Concordia University

Department of Computer Science and Software Engineering

Software Process
SOEN 341/4 S --- 2016

Project Scope and Plan Document

Team members information	
Name	**SID**
Emili Vasseva	27526741
Sean Marcoux	27511876
Bruce Edouard Brazier	27419562
Dias Marat	27277911
Le Vinh Dang	26844987
Adriel Fabella	27466005
Gabriele Bavaro	27399103
Ying-Chen Chu	27415710
Alex Eladas	27041462
Salma Aly	27176414
Adil Hssaini	24832396
Nicolas Frazer-McKee	27068956

Grading Sheet

Section	Evaluation criteria (see instructions in the template for details)	Grading
all	10 marks are allocated for excellence, professionalism and quality of work above and beyond the correct meeting of specifications..	/10
1	Presentation of this document	/5
2	Completeness and accuracy with regard to initial project description	/1
3.1 . . 3.2 3.3	Completeness and accuracy of the project functional requirements expressed as formal use cases, including difficulty and importance indicators completeness and accuracy of the diagram and description of the domain model completeness and accuracy with regard to initial project description accuracy with regard to initial project description, difficulty and importance ratings	/15 . /3 . /1
4.1	Description of all team members' capacities and schedule restrictions	/1
5	List of goals removed from the project. For each goal removed, give justifications in light of the resources available	/`1
6.1 . 6.2	Clarity of textual description, validity of rationale, clarity and appropriateness of diagram, list of modules responsibilities List of technologies used, validity of rationale	/2 . /1
7.1 . 7.2 . 7.3 . 7.4 7.5 7.6	Completeness of list of activities, clarity of their stated purpose, as well as statement of what artifacts they are producing Completeness of list of artifacts to be produced during the project, validity of roles description of each artifact Cost estimation of each individual artifact, validity of explanation of cost estimation, total cost estimate Mapping of activities to individual project members Accurate and complete presentation of milestones Assessment of risks `	/1 . /2 . /2 . /1 /1 /1
8	Early Prototyping	/2
Total		/50

DO NOT REMOVE THIS PAGE WHEN SUBMITTING YOUR DOCUMENT

2. Project Description

The following document describes the fundamental scope and plan of a group project to be done in the Software engineering course, Software Process (341). The purpose of this project is to create an interactive multi-year schedule builder for software engineering students. In addition, being a team project, it emphasizes on the value of team cooperation and organization. The team name is “the Schedulers” and is composed of 12 students, having specific roles of front-end programmer, back-end programmer and documenters.

The document provides a foundation to follow when implementing the system. It contains the system’s functionality, meaning requirements and behaviors, and the resources and technologies used to achieve the end goal.

The overall goal of this project is to create a system that can generate a schedule. The application should allow a user to log in to the network after signing up. It shall also invite the user to add his/her classes and display his/her preferences, for example: choosing only night classes. The system shall create a schedule, based on the inputs of the user, to be displayed on screen.

3. Goals and Constraints

3.1 FUNCTIONAL REQUIREMENTS

1. **Login**

1.1 The system shall allow users to login using valid login credentials.

2. **Registration**

2.1 A new user shall be able to create an account on the scheduler system.

2.2 A user shall be able to view the default schedule prior to registration

3. **Account Management**

3.1 The system shall allow the user to view all account information within 2 seconds.

3.2 The user may modify mutable account information.

3.3. The system may generate a confirmation message describing account information changes.

3.4 The system shall update the user's database with the account information changes.

3.5 The user shall be able to reset their password.

4. **Record Management**

4.1 Upon first time login, the system shall allow the user the creation of a student record, based on the set of courses previously taken.

4.2 The user shall be able to update and modify the information provided initially as well as any information entered subsequently.

4.3 The system shall store the information entered during the creation or the modification of the record once the student commits to saving.

4.4 The system shall discard all entered information if the user's session is terminated or interrupted prior to committing to save.

4.5 The system shall allow the user to view the saved student record at any time during an active session.

4.6 The user may delete the existing student record at anytime during an active session, and may recreate a new one.

4.7 The system shall allow the existence of one and only one student record per user account.

4.8 The system shall be able to generate a list of the courses taken based on the number of semesters finished by the student.

4.9 The user shall be able to add update their taken classes based on the auto generated list of taken courses.

5. **Course Selection**

5.1 The system shall allow registered and unregistered users alike to view the list of all courses and their respective descriptions and schedules.

5.2 The system shall allow the user to add needed courses for the generation of a schedule.

5.3 The system shall allow the user to remove previously added needed courses.

6. Schedule Preferences

6.1 The system shall allow the user to set his preferences.

6.2 The system shall allow the user to modify his previously set preferences.

7. Schedule Generation

7.1 The system shall allow the user to generate a schedule according to the courses added and or without the preferences

7.2 The system shall allow the user to modify options concerning the generation of the schedule

7.3 The system shall allow the user to save a copy of the generated schedule

7.4 The system shall allow the user to delete saved schedules

7.5 The system shall allow the user to print a generated schedule

7.6 The system shall allow the user to view a generated schedule

7.7 The system shall allow the user to change the section for an added course.

9. Logout

9.1 **The user shall be able to logout from the system and close the current session.**

10. Administrator

10.1 The administrator shall be able to add one or more courses to all programs.

10.2 The administrator shall be able to edit all courses.

10.3 The administrator shall be able to remove one or more courses from all programs.

10.4 The administrator shall be able to add one or more sections to all courses.

10.5 The administrator shall be able to edit all sections.

10.6 The administrator shall be able to delete one or more sections from all courses.

10.7 The system shall save any and all modifications performed by the administrator, only if the commit to changes operation terminates successfully.

