan. 4. Click “Reset Plan”.	Changes made are discarded and the Plan reverts to its original state as per the database records.
EP03	Edit – Back	Making changes to an existing Study Plan then clicking “Back” will discard any unsaved changes.	1. Enter Student ID with an existing study plan. 2. Click “Edit”. 3. Move Units to different positions in the Plan. 4. Click “Back”.	No changes are made to the plan, and the user is returned to the Home screen.
DP01	Default Plan – Save	Editing a Default Plan and clicking Save updates the Default Plan in the database.	1. Click “Default Plans”. 2. Choose a Course and click “Select”. 3. Move Units to different positions in the Plan. 4. Click “Save Plan”.	Changes made to the Default Plan are successfully updated/saved to the database.
DP02	Default Plan – Reset	Making changes to a Default Plan then clicking “Reset” will revert any unsaved changes.	1. Click “Default Plans”. 2. Choose a Course and click “Select”. 3. Move Units to different positions in the Plan. 4. Click “Reset Plan”.	Changes made are discarded and the Default Plan reverts to its original state as per the database records.
DP03	Default Plan – Back	Making changes to a Default Plan then clicking “Back” will discard any unsaved changes.	1. Click “Default Plans”. 2. Choose a Course and click “Select”. 3. Move Units to different positions in the Plan. 4. Click “Back”.	No changes are made to the Default Plan, and the user is returned to the Home screen.

10 Known Issues & Risks

Issues:

- When a user removes the CSPG Application access from the Google Account, and the web app is then re-run and permission is given once more, there will be an error uploading a completed plan to Google Drive. These exception is caught by the system, and the user is prompted to re-save and upload the plan. This problem will only occur on the very first run of the web app, and only if the web app permission has been removed from the users Google account previously.It does not affect new users.

- Due to a lack of online resources and information, we were unable to implement the requirement of a combined CAS/Google authorisation, with delegation. Delegation to Facebook and Yahoo was achieved. Currently any user with a Google account can log in, as there is no CAS authentication. This may need to be modified to be compatible with the RMIT system when deployed. This has been authorised by the sponsor.

- Due to restrictions on the RMIT Google accounts, the web application does not allow a user to upload a plan to Google Drive with an RMIT account, as third party application access is blocked on these accounts.This feature has not been tested with a staff account. Plans are however, able to be uploaded to a student account if the user logs in with a non-RMIT Google account.

- The Google Drive upload has been coded to work with only 6 semesters.This feature will not work fully on courses above 6 semesters. Additionally, this feature has not been programmed to output Exemptions, as per sponsor request in the week 3 sponsor meeting.

- The Bachelor of Software Engineering has units that span multiple sessions. Due to late testing of this course it has been discovered that it is possible for the algorithm to re-arrange the parts of these units so that they are out of order. This bug was discovered very late in the timeline, and may not be fixed for the production version. The user will have to manually intervene and rearrange units in the modify step if this occurs.

- A late bug has been discovered with the live site only, in the form of an intermittent SQL database exception when accessing the site. We believe this may be caused by an authentication cookie timing out and becoming invalid, and the site is unable to handle a bad cookie. Unfortunately the person that worked on this code has been absent from the project since week 5, and we are unable to get him to fix it, and nobody else has been able to fix it with such little time remaining in the project after it was discovered.

Risks:

- As the web app has not been tested extensively by the client, there may be some minor unhandled exceptions that occur. A more thorough client testing is necessary, with a maintenance plan to address any problems.

11 Other Considerations

- RMIT does not currently host any .NET applications, or have a Microsoft Server available for hosting. For deployment this will be required.

12 Appendix