Concordia University
Engineering and Computer Science

Software Process, SOEN 341/4 S, Winter 2016
Dr. Shang
Dr. Fancott
Mr. Morse

TimeTurner by team YAWD

Project Scope and Plan Document – Deliverable 1

Team members information	
Name	**SID**
Daniel Di Corpo	**26331602**
Dimitri Topaloglou	**29358269**
Claudia Della Serra	**26766048**
Philip Lim	**27485506**
Aline Koftikian	**27764162**
Ryan Lee	**27752504**
Erin Benderoff	**27768478**
Ideawin-Bunthy Koun	**26314155**
Kevin Yasmine	**27195346**
Marc-Andre Leclair	**27754876**
Lori Dalkin	**27738293**
Bryce Drewery-Schoeler	**27283199**

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

TABLE OF CONTENTS

1. Introduction

Every year, newly admitted students at Concordia University are delegated a course sequence based on their matriculated program. Students are required create a schedule for themselves based on this course sequence and follow it thoroughly until the end of their degree. New students doing this on their own for the first time, and even experienced students, may encounter a number of difficulties when making schedules to fit this sequence, such as struggling to meet proper course prerequisites and managing time conflicts amongst multiple courses. This process is often long, tedious, and takes away from valuable time students may instead spend studying, working, or engaged in extracurricular activities. The following project, developed by a group of 12 undergraduate Software Engineering students at Concordia University, was created to simplify this process.

TimeTurner will be a web application implemented by Concordia University in order to aid students in planning and creating a schedule of courses for the duration of their degrees. Important factors to be kept in mind throughout the creation of this software system are ease of use, efficiency, speed, and overall effectiveness of the application in making scheduling of classes quick, easy, and personalized to students' needs. Students will be given a sequence of classes recommended to them by their faculty, much the same as current students in the Faculty of Engineering and Computer Science are given at the start of their degrees. Based on this sequence and each student's preferences, personalized schedules will be generated for the students' entire degree, thus saving the students much time and energy.

2. Project Description

The proposed and outlined web application, known as TimeTurner, is designed to auto-generate a student's course sequence from their first semester up until the end of their degree. It takes into account user input preferences and and any previously completed courses or course prerequisites before creating this sequence. Preferences can be made by the student and include options such as night classes or having particular days off. The application will notify the user if a certain preference suggested results in an impossibility or conflict in the sequence. This sequence generator will be able to create a sequence at any point throughout the user's degree, if sudden change in circumstances were to arise.

The goal of this application is to simplify the method with which students may decide and schedule their courses. If a course must be redone, the generator can decide what other courses should move where in regards to the remaining courses to be completed, which can be done in seconds, rather than hours. It saves the time of the user, in a simple and efficient manner. Ultimately, the system's end goal will be to simplify a student's task of creating their own course schedule in order to allow students to redirect their time to other more important activities, thus making course registration much simpler, quicker, and easier.

3. Goals and Constraints

3.1 Functional Requirements

3.1.1 Login

This Use Case enables any user to log on onto the scheduler to access their own profile to then effectuate whatever they need to in order to accomplish what they initially needed to. The log in action requests the attention of the server to pull the information that the user requested. This is done by the user entering his personal information (ID in this case) and his password in the password box. This, if done correctly, will allow the user to see his account information such as his generated schedule.

3.1.2 Logout

When the user is done his work on his TimeTurner profile, he or she will be able to log out from their account, cutting all their connection with any data associated to their username. This means that the user will have to go through the log in action again if he or she wants to access his personal information such as schedule.

3.1.3 Create Course Sequence

Once a student is logged in, they may create a course sequence. This course sequence will prompt the student for preferences upon when they want their classes, how many classes they desire per semester, etc. For instance, this could include morning versus evening classes. The course sequence will include the following four years (assuming the user is a student who just started his or her undergraduate program). Once the course sequence is created, a student will be able to access it to generate a schedule.