10.8 The system shall discard all changes if the administrator's session is terminated unexpectedly.

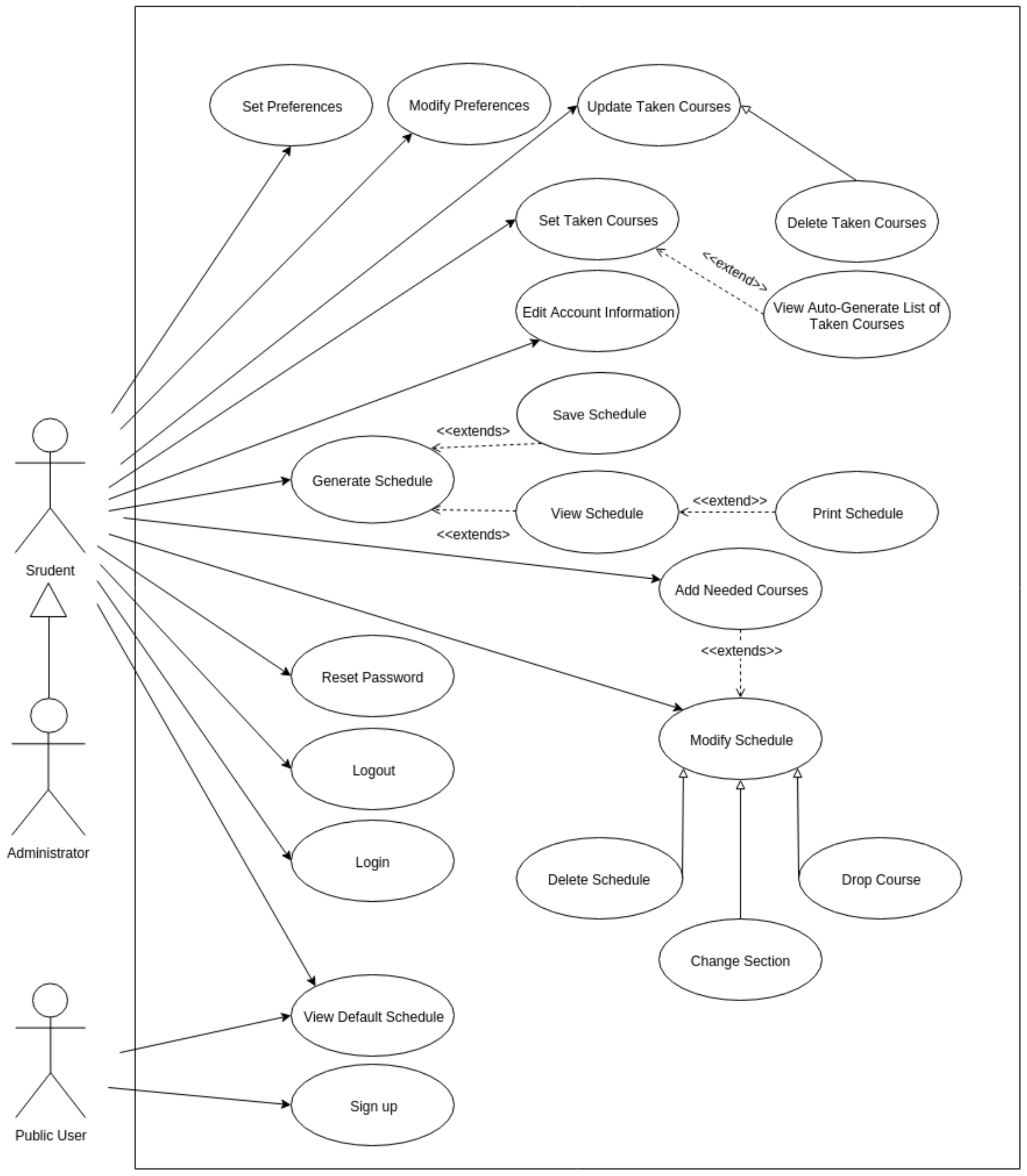


Figure 1. Admi, Student, and Public User UCD

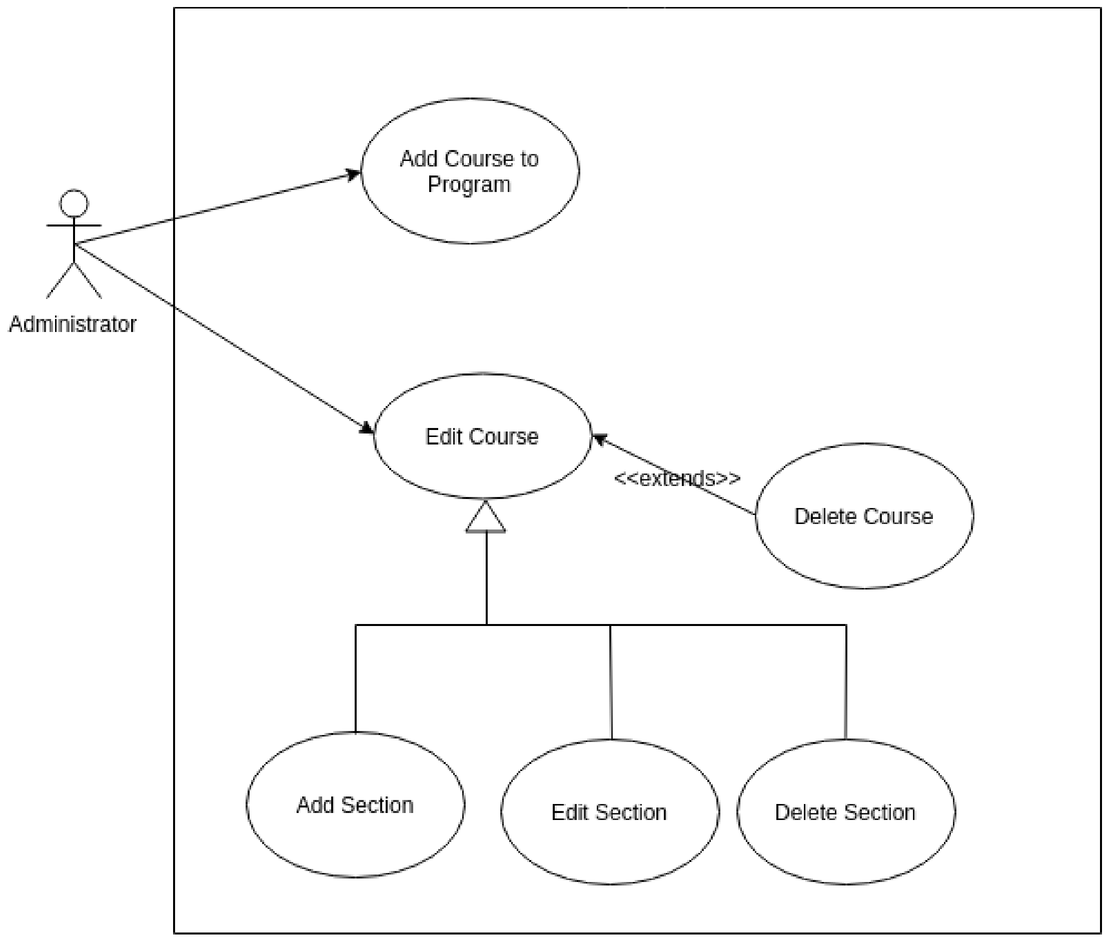


Figure 2: Administrator use case diagram

Name:	Login	Author	Salma Aly
Identifier:	UC1	Version:	1.2
Date Created:	Jan 30, 2016	Last Modified:	Feb 8, 2016
Importance:	5/5		
Actor(s):	Student or Administrator		
Goal:	To access the user's profile in the scheduler system		
Summary:	The user enters their username and password which will be validated by the system in order to give access to the user and retrieve their information.		
Related use-cases:	Logout		
Preconditions	1- The user has successfully accessed the login page 2- The user is not already logged in 3- The user has already signed up (created a profile) on the system.		
Trigger:	User prompts system to validate login information		
Basic Flow:	1-User enters username and password 2- The system verifies the user identity 3-The user accesses their account		
Post-Conditions:	**Success:** The user is logged in and has access to their profile **failure:** The user cannot access their profile and is prompted to login		
Minimum Guarantee:	The user can view the login page		
Risk Assessment:	Low		

Name:	Logout	Author	Salma Aly
Identifier:	UC2	Version:	1.2
Date Created:	Jan 30, 2016	Last Modified:	Feb 8, 2016
Importance:	5/5		
Actor(s):	Student or Administrator		
Goal:	To sign out of the current session		
Summary:	Using the sign-out option the user attempts to sign out and close the current session		
Related use-cases:	Login		
Preconditions	The user is already logged in		
Trigger:	User Selects Signout option		
Basic Flow:	1-The user chooses signout option 2-The user is logged out and the session is terminated 3- The user is redirected to the login page		
Post-Conditions:	**Success:** 1-The user is logged out		

	2- The user is redirected to the login page **Failure**: The user is still logged in
Minimum Guarantee:	The user's last activity is saved
Risk Assessment:	Low

Name:	Reset Password	Author	Salma Aly
Identifier:	UC3	Version:	1.2
Date Created:	Jan 30, 2016	Last Modified:	Feb 8, 2016
Importance:	5/5		
Actor(s):	Student		
Goal:	To reset password		
Summary:	The system sends the user an email with temporary login credentials that allows them to login and access the change password section in the system		
Related use-cases:	Login		
Preconditions	1-The user has already signed up to the system 2-The user has provided a valid email address		
Trigger:	The user initiates a password reset		
Basic Flow:	1-The selects reset password option 2-The system sends an email with temporary login credentials 3-The user logs in and resets password from the reset password section in their account		
Post-Conditions:	**success:** Password is re-set **failure:** The password is unchanged and an alert is sent to the user		
Minimum Guarantee:	The user can view the login page with reset password option		
Risk Assessment:	Low		

Name:	Signup	Author	Salma Aly
Identifier:	UC4	Version:	1.2
Date Created:	Jan 30, 2016	Last Modified:	Feb 8, 2016
Importance:	5/5		
Actor(s):	Public User		
Goal:	To create an account on the scheduler system		
Summary:	The user signs up to create an account on the system by filling information to setup their account.		
Related use-cases:	Login		

Preconditions	The user has successfully accessed the system's Login/SignUp page
Trigger:	User activates the "Sign up" process
Basic Flow:	1-The user selects signup option 2- The user enter valid information corresponding to the specified fields. 3- The user submits their information 4-The system registers the user and creates an account for them
Post-Conditions:	**success:** The User has an account on the system **failure:** The user does not have an account and the system indicates to the user that the account was not created
Minimum Guarantee:	The user is redirected to the Login/SignUp page
Risk Assessment:	Low

Name:	View Default Schedule	Author	Salma Aly
Identifier:	UC5	Version:	1.2
Date Created:	Feb 2, 2016	Last Modified:	Feb 8, 2016
Importance:	2/5		
Actor(s):	Student or Public User		
Goal:	To view the default generated schedule for in sequence student		
Summary:	Students have the option of viewing the default schedule created by the scheduling system without the need to login or signup to the system.		
Related use-cases:	View Schedule		
Preconditions	The user has successfully accessed the landing page		
Trigger:	The user loads a generic automated schedule		
Basic Flow:	1-The user select view default schedule 2-The system displays the default schedule		
Post-Conditions:	**success**: The user successfully viewed the default schedule **failure**: The user cannot view the schedule		
Minimum Guarantee:	The system shows the landing page to the user		
Risk Assessment:	Low		

Name:	Modify Schedule		
Identifier:	UC6	Version:	1.1
Date Created:	8/2/2016	Last Modified:	9/2/2016
Importance:	5/5		
Actor(s):	Student		
Goal:	Access modification feature for current schedule		
Summary:	The Student may edit their current schedule.		
Related use-cases:	Delete Schedule, Change Section, Drop Course		
Preconditions	-User has been authenticated -Schedule has been generated		
Trigger:	User chooses to edit current schedule		
Basic Flow:	1. View Account Information 2. Enter changes to Account information 3. Save changes		
Post-Conditions:	**success:** The modified schedule information is saved in the system **failure:** The System cannot alter the user's schedule information		
Minimum Guarantee:	The system lets the user view account information		
Risk:	The information may not save properly.		
Author(s):	Nicolas Frazer-McKee		

Name:	Auto Generate List of Taken courses	Author	Salma Aly
Identifier:	UC7	Version:	1.2
Date Created:	Feb 7, 2016	Last Modified:	Feb 8, 2016
Importance:	5/5		
Actor(s):	Student		
Goal:	To View the list of courses that should have been completed with in the number of semesters the user has finished.		
Summary:	User enters the number of semesters finished so far and the scheduler generates the list of courses that correspond to the semesters taken based on the course sequence.		
Related use-cases:			
Preconditions	1-The user has a valid account on the system 2-The user is logged in to their account 3-The user has finished at least one semester		
Trigger:	User prompts system to validate login information		
Basic Flow:	1-User goes to the courses and preferences page 2-User enters the number of semesters taken		

	3-user selects auto generate course list
Post-Conditions:	**Success:** A list of courses corresponding to the number of semesters finished appear **Failure:** 1- The list is not generated or displayed properly 2- The user is redirected to the login page
Minimum Guarantee:	All other account information remain unchanged
Risk Assessment:	Low

Name:	Set Taken Courses	Author	Salma Aly
Identifier:	UC8	Version:	1.2
Date Created:	Feb 7, 2016	Last Modified:	Feb 8, 2016
Importance:	5/5		
Actor(s):	Student		
Goal:	Save the successfully finished courses to the student course record		
Summary:	The system allows the user to indicate which courses they have successfully finished in order to create an academic record for them.		
Related use-cases:			
Preconditions	1-The user has a valid account on the system 2-The user is logged in to their account 3-The user has already successfully finished one or more courses.		
Trigger:	User prompts system to validate login information		
Basic Flow:	1-User goes to the Courses and Preferences page 2-User enters the number of semesters finished and selects generate list 3- A list of courses appear 4-User selects the courses they have successfully finished 5- User selects add class to add courses that are not on the list		
Post-Conditions:	Success: User's account is updated with the courses taken so far. Failure: The courses are not saved to the student's account		
Minimum Guarantee:	Previously added courses remain saved in the user's course information.		
Risk Assessment:	Low		

Name:	Add needed course	Author:	Ying-Chen Chu
Identifier:	UC9	Version:	1.0
Date Created:	2015-02-03	Last Modified:	2015-02-08
Importance:	5/5		
Actor(s):	Student, Administrator		
Goal:	Add a needed course to the schedule		
Summary:	Select a course to be added to the list of courses for the schedule generator.		
Related use-cases:	-		
Preconditions	1. User has been authenticated 2. Course requirements are met		
Trigger:	User activates the “ Add Course'" process		
Basic Flow:	1. Choose a course 2. Save the course selection to the system		
Post-Conditions:	**Success:** Course is added to the user's list of courses **Failure:** System fails to process task and displays an error message.		
Minimum Guarantee:	Previous state of the system remains unchanged.		
Risk Assessment:	Low		

Name:	Drop course	Author:	Ying-Chen Chu
Identifier:	UC10	Version:	1.0
Date Created:	2015-02-03	Last Modified:	2015-02-08
Importance:	5		
Actor(s):	Student, Administrator		
Goal:	Remove a course from the schedule		
Summary:	Remove a course from the list of selected courses for the schedule generator.		
Related use-cases:	UC "**Modify Schedule**"		
Preconditions	-User has been authenticated -At least one course has been selected		
Trigger:	User activates the "drop course" process.		
Basic Flow:	1. Choose a course 2. Remove the course from the list of selected courses 3. Save the course removal to the system		
Post-Conditions:	**Success:** Course is removed from the user’s list of courses **Failure**: System fails to process task and displays an error message		
Minimum Guarantee:	Schedule remains the same		
Risk Assessment:	Low		

Name:	Set Preferences	Author	Ying-Chen Chu
Identifier:	UC11	Version:	1.0
Date Created:	2015-02-03	Last Modified:	2015-02-08
Importance:	5/5		
Actor(s):	Student, Administrator		
Goal:	Input the schedule preferences to the system		
Summary:	Set the schedule preferences and save them to the system.		
Related use-cases:	-		
Preconditions	User has been authenticated		
Trigger:	Previous state of the system remains unchanged.		
Basic Flow:	1. Input the schedule preferences 2. Save the preferences to the system		
Post-Conditions:	**Success:** The user's preferences are saved to the system. **Failure:** System fails to process the task, and an error messages is displayed.		
Minimum Guarantee:	Previous state of the system remains unchanged.		
Risk Assessment:	Low		

Name:	Modify Preferences	Author	Ying-Chen Chu
Identifier:	UC12	Version:	1.0
Date Created:	2015-02-03	Last Modified:	2015-02-08
Importance:	5/5		
Actor(s):	Student, Administrator		
Goal:	Modify the schedule preferences saved to the system		
Summary:	Access the current preferences and change any of the preferences from the list of preferences. Then, save this new list of preferences.		
Related use-cases:	-		
Preconditions	-User has been authenticated -A schedule preference was set		
Trigger:	User modifies and inputs new preferences		
Basic Flow:	1. Change any of the preferences from the list of preferences. 2. Save the list of preferences to the system		
Post-Conditions:	**Success:** The user's modified preferences are saved to the system. **Failure:** System fails to modify the preferences, and an error message is displayed.		
Minimum Guarantee:	Previous preferences remains unchanged.		
Risk Assessment:	Low		

Name:	Generate Schedule	Author	Ying-Chen Chu
Identifier:	UC13	Version:	1.0
Date Created:	2015-02-03	Last Modified:	2015-02-08
Importance:	5/5		
Actor(s):	Student, Administrator		
Goal:	Generate a schedule		
Summary:	Based on the course selection list and or without the list of preferences, the system generates schedule(s) for the selected courses.		
Related use-cases:	UC "**View Schedule**" and UC "**Save Schedule**"		
Preconditions	-User has been authenticated -A schedule preference was set		
Trigger:	User activates the "Generate Schedule" process		
Basic Flow:	1. A schedule generation is requested by the user to the system. 2. Schedule(s) are displayed to the user from the system.		
Post-Conditions:	**Success:** Schedule(s) for the course selected (and or without the preferences) is generated. **Failure:** System fails to generate a schedule and an error message displays		
Minimum Guarantee:	Previous state of the system remains the same.		
Risk Assessment:	High		

Name:	Save Schedule	Author	Ying-Chen Chu
Identifier:	UC14	Version:	1.0
Date Created:	2015-02-03	Last Modified:	2015-02-08
Importance:	5/5		
Actor(s):	Student, Administrator		
Goal:	Save a generated schedule		
Summary:	Save a schedule generated by the system by keeping a copy of the schedule on the system, accessible to the user.		
Related use-cases:	UC "**View Schedule**" and UC "**Save Schedule**"		
Preconditions	-User has been authenticated -Schedule(s) were generated		
Trigger:	Users activates the "save schedule" process.		
Basic Flow:	1. User chooses the schedules he wishes to save 2. User sends a request to the system to save these schedules 3. The chosen schedules are saved to the system		
Post-Conditions:	**Success:** Saved schedules are accessible to the user from the system **Failure:** System fails to save the schedule, and an error message displays.		
Minimum Guarantee:	Previous state of the system remains the same.		
Risk Assessment:	Low		

Name:	Delete Schedule	Author	Ying-Chen Chu
Identifier:	UC15	Version:	1.0
Date Created:	2015-02-03	Last Modified:	2015-02-08
Importance:	5/5		
Actor(s):	Student, Administrator		
Goal:	Delete a saved schedule		
Summary:	Delete a schedule from the list of saved schedules		
Related use-cases:	UC "**Modify Schedule**"		
Preconditions	-User has been authenticated -At least one schedule in the list of saved schedules		
Trigger:	Users activates "delete schedule" process.		
Basic Flow:	1. User chooses the schedule(s) he wishes to delete 2. User sends a request to the system to delete these schedule(s) 3. The chosen schedules are removed from the list of saved schedules.		
Post-Conditions:	**Success:** Deleted schedules are no longer saved, nor accessible through the system. **Failure:** System fails to delete schedule and an error message is displayed.		
Minimum Guarantee:	Previous state remains the same: schedule is not deleted.		
Risk Assessment:	Low		

Name:	Print Schedule	Author	Ying-Chen Chu
Identifier:	UC16	Version:	1.0
Date Created:	2015-02-03	Last Modified:	2015-02-08
Importance:	3/5		
Actor(s):	Student, Administrator		
Goal:	Print a saved or generated schedule		
Summary:	Print a schedule generated by the system. The schedule has either just been generated or can be accessed through the list of saved schedules.		
Related use-cases:	UC "**View Schedule**"		
Preconditions	-User has been authenticated -At least one schedule in the list of saved schedules or at least one schedule has been generated		
Trigger:	Users activates "print schedule" process.		
Basic Flow:	1. User decides whether to print a schedule from the saved schedule list or from the freshly generated schedule(s). 2. User sends a request to the system to print the chosen schedule 3. The system sends a request to a printer to print the chosen schedule		
Post-Conditions:	**Success:** The schedule is printed **Failure**: Failure to print the schedule, and a failure message is displayed.		
Minimum Guarantee:	Previous state of the system remains unchanged: schedule remains the same.		
Risk Assessment:	Medium		

Name:	View Schedule	Author	Ying-Chen Chu
Identifier:	UC17	Version:	1.0
Date Created:	2015-02-03	Last Modified:	2015-02-08
Importance:	5/5		
Actor(s):	Student, Administrator		
Goal:	View a saved schedule		
Summary:	View a schedule from the list of saved schedules		
Related use-cases:	UC "**Generate Schedule**"		
Preconditions	-User has been authenticated -At least one schedule in the list of saved schedules		
Trigger:	Users activates "view schedule" process.		
Basic Flow:	1. User chooses the schedule he wishes to view from the list of saved schedules 2. User sends a request to the system to view the schedule 3. The system displays the schedule to the user		
Post-Conditions:	**Success:** The user sees the scheduler he choose. **Failure:** System fails to display the schedule, and an error message is displayed.		
Minimum Guarantee:	Previous state of the system remains the same		
Risk Assessment:	Low		

Name:	Change Section	Author	Ying-Chen Chu
Identifier:	UC18	Version:	1.0
Date Created:	2015-02-03	Last Modified:	2015-02-08
Importance:	5/5		
Actor(s):	Student, Administrator		
Goal:	Change course section		
Summary:	Change the course section for a selected course from the list of selected courses.		
Related use-cases:	UC "**Modify Schedule"**		
Preconditions	-User has been authenticated -At least one course has been selected		
Trigger:	User activates the "change section" process.		
Basic Flow:	1. Choose a course 2. Remove the course from the list of selected courses 3. Save the course removal to the system		
Post-Conditions:	**Success:** The user is removed from the previous section to a new section for the course. **Failure:** System fails to process task and an error message is displayed.		
Minimum Guarantee:	Schedule remains the same		
Risk Assessment:	Medium		

Name:	Edit Account Information	Author	Adil Hssaini
Identifier:	UC19	Version:	1.0
Date Created:	Feb 4, 2016	Last Modified:	Feb 9, 2016
Importance:	5/5		
Actor(s):	Registered User / Administrator		
Goal:	To edit an existing account information.		
Summary:	The user adds or modifies entries into the existing account information. The system stores the newly entered information and updates account.		
Related use-cases:			
Preconditions	1. User is logged on. 3. System has accessed the account menu.		
Trigger:	User activates the “Edit Account Information” process.		
Basic Flow:	1. User selects account Information to edit. 2. System prompts to enter information. 3. User inputs information. 4. User commits to saving. 5. System responds with outcome of the operation.		
Post-Conditions:	**Success:** Student record is Updated successfully. **Failure:** System fails to process task and displays an error message.		
Minimum Guarantee:	All other system data, configurations and functionalities remain unchanged.		
Risk Assessment:	Low		

Name:	Set Taken Courses	Author	Adil Hssaini
Identifier:	UC20	Version:	1.0
Date Created:	Feb 4, 2016	Last Modified:	Feb 9, 2016
Importance:	5/5		
Actor(s):	Registered User / Administrator		
Goal:	To create a student record to be stored by the system		
Summary:	The user provides the system with a set of previously taken courses. The system stores that information and uses it to create a record for the specified account.		
Related use-cases:			
Preconditions	1. User is logged on. 2. System has accessed the account menu. 2. System has no existing record associated with the account.		
Trigger:	User activates the “Set Taken Courses” process		
Basic Flow:	1. User selects the “add a class” feature. 2. System prompts to enter information. 3. User inputs information. 4. User commits to saving. 5. System responds with outcome of the operation.		
Post-Conditions:	**Success:** Student record is created successfully.		

	Failure: System fails to process task and displays an error message.
Minimum Guarantee:	All other system data, configurations and functionalities remain unchanged.
Risk Assessment:	Low

Name:	Update Taken Courses	Author	Adil Hssaini
Identifier:	UC21	Version:	1.0
Date Created:	Feb 4, 2016	Last Modified:	Feb 9, 2016
Importance:	4/5		
Actor(s):	Registered User / Administrator		
Goal:	To update an existing student record.		
Summary:	The user adds or modifies entries into the student record. The system stores the newly entered information and updates record.		
Related use-cases:			
Preconditions	1. User is logged on. 3. System has the record menu.		
Trigger:	User activates the " Update Taken Courses" process		
Basic Flow:	1. User selects the "edit" feature. 2. System prompts to enter information. 3. User inputs information. 4. User commits to saving. 5. System responds with outcome of the operation.		
Post-Conditions:	**Success:** Student record is updated successfully. **Failure:** System fails to process task and displays an error message.		
Minimum Guarantee:	All other system data, configurations and functionalities remain unchanged.		
Risk Assessment:	Low		

Name:	Delete Taken Courses	Author	Adil Hssaini
Identifier:	UC22	Version:	1.0
Date Created:	Feb 4, 2016	Last Modified:	Feb 9, 2016
Importance:	2/5		
Actor(s):	Registered User / Administrator		
Goal:	To delete an existing student record.		
Summary:	The user deletes the existing student record associated with account. The system has no record associated with the account afterwards.		
Related use-cases:	UC **"Update Taken Courses"**		
Preconditions	1. User is logged on. 3. System has accessed the record menu.		
Trigger:	User activates the " Delete Taken Courses" process		

Basic Flow:	1. User selects the “delete” feature. 2. System prompts for confirmation. 3. User commits to deleting. 4. System responds with outcome of the operation.
Post-Conditions:	<u>**Success:**</u> Student record is deleted successfully. <u>**Failure:**</u> System fails to process task and displays an error message.
Minimum Guarantee:	All other system data, configurations and functionalities remain unchanged.
Risk Assessment:	Low

Name:	Add Course to Program	Author	Adil Hssaini
Identifier:	UC23	Version:	1.0
Date Created:	Feb 2, 2016	Last Modified:	Feb 9, 2016
Importance:	5/5		
Actor(s):	Administrator		
Goal:	To add a new course to a specific program.		
Summary:	The Administrator updates the list of required courses for a specific program by adding a new course.		
Related use-cases:			
Preconditions	1. Actor is logged on as administrator. 2. System has accessed the program menu. 3. The course is not part of the program sequence stored by the system.		
Trigger:	Administrator activates the “Add Course to Program” process.		
Basic Flow:	1. Administrator selects the “Add Course to Program” option. 2. System prompts for information about course. 3. Administrator inputs information. 4. Administrator commits to adding the course. 5. System responds with the outcome of the operation.		
Post-Conditions:	<u>**Success:**</u> Course is added successfully to the program listing. <u>**Failure:**</u> The system fails to process the task and displays an error message.		
Minimum Guarantee:	List of courses in the program stored by the system will not be affected.		
Risk Assessment:	Low		

Name:	Edit Course	Author	Adil Hssaini
Identifier:	UC24	Version:	1.0
Date Created:	Feb 2, 2016	Last Modified:	Feb 9, 2016
Importance:	5		
Actor(s):	Administrator		
Goal:	To edit a course's description or information.		
Summary:	The administrator modifies the existing description or details of a course.		
Related use-cases:			
Preconditions	1. Actor is logged on as administrator. 2. System has accessed the course menu.		
Trigger:	Administrator activates the “Edit Course” process.		
Basic Flow:	1. Administrator selects the “Edit Course” option. 2. System prompts for new information. 3. Administrator inputs information. 4. Administrator commits to Editing the course. 5. System responds with the outcome of the operation.		
Post-Conditions:	**Success:** Course is edited successfully. **Failure:** The system fails to process the task and displays an error message.		
Minimum Guarantee:	All other courses present in the list of courses required for the program will not be affected.		
Risk Assessment:	Low		

Name:	Delete Course	Author	Adil Hssaini
Identifier:	UC25	Version:	1.0
Date Created:	Feb 2, 2016	Last Modified:	Feb 9, 2016
Importance:	5		
Actor(s):	Administrator		
Goal:	To remove a course from a program listing.		
Summary:	The Administrator deletes a course from the list of required courses for a specific program.		
Related use-cases:			
Preconditions	1. Actor is logged on as administrator. 2. System has accessed the course menu.		
Trigger:	Administrator activates the “Delete Course” process.		
Basic Flow:	1. Administrator selects the “Delete Course” option. 2. System prompts for confirmation. 3. Administrator commits to deleting the course. 4. System responds with the outcome of the operation.		
Post-Conditions:	**Success:** Course is deleted successfully. **Failure:** The system fails to process the task and displays an error message.		