3.1.4 Browse Course List

This use case allows the user to browse a list of all relevant courses from the course calendar. The user has to log into the system and select the ‘Browse Course List’ option. The system must display a page containing all the courses in order by name. The user can scroll down the page and view any courses he or she is interested in with their description.

3.1.5 View Chosen Courses

This use case allows the user to view his or her program’s course sequence according to the user’s preferences or the generic course sequence if a schedule has not yet been generated. The user needs to log into the scheduler and select the ‘View Course Sequence’ option. The system will then display a page which contains the course sequence for the next four years that the student has to follow.

3.1.6 Generate Schedule

This use case allows the system to generate a schedule by first allowing the user to select his or her preferences. Once selected, the system will generate a schedule based on these preferences and the student's records, such as previously enrolled courses, prerequisites and co-requisites as well as currently enrolled courses. The schedule will also be kept inside the system once it has been generated.

3.1.7 View Saved Schedules

This use case allows the user, a student, to view a previously generated schedule that was saved to the system, either during the current session or a previous session. For this use case, the user must be logged into the system and have at least one schedule saved to the system. First, the user views a list of their saved schedules and selects which one they wish to view. Then, the system displays that schedule for the user to review.

3.1.8 View Academic Record

This use case allows the user, a student, to view their academic record. The academic record includes all completed courses and courses in which the student is currently enrolled. The record displays the name, number of credits and grade (if completed) for each course. The use case requires that the user be logged into the system and have completed and/or be enrolled in at least one course. First, the user requests to view their academic record. The system then produces the record for the user to view.

3.1.9 Remove Course from Schedule

This use case allows the user, a student, to remove a course from their generated schedule. It requires that the user be logged into the system and have generated a schedule containing one or more courses. First, the user selects which course they would like to remove from the schedule. Next, the system prompts the user to confirm if they are sure they wish to remove the course. Once the user confirms the removal, the system removes that course from the schedule and generates a modified schedule which does not contain the deleted course.

3.1.10 Add Course

This use case allows a student to search for and manually add a course to their schedule. The student searches for a course and selects a possible section from what is available to them. The system verifies that the student has completed all prerequisites required to take the course selected. If the student is eligible, the course will be added to the generated schedule.