Minimum Guarantee:	All other courses present in the list of courses required for the program will not be affected.
Risk Assessment:	Low

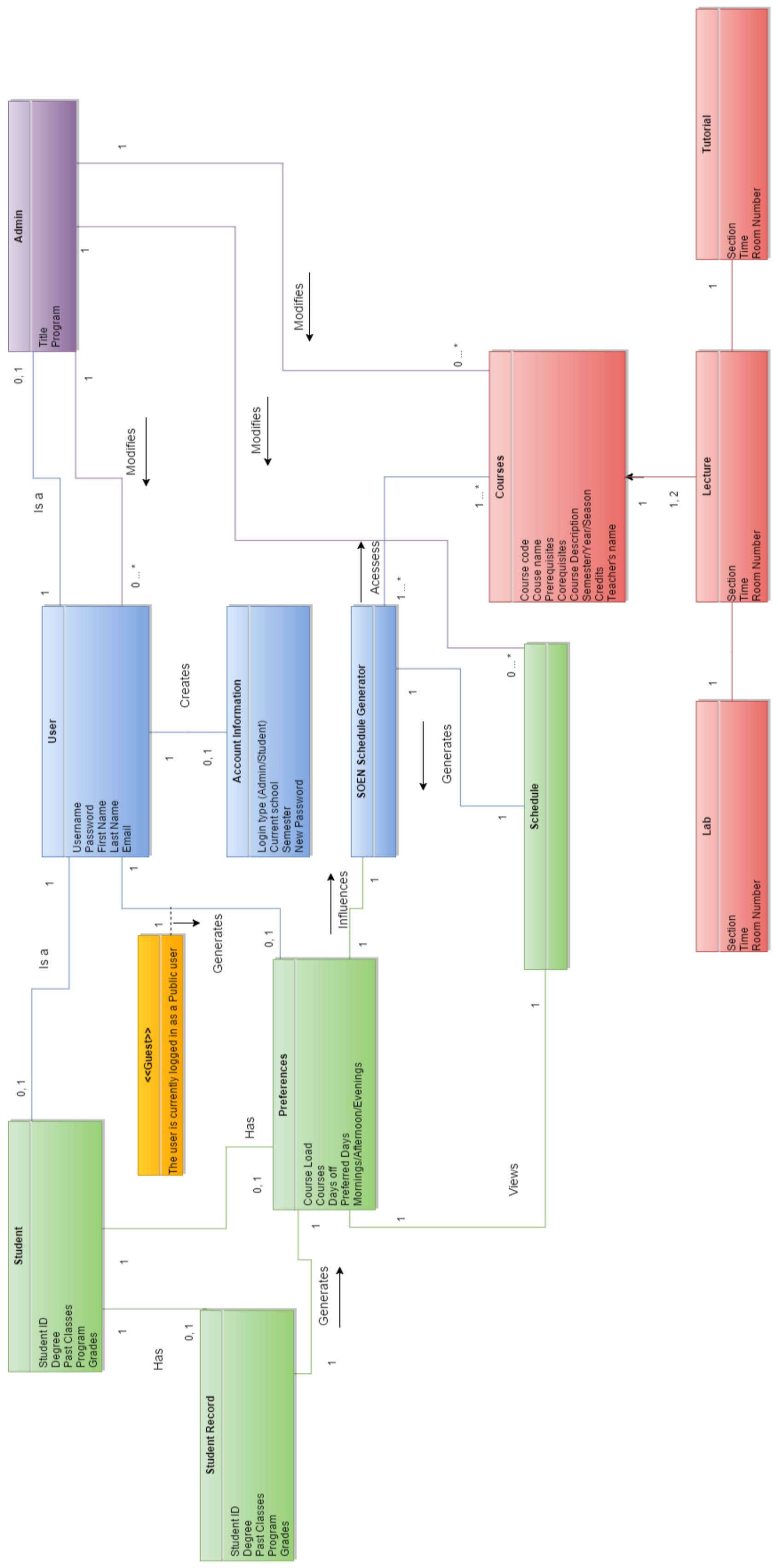
Name:	Add Section	Author	Adil Hssaini
Identifier:	UC26	Version:	1.0
Date Created:	Feb 2, 2016	Last Modified:	Feb 9, 2016
Importance:	5		
Actor(s):	Administrator		
Goal:	To add a course section.		
Summary:	The administrator adds a section to a course in the list of required courses for a program.		
Related use-cases:	UC – **Edit Course**		
Preconditions	1. Actor is logged on as administrator. 2. System has accessed the course menu.		
Trigger:	Administrator activates the “Add Section” process.		
Basic Flow:	1. Administrator selects the “Add Section” option. 2. System prompts for section details. 3. Administrator inputs information. 4. Administrator commits to adding the section. 5. System responds with the outcome of the operation		
Post-Conditions:	**Success:** Section is created successfully. **Failure:** The system fails to process the task and displays an error message.		
Minimum Guarantee:	All courses and sections previously stored by the system will not be affected.		
Risk Assessment:	Low		

Name:	Edit Section	Author	Adil Hssaini
Identifier:	UC27	Version:	1.0
Date Created:	Feb 2, 2016	Last Modified:	Feb 9, 2016
Importance:	5		
Actor(s):	Administrator		
Goal:	To edit information or details of a section.		
Summary:	The administrator modifies the existing details of a section.		
Related use-cases:	UC – **Edit Course**		
Preconditions	1. Actor is logged on as administrator. 2. The system has accessed the section menu.		
Trigger:	Administrator activates the “Edit Section” process.		

Basic Flow:	1. Administrator selects the “Edit Section” option. 2. System prompts for section's new details. 3. Administrator inputs information. 4. Administrator commits to editing the section. 5. System responds with the outcome of the operation
Post-Conditions:	**Success:** Section details are modified successfully. **Failure:** The system fails to process the task and displays an error message.
Minimum Guarantee:	All courses and sections previously stored by the system will not be affected.
Risk Assessment:	Low

Name:	Delete Section	Author	Adil Hssaini
Identifier:	UC28	Version:	1.0
Date Created:	Feb 2, 2016	Last Modified:	Feb 9, 2016
Importance:	5		
Actor(s):	Administrator		
Goal:	To delete a course section.		
Summary:	The administrator deletes a course section from the list of available sections.		
Related use-cases:	UC - “**Edit Course**”		
Preconditions	1. Actor is logged on as administrator. 2. System has accessed to the section menu.		
Trigger:	Administrator activates the “Delete Section” process.		
Basic Flow:	1. Administrator selects the “Delete Section” option. 2. System prompts for confirmation. 3. Administrator commits to deleting the section. 4. System responds with the outcome of the operation		
Post-Conditions:	**Success:** Section is deleted successfully. **Failure:** The system fails to process the task and displays an error message.		
Minimum Guarantee:	All courses and other sections previously stored by the system will not be affected.		
Risk Assessment:	Low		

3.2 DOMAIN MODEL



3.3 CONSTRAINT AND QUALITIES

Non Functional Requirements

3.3.1 Product

3.3.1.1 Security

The system shall be developed and coded with security at the forefront of concerns. The scheduler is implemented with React as a web application hosted by an Apache Web Server, and therefore provides default configurations as well as custom configurations that significantly help reduce XSS vulnerabilities and prevent attacks such as information leakage or PHP injections.

3.3.1.2 Privacy

To enforce and ensure privacy, practices such as session locks and expiry logouts shall be enforced. An encryption algorithm will also be implemented to provide an additional layer of abstraction to registered user's data.

3.3.1.3 Maintainability

The scheduler has several features that render its maintenance process more flexible than similar products. The use of React is in line with the nature of the system based on data that changes overtime. In addition, JavaScript being a very flexible and powerful language, makes expansions and additional options easily feasible as opposed to the standard directives or templates.

3.3.1.4 Compatibility

The system shall be compatible with all mainstream platforms browsers.
The system shall work on innate as well as on virtual machines.
The system shall be accessed worldwide unless restricted by recipients' service providers or network settings.

3.3.1.5 Portability

The scheduler is not linked to any specific database. It is designed to operate as a public interface enabling access to multiple universities' programs. Options to link to selected databases are possible and are taken into account by the design.

3.3.1.6 Performance

The scheduler guarantees operations execution with minimal complexity. Coding aims at optimizing hardware and therefore reducing the system's and user devices' response time.

3.3.2. Organizational

3.3.2.1 Development

The system is designed from a prototype; a basic functional prototype will be developed and used in order to evaluate the Feasibility of requirements and strength of the design.

The system shall pose full documentation for requirements and design models. The documentation will be broken down and separated into sections in order to provide a work breakdown structure.

3.3.2.2 Operational

The system is designed to function with various sizes of databases: as long as the appropriate databases respect the current SQL schematic for tables and relationships, the database can be changed. Furthermore, the web scheduler shall possess a model view controller design pattern.

A model shall be used for the object oriented back-end to manage data. This model can respond to requests from the front end view and the overall controller. This controller will direct user input and general management directives in order to change the state of the model and its data.

3.3.2.3 Environmental

The system requires internet for all user access: the web application can only be utilized through a web browser on a device with internet access. The system requires access to databases for user information. This information is stored in SQL databases that must be accessed to perform all basic operations (login, see schedule).

3.3.3. External

3.3.3.1 Regulatory

The Schedulers' team shall conform to Concordia University's Academic Code of Conduct. The web application will also respect trademarks and intellectual property, and the supporting documentation will cite and reference all works used.

3.3.3.2 Legislative

For accounting purposes, the scheduler's team's captains will keep a time record for meetings. A time log record for each team meeting, created by a team leader, will be uploaded to the supporting documents for the Scheduler's team. This shall be added to the individually kept team member records and logs for time spent on each section: Individual members will record the time spent on each assigned sections.

4. RESOURCE EVALUATION

4.1 HUMAN RESOURCES

Team Member	**Nicolas Frazer-McKee**
Role	Documentation
Knowledge	Javascript, PHP, HTML, CSS, Java, c++, SQL
Experience	- Personal web page editing and hosting - Formal documentation writing as scientific papers
Strengths	- Formal writing - Diagram creation
Availability	Evenings during the week

Team Member	**Le Vinh Dang**
Role	Back-End Programmer
Knowledge	SQL, C++, Ruby
Experience	- Teamwork experience - Object-oriented programming - Work experience as software developer
Strengths	Debugging - Problem solving - SQL - Object Oriented
Availability	Any Time

Team Member	**Dias Marat**
Role	Back-End Programmer
Knowledge	Java, Javascript, HTML5, CSS3, PHP, Arduino, Objective C, C#
Experience	- Developed IPhone Applications and video games for several years - Developed server side applications using NodeJS and worked on backend previously - Team projects
Strengths	- Debugging - Problem solving - Mobile Development and Web Development
Availability	Any time

Team Member	**Bruce Edouard Brazier**
Role	Back-end Lead
Knowledge	Javascript, PHP, HTML, CSS, Java, Bash,SQL
Experience	- 1 work term - Work/Personal Experience maintaining a website (Client/Server side) - Work experience working with Software Deployments and Scripting
Strengths	- Troubleshooting - Process Automation - Team coordination
Availability	Any time

Team Member	**Sean Marcoux**
Role	Front-end Lead
Knowledge	React, Javascript, PHP, HTML, CSS, Java
Experience	- 1 work term - Work experience developing React application in teams - Work experience developing improvements and bug fixes for a Java, eclipse RCP project
Strengths	- Developing refined user experiences - Debugging - Problem Solving - Java + React
Availability	6 hours a week

Team Member	**Adriel Fabella**
Role	Front-end programmer
Knowledge	Javascript, PHP, SQL , HTML, CSS, Java.
Experience	- Part-time web designing and developing - Volunteer experience with PHP/ MySQL - Team Projects
Strengths	- Web developing - Communication skills - Problem solving
Availability	Any time

Team Member	**Adil Hssaini**
Role	Design & Development
Knowledge	C++, java, PHP, SQL, HTML, CSS
Experience	- Software maintenance experience - Prior involvement in large projects
Strengths	- Time and resource management - Versatile background - Team organization
Availability	4 to 6 hours per week

Team Member	**Emili Vasseva**
Role	Team leader and documenter
Knowledge	Javascript, PHP, SQL , HTML, CSS, Java, C++, Prolog, Lisp, AspectJ
Experience	- Contractual web developer using HTML, CSS, JavaScript, PHP, MySQL - Currently developing a small game in Unity - Team Projects
Strength	- Web development - Problem solving - Leadership - Object-oriented programming
Availability	Any time

Team Member	**Ying-Chen Chu**
Role	Documentation
Knowledge	Java, C++, Ruby, Ruby on rails, HTML, SQL, Arduino
Experience	- Work term as software developer - Developed a simulation framework - Team projects
Strengths	- Object-oriented programming - Problem solving
Availability	6 hours per week

Team Member	**Alex Eladas**
Role	Documentation and Corrector
Knowledge	C++, HTML,CSS
Experience	- 1 work term - Worked on designing and implementing a new process. - Team projects
Strengths	- Object-oriented programming - Problem solving - Analytical Skills
Availability	Anytime

Team Member	**Gabriele Bavaro**
Role	Front-end programmer
Knowledge	JavaScript, HTML, PHP, SQL, CSS, Java, C++, Python
Experience	Created numerous websites through WordPress and other web tools Helped program for a mars rover robot for Space Concordia Team projects
Strengths	Web development Problem solving Communication and teamwork skills
Availability	Any time

Team Member	**Salma Aly**
Role	Documenter
Knowledge	C++, Java, Python
Experience	- Teamwork experience - Object-oriented programming - Work experience as software developer - Documentation
Strengths	- Debugging - Problem solving - Testing - Object Oriented Programming
Availability	5 hours/Week

4.2 TECHNICAL RESOURCES

The following section has been divided into several subsections which discuss technical resources associated with documentation, programming languages and software, hardware, operating systems, communication and management.

4.2.1 Documentation

The software applications that are being used for editing and reviewing the source documentations and codes are Google Docs/Drive and Microsoft word. Google Docs is a flexible program that allows team members to brainstorm and work concurrently on a document in real-time. Google Drive is used to store Google docs files in a shared folder accessible by all team members. Microsoft Word was used for more individual documentation from team members and for greater flexibility in organizing documented information. Adobe Reader was utilized in order to render documents into a format that could be read by all team members and their affiliates. WireFrame was used to design and showcase the front end interfaces and pages that would become part of the completed product.

4.2.2 Programming Languages and Software

The web server used to store the database is the WampServer64 2.5. The WAMP server, which stands for Windows, Apache, MySQL and PHP, provides those programs for use. With the WampServer64 version 2.5, it will support the Apache : 2.4.9, the MySQL : 5.6.17, the PHP : 5.5.12, the PHPMyAdmin : 4.1.14, the SqlBuddy : 1.3.3, and the XDebug : 2.2.5. The programming languages which will be used to construct the final product are HTML, PHP, JavaScript and CSS.

4.2.3 Hardware

In addition to the above programs, laptops and desktops will be used to install and carry the WAMP database and other software that will be required for the project. The laptops are not of a uniform variety but instead come from a wide range of manufacturers. They are DELL, HP and MAC laptops and desktops. In addition to containing the above server, the laptops and desktops also contain React, a JavaScript software used to develop the front end of the final product.

4.2.4 Operating Systems

In order to facilitate uniformity amongst team members the team laptops/desktops must have the following minimum requirements:

- 150 GB of memory
- 4 GB of RAM
- Intel Premium 4 or AMD Athlon x64

- WI-FI internet access
- Windows 7/Linux operating systems
- Headsets, earphones and speaker setups (to allow for discussions through skype)

4.2.5 Communication and Management

All code and document files related to the project are sorted and stored on Github under the repository Schedule-Builder through the use of Github accounts. All team members have access to Github folders, files and their content. To facilitate communication between team members, Facebook, Slack and Skype are used for holding discussions and meetings.

5. SCOPING

In order to fulfill the requirements in section 3.1 and 3.3, Team leaders with more experience were assigned to each of the 3 sub-teams; front end-html and web design, back end-OOP with PHP, and documentation- further work breakdown structure. This breakdown allowed to streamline parts of the project efficiently; early in the project, one type of user was removed as it was judged to be unnecessary and detracted from the main user while adding a fair amount of complexity: Professor. The removal of this user simplified the system by rendering a request feature obsolete. This represented a significant difficulty for the programming teams and was ultimately useless.

The resulting system's full scope as a web application to be used for Software Engineering students is highlighted by the following lists of included (scoped in) and excluded (scoped out) features that extend the minimum requirements.

5.1 SCOPED IN

The System will allow 3 types of users: Student, Public user and Administrator.

- The Administrator is a course and department moderator who can access, modify and add any course or section.
- The Public user enables generation of a default schedule feature for the system and the ability to view this schedule without filling in any of the preferences or taken classes.
- The Student represents the focal user who has access to the schedule generation. This user may login to access the system and any previously saved data pertaining to the account or schedule. In order to generate a schedule for the student, the student will access and edit the preferences, the taken courses, account information and login.

The preferences consist of choices made from a list of possible constraints that will be applied to the schedule generator. The student may alter the schedule-to allow any final changes. These changes can be saved for future editing or consulting or even deleting.

5.2 SCOPED OUT

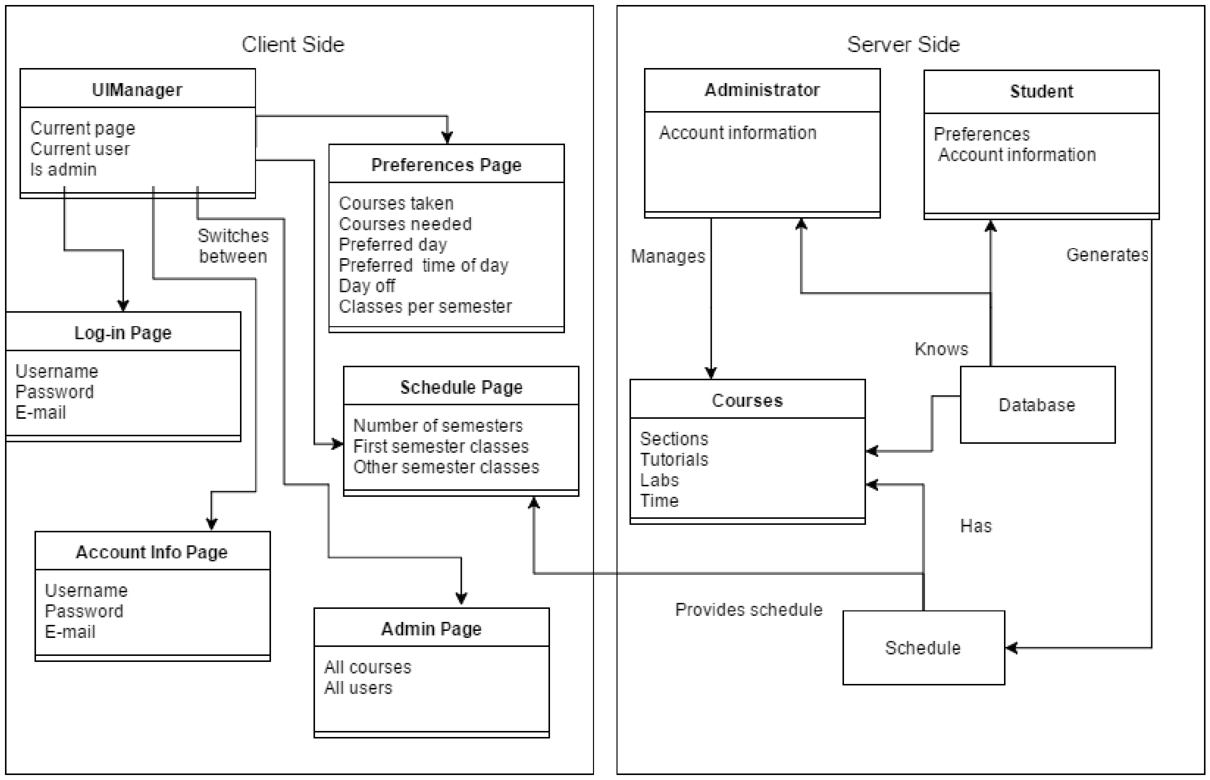
For this first deliverable, the programming teams have elected to scope out minor functionalities that represented significant programming hurtles:

- Print schedule: this is a feature which does not influence the system and can easily be removed.
- Save schedule: this feature was judged to be redundant as the system will automatically save a schedule.
- Delete schedule, Change Section, Drop course
- Auto generated schedule: in order to simplify the schedule generation, the back-end programming team has opted to make the schedule immutable.

6. SOLUTION SKETCH

6.1 ARCHITECTURE

Unlike the standard MVC architecture, the model, the view and the controller are not taken care by the same framework. In our architecture, the view is handled using React and data manipulation as well as database queries, are handled by Laravel. This means that the application is divided between client side and the server side. The client side handles everything to do with the view (everything the user sees and interacts with) while the server side handles everything else.



6.1.1 Server Side

For the server side, the components are the users (students, admin), the schedule, the database and the courses. The database contains information about the students, the administrators and the courses. When the users modify their preferences and their information, it is updated in the database by a query. The students interact with the schedule component when they generate their schedule based on their preferences. The schedule component then fetches their preferences and generate the appropriate schedule. The administrator component can manage the courses and their properties. After the required information is gathered through the user page, the appropriate courses are then modified with a database query. Finally, the schedule provides the client side with the data that is to be displayed on the pages such as the student schedule or the full course sequence. This setup allows us to control the information of the students and the administrators. Ensuring that when they are needed, they can be accessed through the database. This will also facilitate the process of generating the algorithm since the components are independent and will be easier to manipulate.

6.1.2 Client Side

The main components for the client side are the UIManager and the components for each page. These will all be React components. The UIManager will be the necessary main React component and it will handle switching between all of the pages and hold the data that is common to all of them: the active user and if that user is an admin. This structure is the best way to handle the UI because switching between pages will be as simple as changing which component is being rendered. It also allows simple communication between pages through the UIManager.

The page components are the log-in, preferences, account info, schedule, and admin pages. The log-in page needs to keep track of any input the user enters, which are username, password, and e-mail (if the user is registering for the first time). The account info page needs the same info, but this needs to be the information obtained from the server. The preferences page is where the user sets the courses they've taken, the courses they still need to take, and the preferences they have for their schedule. The preferences page component will keep track of all this info and will obtain any of it from the server if the user already input preferences in the past. From the preferences page, the user can click a button to build the schedule. This will generate the schedule on the server side and return the schedule information on the schedule page. The first semester classes here is a separate variable because these classes need the additional information of time, classroom, section, and teacher. The remaining semesters will simply be a list of classes for their recommended course sequence. Finally, the admin page will obtain a list of all courses and a list of all users registered in the database, allowing the admin to edit them.

6.2 TECHNOLOGIES IN USE

6.2.1 Programming Languages

6.2.2.1 HTML

HTML is a computer and markup language that allows to create web sites and web documents. This language will consists of the very backbone of the website, when it comes to filling up the webpage with text and dialogs.

6.2.1.2 CSC

CSC is the language within the markup language that allows to manipulate the design of the web document, meaning positioning, color and overall presentation.

6.2.1.3 JavaScript

JavaScript is the main client side programming language used for creating interactive websites. JavaScript support is built right into all the major browsers and can support object oriented programming. JavaScript will be the main dynamic language used by the front end team and everyone in the team possesses experience working with JavaScript.