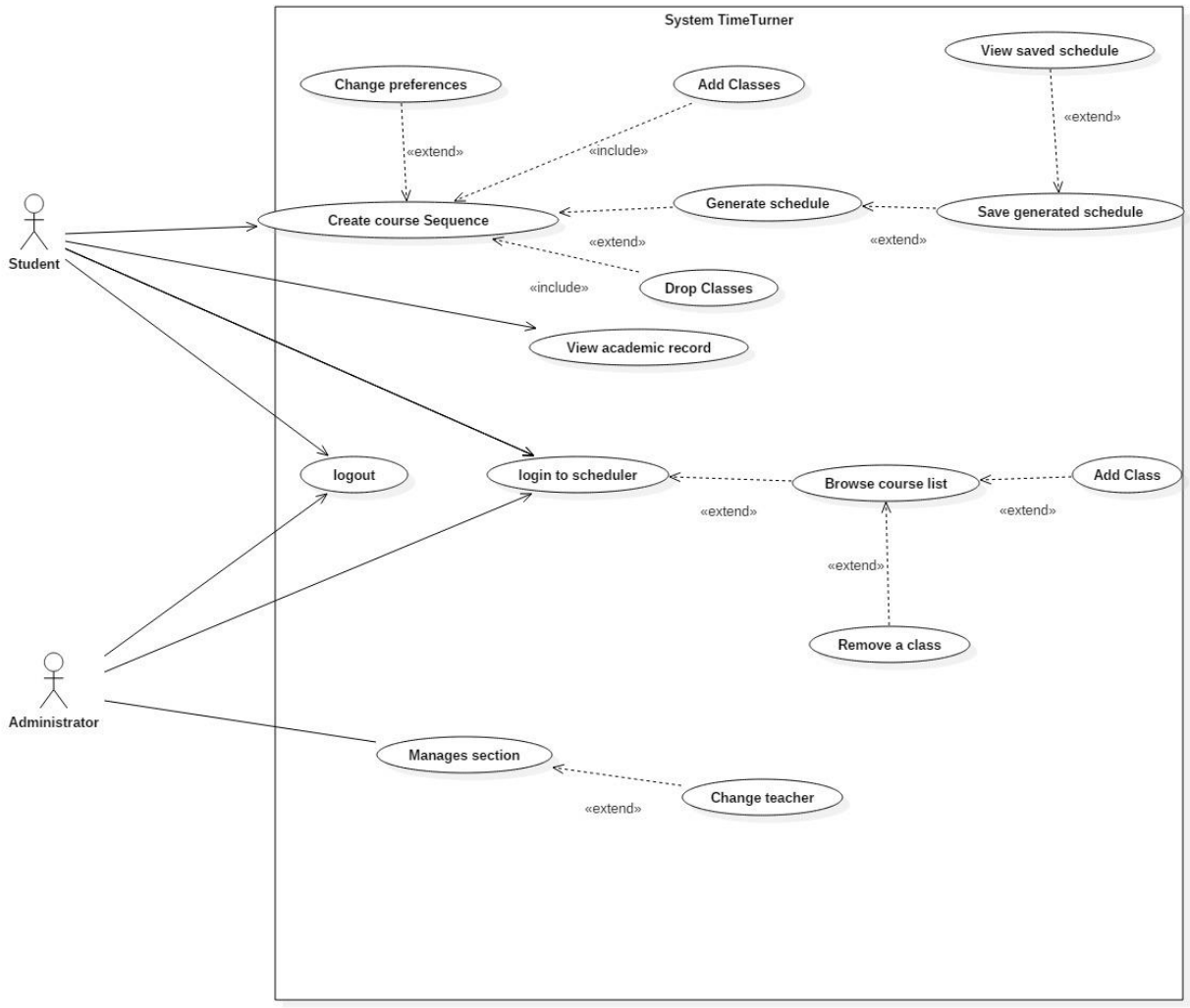
3.1.11 Save Generated Schedule

This use case allows a student to save a schedule they have generated to their accounts. After generating a schedule, the student can request the system save their generated schedule. The system will then record the information pertaining to the desired schedule and save it with the user records. The student can then access this schedule at a later time.

3.1.12 View Generated Schedule

The View Generated Schedule use case allows a student user to view their schedule in a weekly format. The time at which each class starts and ends will be displayed in a visibly appealing manner. Each class will have the teacher, room number, type of class, and course code displayed as well. The default schedule shown would be that of the current week.

3.1.13 Use Case Diagram



3.1.14 Use Cases

Use Case ID:	UC1	
Use Case Name:	Login to Scheduler	
Created By:	Marc-Andre Leclair	Last Updated By: Claudia Della Serra
Date Created:	January 25th, 2016	Last Revision Date: February 8, 2016
Actor(s):	Student, Administrator	
Goal/Actor Goals:	A student or Administrator wants to Login in their account	
Description/Summary:	The Student or Administrator wants to login into their account. The initial webpage contains a login box where the he or she can enter their username and password. The request is sent and it is check in the database whether or not the account exist or if the password is correct	
Preconditions:	❖ The user has an internet connection. ❖ The user has valid login credentials.	
Post-conditions:	The user is authenticated and logged in.	
Minimum Guarantee:	User will not log in	
Basic Flow:	1. Student enters its login information 2. Server verifies if it is in the database 3. Student is logged in.	
Risk assessment:	Low	
Importance assessment:	5/5	