6.2.1.4 React

React is an open source JavaScript library which contains a template language and some function hooks to efficiently render HTML. React manages all UI updates when data has been changed and will update only those changed data. This is efficient because the user can tell how a component will render by looking at one source file. A program flow does not need to be traced which can be efficient when working in a big team.

6.2.1.5 PHP

PHP is a server side scripting programming language used for web development. It is very well documented and can support objected oriented programming. PHP will be used for the backend and most of the team members have experience with PHP.

6.2.2 Framework

6.2.2.1 Laravel

Laravel is an open source PHP web application framework that allows rapid development of web applications. Laravel uses MVC architecture and has features such as module package manager, template engine, database seeding, routes, authentication, and object oriented design. This will provide clean and manageable code.

6.2.3 Integrated Development Environments/Editors

6.2.3.1 PhpStorm

PhpStorm is an Integrated Development Environment for Windows and Mac OS that allows developers to code their projects in PHP. It has syntax highlighting, plugins, different types of frameworks supported such as Symfony, Laravel, CakePHP, built in support for databases, version control, debugging and testing. PhpStorm increases the productivity of developers.

6.2.4 Source Code and Revision Management

6.2.4.1 Git

Git is a source code management system used for software development. It allows developers to save different versions of their projects at different points in time and compares them to one another. Git allows developers to contribute to a repository (project) even if the developer is not connected to the Internet. It stores a local copy of the project on the local repository and changes made on the local repository can be pushed to the main repository. This promotes organization and maintains previous versions of the project.

6.2.5 Collaboration Software

6.2.5.1 GitHub

Github is a website that hosts Git repositories and has all the functionalities of Git. It provides bug tracking, feature requests and wikis for projects. It is used in this project as it allows efficient collaboration between developers.

6.2.5.2 Google Docs

Google Docs is an online word processor that allows individuals to edit and collaborate on documents in real-time. It is free and can be accessed by anybody.

6.2.5.3 Draw.io

Draw.io is a software application that provides tools to draw domain models, UML, use cases diagrams and etc. It allows collaboration between individuals and can be used as a plugin to Google Drive.

6.2.5.4 MockFlow – Wireframe pro

MockFlow – Wireframe pro is an application used to create the mockup of the website. It allows collaboration where the whole team contributes to creating the backbone using the available widgets and elements and to display comments and reactions.

6.2.5.6 Skype

Skype is a free video chat application that allows users to do video conference calls and exchange documents. For this project it is used between sub sections of the teams because it is more efficient.

7. PLAN

The following section depicts the tentative schedule that will be applied during the rest of the project. Each table represents an activity and its artifact and their respective number of hours of execution. Furthermore, each activity has been assigned to various team members.

7.1 ACTIVITIES, ARTIFACTS AND ACTIVITES ASSIGNMENTS

7.1.1 Deliverable 0 - System Overview

Due date: January 13th, 2016

The purpose of this deliverable is to familiarize ourselves with the project, and therefore create a domain model on how the software should behave.

Activity:	**Team Assignments**
Purpose	Assigning roles to the team in terms of their preferences and strength/weaknesses and electing a team leader
Artifact #1	Team members list
Description	List with the name and role of each team member
Combined total work hours	1
Due date	January 8th 2016
Participants	Emili, Sean, Dias, Bruce,

Activity	**System Definition**
Purpose	A concise description of the software to be developed with its purpose, functions and its classes of users.
Artifact #1	Domain Model
Description	The principal entities and their relationships. Not including any methods.
Combined total work hours	4
Due date	January 9th -January 11th 2016
Participants	Salma, Ying-Chen, Adriel, Gabriele, Le Vinh, Alex

7.1.2 Deliverable 1: Requirements, Scope and Plan

Due date: February 10th, 2016

The purpose of this deliverable is to work on the basic structure (UCD, DM and basic architecture), to create a plan for the project, as well as creating a small prototype.

Activity	Defining Requirements
Purpose	To describe the functionality of the system in terms of processing each user actions. Defining the main functions of The Scheduler that take place when generating an output
Artifact #1	Use Case Diagram
Description	A diagram explaining the interactions between the actors and functions of the system and showing the relationship between the use cases.
Combined total work hours	10
Artifact #2	Use Cases
Description	A complete list of all the use cases included in the system.
Combined total work hours	12
Artifact #3	Domain Model
Description	Updated domain model containing the attributes and associations between each class objects.
Combined total work hours	4
Due date	January 22th February 7th 2016
Participants	Salma, Adil, Ying-Chen, Nick

Activity	Architecture
Purpose	A preliminary description of the high-level structure showing the early version of the proposed solution and the reasons leading up to this design.
Artifact #1	Non-Functional Requirements
Description	The constraints the system will undoubtedly meet throughout its development.
Combined total work hours	10
Due date	January 29th-February 7th 2016
Participants	Bruce, Sean,

Activity	**Resources**
Purpose	Evaluating the experience and knowledge each team member can bring to the project. Presenting the list of the available technologies for the project.
Artifact #1	Technologies used
Description	A list of the different hardware, software or any other tool that could be used for the system's development.
Combined total work hours	2
Due date	February 6th-February 7th 2016
Participants	Gabriel, Adriel

Activity	**Planning**
Purpose	Describing every activity and documentation to be completed throughout the development of the system
Artifact #1	Estimation
Description	A time and cost estimation for the completion of the project
Combined total work hours	2
Artifact #2	Schedule
Description	A diagram showcasing the timetable for each main phases. (Gantt Chart)
Combined total work hours	2
Artifact #3	Risks
Description	A list of the various risks that could be encountered during the development of the system
Combined total work hours	3
Due date	February 4th-February 8th
Participants	Emili, Alex

Activity	**Prototyping**
Purpose	An early version of the system proving that the technologies used are proper for the project
Artifact #1	Working framework
Description	An initial design of the system that describes its main functions.
Combined total work hours	10
Artifact #2	Server Connection
Description	An initial call to the servers implemented in the prototype demonstrating the information storage
Combined total work hours	15
Due date	February 6th February 9th 2016
Participants	Sean, Bruce, Le Vinh, Dias, Adriel, Gabriel

7.1.3 Deliverable 2: Design

Due date: March 9th, 2016

The purpose of this deliverable is to develop the full structure and design of the software, and create a rapid prototype out of these.

Activity	Detailed Architecture
Purpose	Overall structure of the system
Artifact #1	4+1 Architectural View
Description	High-level structure of the system, composed of 5 views: logical view, process view, development view, physical view and scenarios. Used to describe a large system into multiple subsystems.
Combined total work hours	10
Artifact #2	Subsystems Interface Specifications/Module Interface Specifications
Description	Description of each subsystems meant to complete specific services, and their parameters (invalid/valid values) passed in functions.
Combined total work hours	25
Start/End dates	February 11th - February 29th
Participants	Sean, Bruce, Emili, Nick

Activity	Detailed Design
Purpose	Complete class description of each subsystem
Artifact #1	UML Class Diagram
Description	Connection between classes of each subsystems
Combined total work hours	12
Artifact #2	Dynamic Design Scenarios
Description	2 dynamic design of 2 uses cases (using at least 3 system operations). This includes system sequence, operational contracts, and sequence diagrams.
Combined total work hours	6
Artifact #3	Estimation
Description	Estimated cost for integration, testing and documentation for each module.
Combined total work hours	7
Start/End dates	February 24th - March 8th
Participants	Adil, Alex, Salma, Ying-Chen

Activity	Rapid Prototyping
Purpose	Programming of the prototype designed using the architecture and design description.
Artifact #1	Rapid Prototyping report
Description	Listing and commenting on classes/modules/drivers used for the rapid prototype.
Combined total work hours	22
Artifact #2	Testing
Description	Testing code and report of the rapid prototype
Combined total work hours	15
Artifact #3	Risks
Description	Update of the risks, cost estimate and scoping from the deliverable 1 to deliverable 2
Combined total work hours	4
Start/End dates	February 24th - March 8th
Participants	Dias, Adriel, Gabriele, Le Vinh

7.1.4 Deliverable 3: Testing

Due date: April 6th, 2016

The goal of this prototype is to finalize the programming with the respect test report. Furthermore, an instruction manual and a final cost estimate has to be documented.

Activity	Final prototype
Purpose	Final prototype of the a fully working software
Artifact #1	Test report and instructions manuals
Description	1. Test report on the entire making of the system 2. Instruction manual for future users
Combined total work hours	60
Start/End dates	March 10th - March 31st
Participants	Sean, Gabriele, Adriel, Le Vinh, Bruce, Dias

Activity	Testing Report
Purpose	Final report on all testing done on the final product
Artifact #1	Test coverage
Description	1. Listing of all tested items, and why. 2. Identification of 5 classes/methods and why they were tested.
Combined total work hours	20
Artifact #2	Test cases
Description	1. Two mid-level units tests, with their respective test cases and descriptions. 2. Requirements testing and their test cases 3. Test cases of potential extreme system usages, and their respective description 4. Test testing regarding the security of the system
Combined total work hours	15
Start/End dates	March 10th - April 1st
Participants	Emili, Adil, Salma, Nick

Activity	System delivery
Purpose	Instructions on the system
Artifact #1	Installation Manual
Description	Step by step instructions on how to install the system.
Combined total work hours	6
Artifact #2	Users Manual
Description	Step by step instructions on how to use the system.
Combined total work hours	6
Start/End dates	April 1st - April 5th
Participants	Alex, Ying-Chen

Activity	Final cost estimate
Purpose	Final coverage on the total amount of hours and money spent on the project
Artifact #1	Working hours
Description	Final coverage on the number of hours put into the project by each person.
Combined total work hours	10
Artifact #2	Cost
Description	Final coverage on the costs spend on each individuals work and for technological resources
Combined total work hours	4
Start/End dates	April 1st - April 5th
Participants	Ying-Chen, Alex

7.1.5 Final Deliverable: Complete Report

Due date: April 13th, 2016

This section is the final delivery, consisting of finalizing the report.

Activity	Finalization of the deliverable
Purpose	Completion of the project by submitting a complete and corrected report.
Artifact #1	Final report
Description	Assembling and correction over the report and its content.
Combined total work hours	50
Start/End dates	April 7th - April 12th
Participants	Entire team

Estimated Total Hours: 352

7.2 PROJECT ESTIMATES

7.2.1 Basis for estimates

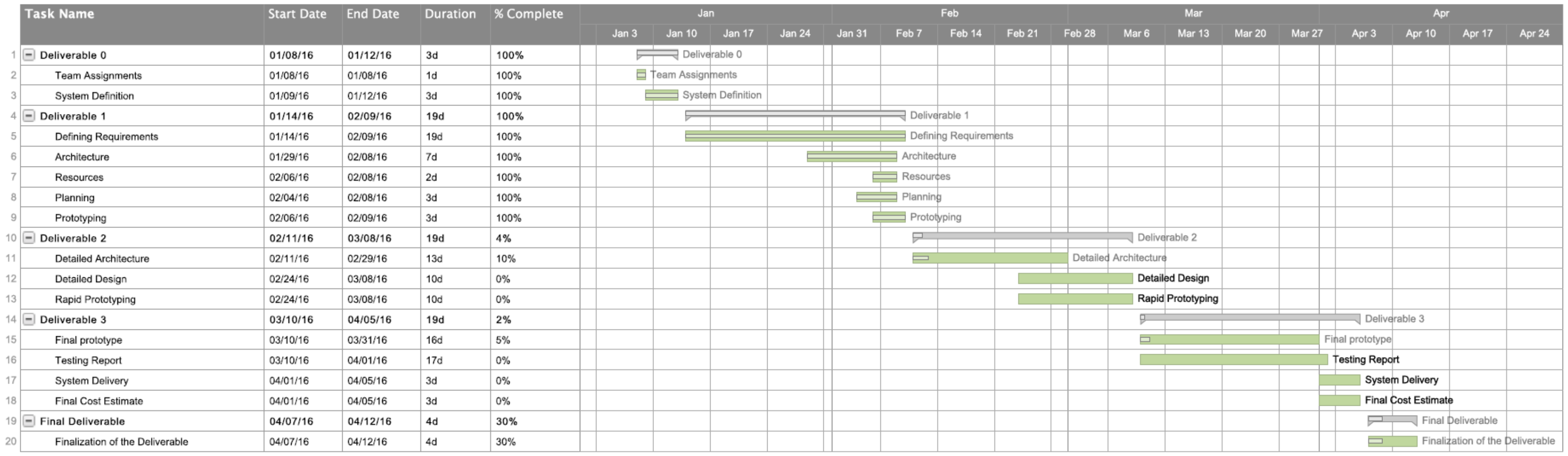
The basis for each artifact estimation came from analyzing the deliverables to be completed for the project. By breaking down all the sections, evaluating the difficulty of each tasks and considering the number of participants, the approximate working hours were calculated and added. The estimation will be revised later into the project if an important problem arises, which could delay the whole working process.