Use Case ID:	**UC2**	
Use Case Name:	Logout of Scheduler	
Created By:	Marc-Andre Leclair	Last Updated By: Claudia Della Serra
Date Created:	January 25th, 2016	Last Revision Date: February 8, 2016
Actor(s):	Student, Administrator	
Goal/Actor Goals:	A student or Administrator wants to Logout of their account.	
Description/Summary:	The Student or Administrator wants to logout of their account. He or she sends a request to the server to close the connection between themselves and the Time Turner.	
Preconditions:	The user is logged out of his account	
Post-conditions:	The user has a personalized course sequence for the future	
Minimum Guarantee:	User will remain logged in	
Basic Flow:	1. The user is already logged in his account. 2. The user indicates that they wish to logout of the system. 3. The user logs out through the webpage terminating the connection.	
Risk assessment:	Low	
Importance assessment:	5/5	

Use Case ID:	UC3	
Use Case Name:	Create Course Sequence	
Created By:	Marc-Andre Leclair	Last Updated By: Claudia Della Serra
Date Created:	January 25th, 2016	Last Revision Date: February 8, 2016
Actor(s):	Student	
Goal/Actor Goals:	A student wants to create a course sequence	
Description/Summary:	The Student wants to create his or her course sequence. The system must provide the user with the option to add their own preferences in order to generate a sequence that is personalized to these preferences.	
Preconditions:	❖ The user is logged into the system ❖ The user is registered in a program.	
Post-conditions:	The user has a personalized course sequence for the future	
Minimum Guarantee:	No course sequence will be created	
Basic Flow:	1. Student requests to make a change to his or her sequence 2. The system must prompt the student to add his or her preferences to the sequence. 3. The student selects zero, one, or more preferences. 4. The system prompts the student to confirm their preferences 5. The student confirms his choices. 6. The course sequence is updated and shown to the student.	
Risk assessment:	High	
Importance assessment:	5/5	

Use Case ID:	UC4	
Use Case Name:	Browse Course List	
Created By:	Ideawin-Bunthy Koun	Last Updated By: Claudia Della Serra
Date Created:	January 25th, 2016	Last Revision Date: February 8, 2016
Actor(s):	Student	
Goal/Actor Goals:	Browse the list of courses from the course calendar.	
Description/Summary:	The user wishes to view available courses and access information pertaining to them. The system must display such information, including sections, times, and locations.	
Preconditions:	❖ User must be logged in as a Student. ❖ Course list must be available.	
Post-conditions:	A list of courses is displayed.	
Minimum Guarantee:	The system fails to display a list of courses and displays an error message to the user.	
Basic Flow:	1. Student logs into the Scheduler. 2. User selects the 'Browse Course List" feature. 3. System displays list of courses. 4. User selects a course from that list. 5. System displays information about the selected course.	
Risk assessment:	Medium	
Importance assessment:	3/5	

Use Case ID:	**UC5**	
Use Case Name:	View Course Sequence	
Created By:	Ideawin-Bunthy Koun	Last Updated By: Claudia Della Serra
Date Created:	January 25th, 2016	Last Revision Date: February 8, 2016
Actor(s):	Student	
Goal/Actor Goals:	Allows the user to view their expected progression in the program.	
Description/Summary:	User can view a complete course sequence of the program the student is enrolled in. The Sequence will show the courses the Student is expected to take, and the order he or she is expected to take them in.	
Preconditions:	❖ User must be logged in as a Student ❖ The system has access to a course sequence.	
Post-conditions:	The system displays a course sequence.	
Minimum Guarantee:	The system fails to display a course sequence and displays an error message to the user.	
Basic Flow:	1. Student logs into the Scheduler. 2. Student selects the option to view his or her course sequence. 3. Course sequence is displayed to the student.	
Risk assessment:	Medium	
Importance assessment:	4/5	