Assuming that the software engineers involved in the development are paid at an average rate of $25/hour.

Estimated Cost for the full project. At an hourly rate of $25/hour.

Hardware	
Computers, Servers:	$0
Software:	
Software/Technologies used:	$0
Software development/Documentation	$8800
Total:	$8800

7.3 SCHEDULE



	Task Name	Start Date	End Date	Duration	% Complete
1	Deliverable 0	01/08/16	01/12/16	3d	100%
2	Team Assignments	01/08/16	01/08/16	1d	100%
3	System Definition	01/09/16	01/12/16	3d	100%
4	Deliverable 1	01/14/16	02/09/16	19d	100%
5	Defining Requirements	01/14/16	02/09/16	19d	100%
6	Architecture	01/29/16	02/08/16	7d	100%
7	Resources	02/06/16	02/08/16	2d	100%
8	Planning	02/04/16	02/08/16	3d	100%
9	Prototyping	02/06/16	02/09/16	3d	100%
10	Deliverable 2	02/11/16	03/08/16	19d	4%
11	Detailed Architecture	02/11/16	02/29/16	13d	10%
12	Detailed Design	02/24/16	03/08/16	10d	0%
13	Rapid Prototyping	02/24/16	03/08/16	10d	0%
14	Deliverable 3	03/10/16	04/05/16	19d	2%
15	Final prototype	03/10/16	03/31/16	16d	5%
16	Testing Report	03/10/16	04/01/16	17d	0%
17	System Delivery	04/01/16	04/05/16	3d	0%
18	Final Cost Estimate	04/01/16	04/05/16	3d	0%
19	Final Deliverable	04/07/16	04/12/16	4d	30%
20	Finalization of the Deliverable	04/07/16	04/12/16	4d	30%

7.4 RISKS

7.4.1 Knowledge of frameworks

The laravel framework for PHP and React.js library for javascript represent a risk, since only a few members of the team are familiar with their use and their learning curve could play in some scheduling issues. In others works, developing the system could take longer than anticipated.

7.4.2 Time

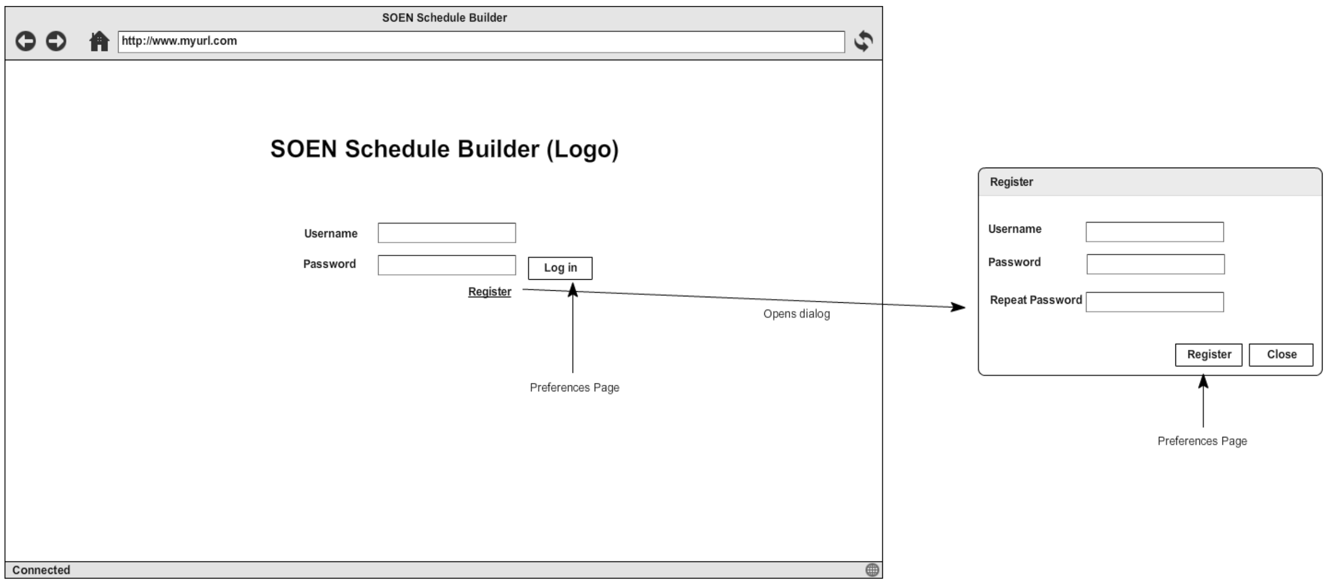
A single semester may not be enough to complete the whole system. Some key features may be left out due to the final deadlines, which would hinder the usability of the Scheduler.

8. PROTOTYPING

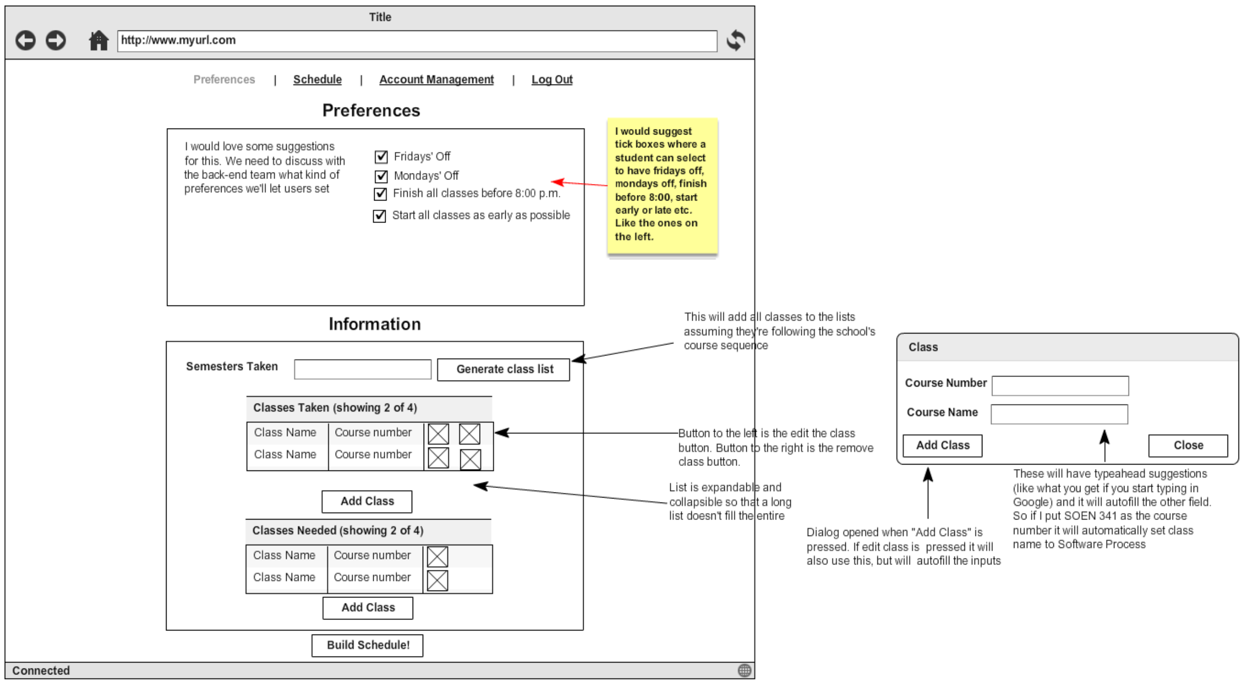
8.1 Mock-ups

To create quick, simple designs of the website we used an online tool called MockFlow.

8.1.1 Log-In Page



8.1.2 Preference Page



I would love some suggestions for this. We need to discuss with the back-end team what kind of preferences we'll let users set

I would suggest tick boxes where a student can select to have fridays off, mondays off, finish before 8:00, start early or late etc. Like the ones on the left.

This will add all classes to the lists assuming they're following the school's course sequence

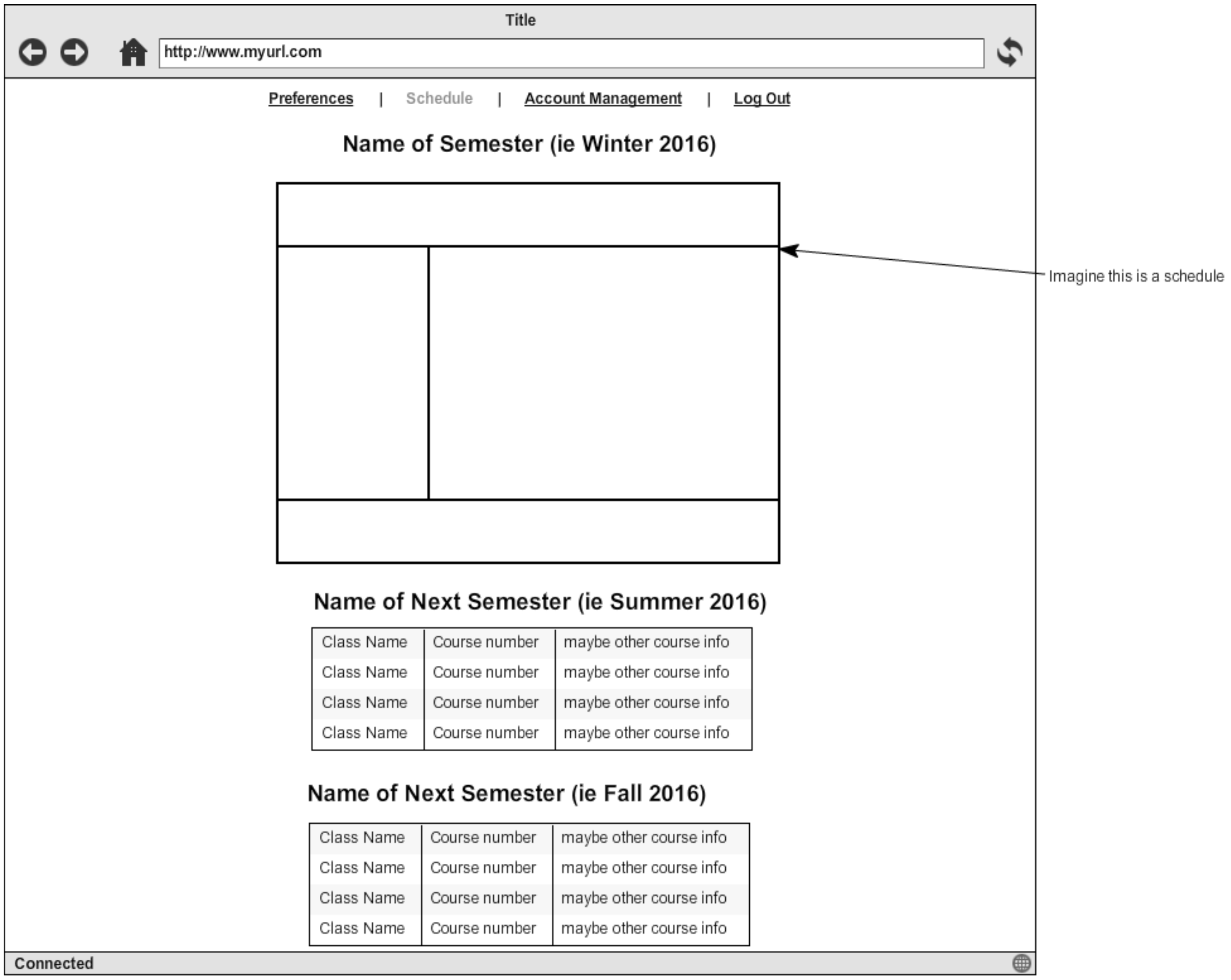
Button to the left is the edit the class button. Button to the right is the remove class button.

List is expandable and collapsible so that a long list doesn't fill the entire

Dialog opened when "Add Class" is pressed. If edit class is pressed it will also use this, but will autofill the inputs

These will have typeahead suggestions (like what you get if you start typing in Google) and it will autofill the other field. So if I put SOEN 341 as the course number it will automatically set class name to Software Process

8.1.3 Schedule Page


Title
http://www.myurl.com
Preferences | Schedule | Account Management | Log Out
Imagine this is a schedule
Connected

Name of Semester (ie Winter 2016)

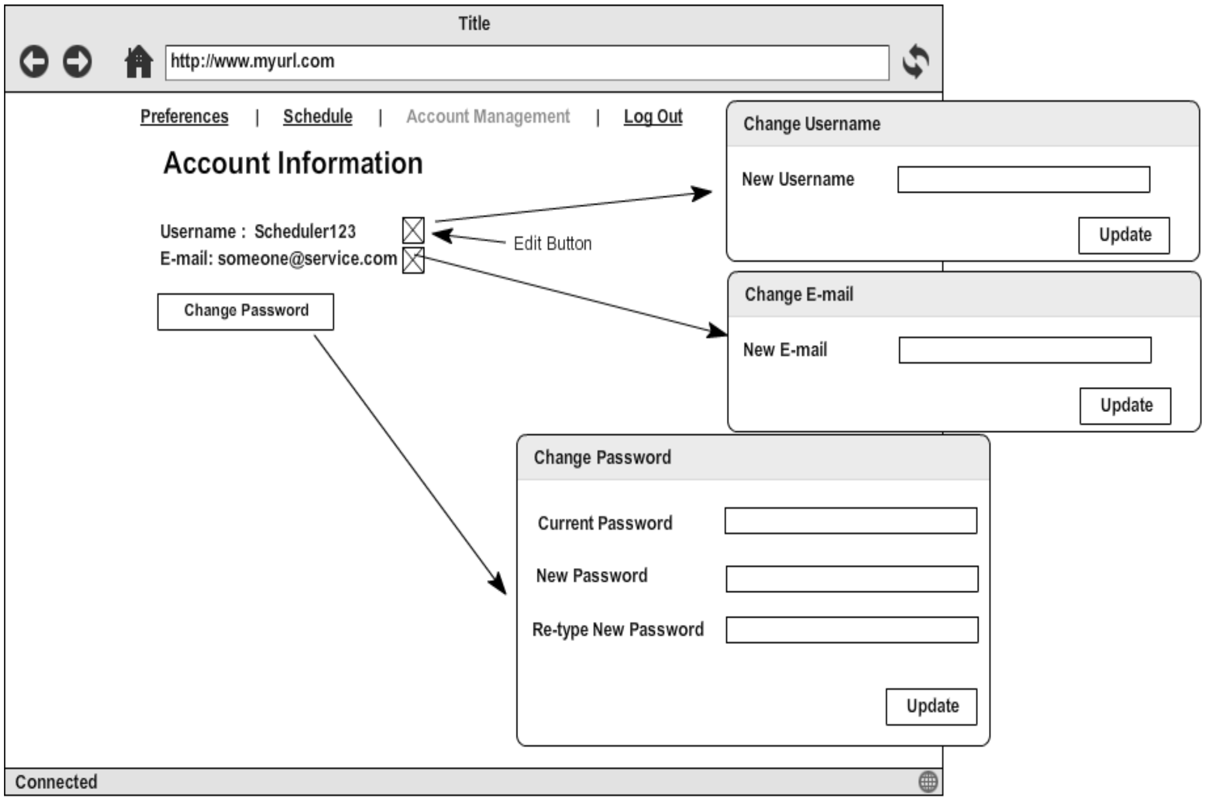
Name of Next Semester (ie Summer 2016)

Class Name	Course number	maybe other course info
Class Name	Course number	maybe other course info
Class Name	Course number	maybe other course info
Class Name	Course number	maybe other course info

Name of Next Semester (ie Fall 2016)

Class Name	Course number	maybe other course info
Class Name	Course number	maybe other course info
Class Name	Course number	maybe other course info
Class Name	Course number	maybe other course info

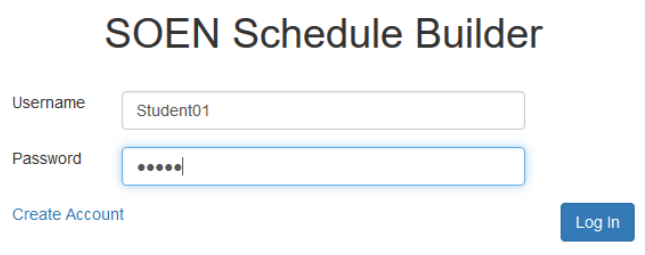
8.1.4 Account Management Page

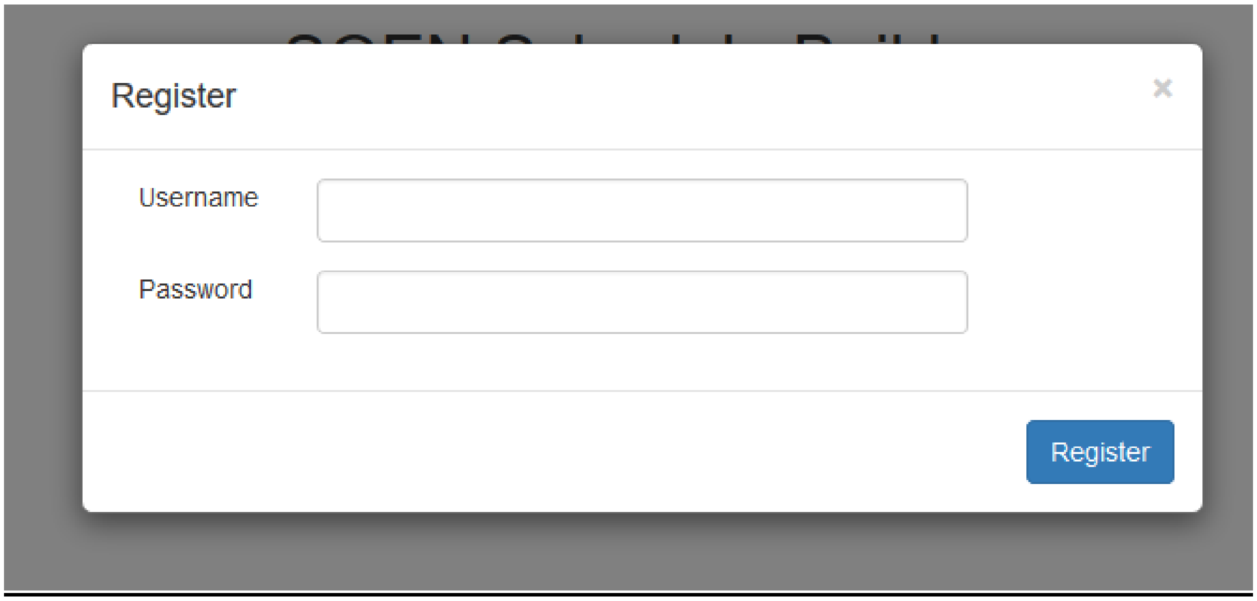


8.2 Prototype

To prove that the different frameworks work together, we decided to just have registration and log-in functionality, with the basic structure of the UI for the first two pages. This allows for some basic communication between the front and back end.

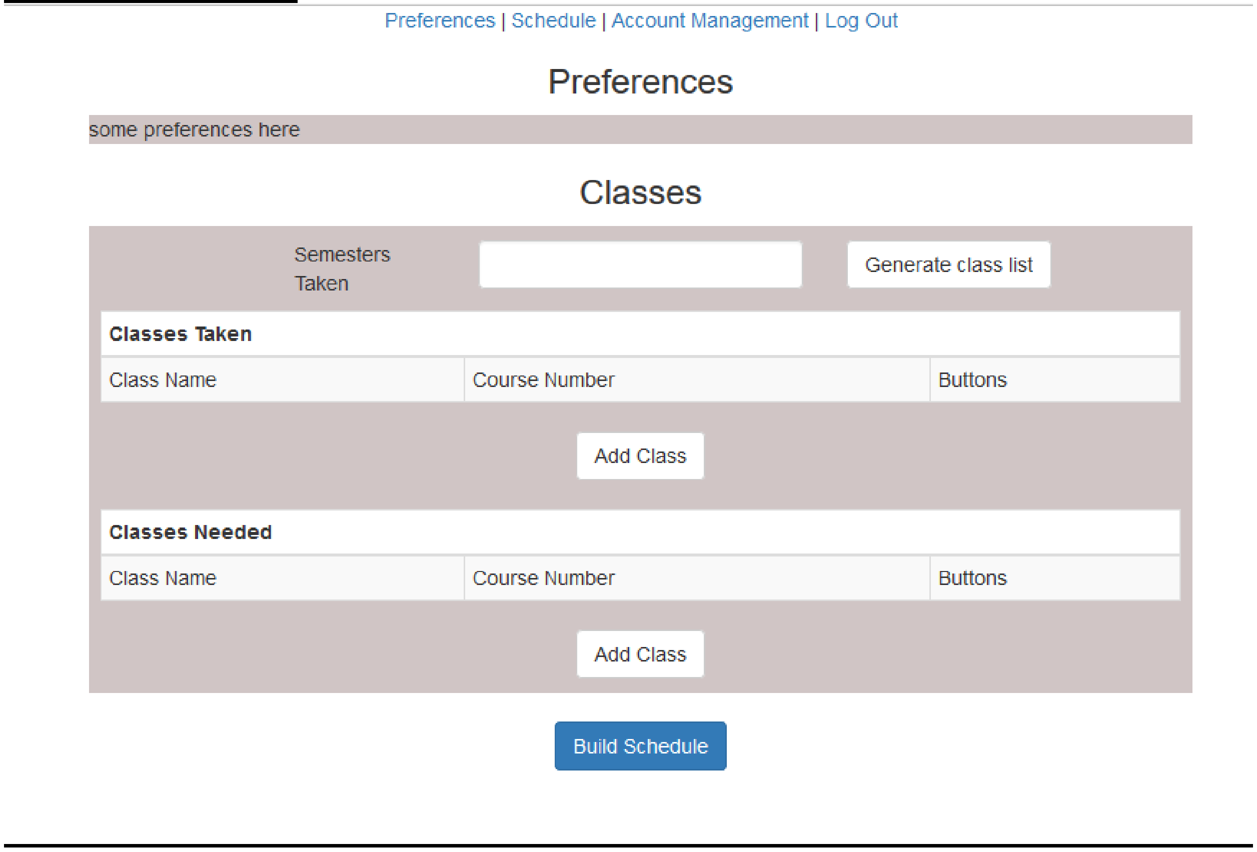
8.2.1 Log-In Page





The registration dialog on the log-in page

8.2.2 Preference Page



Classes Taken		
Class Name	Course Number	Buttons

Classes Needed		
Class Name	Course Number	Buttons