Use Case ID:	UC6	
Use Case Name:	Generate Schedule	
Created By:	Ideawin-Bunthy Koun	Last Updated By: Claudia Della Serra
Date Created:	January 25th, 2016	Last Revision Date: February 8, 2016
Actor(s):	Student	
Goal/Actor Goals:	Obtain a suggested schedule for the next four years.	
Description/Summary:	The system can generate a 4-year schedule for the user based on the student's preferences and constraints and on prerequisite and co-requisite policy.	
Preconditions:	❖ User must be logged in as a Student. ❖ The system must have access to course database. ❖ The system must have access to the student's records.	
Post-conditions:	The system generates a schedule with no overlaps.	
Minimum Guarantee:	The system fails to generate a schedule and displays an error message to the user.	
Basic Flow:	1. User opens a session with the scheduler. 2. User selects the 'Generate Schedule' feature. 3. User selects his or her options and preferences. 4. System responds by displaying a schedule that meets the user's preferences and constraints.	
Risk assessment:	High	
Importance assessment:	5/5	

Use Case ID:	UC7	
Use Case Name:	View Saved Schedules	
Created By:	Erin Benderoff	Last Updated By: Claudia Della Serra
Date Created:	February 7, 2016	Last Revision Date: February 7, 2016
Actor(s):	Student	
Goal/Actor Goals:	The user wishes to view a previously saved schedule.	
Description/Summary:	The user selects a schedule from a list of their previously saved schedules. The system must display this schedule for the user to review.	
Preconditions:	❖ User is logged into the system. ❖ User has previously generated one or more schedules ❖ User has saved at least one generated schedule	
Post-conditions:	The system displays the saved schedule	
Minimum Guarantee:	Saved schedules remain in the system unchanged.	
Basic Flow:	1. The user requests to view saved schedules 2. The user selects which saved schedule to view 3. The system displays the chosen schedule	
Risk assessment:	Medium	
Importance assessment:	3/5	

Use Case ID:	UC8	
Use Case Name:	View Academic Record	
Created By:	Erin Benderoff	Last Updated By: Claudia Della Serra
Date Created:	February 7, 2016	Last Revision Date: February 9, 2016
Actor(s):	Student	
Goal/Actor Goals:	The user wishes to view a list of his or her completed courses and grades for those courses.	
Description/Summary:	The system displays the user's academic record, which includes completed courses and courses currently being taken.	
Preconditions:	❖ User is logged into the system. ❖ User has completed at least one course and/or is presently enrolled in at least one course.	
Post-conditions:	The system displays the user's academic record.	
Minimum Guarantee:	The academic record remains in the system unmodified.	
Basic Flow:	1. The user requests to view their academic record 2. The system generates a list of the user's completed courses, as well as courses in which the user is currently enrolled.	
Risk assessment:	Low	
Importance assessment:	1/5	

Use Case ID:	UC9	
Use Case Name:	Drop Course	
Created By:	Erin Benderoff	Last Updated By: Claudia Della Serra
Date Created:	February 7, 2016	Last Revision Date: February 7, 2016
Actor(s):	Student	
Goal/Actor Goals:	The user wishes to remove a course from a generated schedule.	
Description/Summary:	The user selects which course he or she would like to remove from the schedule. The system must delete this course from the schedule.	
Preconditions:	❖ User is logged into the system. ❖ User has generated a schedule containing at least one course.	
Post-conditions:	The chosen course has been removed from the user's schedule.	
Minimum Guarantee:	The courses on the user's schedule remain unchanged.	
Basic Flow:	3. The user chooses a course from his or her schedule to remove. 4. The system prompts the user to confirm the course removal. 5. The user confirms the course removal. 6. The system removes that course from the schedule 7. The system produces a modified schedule without the removed course.	
Risk assessment:	High	
Importance assessment:	4/5	

Use Case ID:	UC10	
Use Case Name:	Add Course	
Created By:	Lori Dalkin	Last Updated By: Claudia Della Serra
Date Created:	February 7, 2016	Last Revision Date: February 7, 2016
Actor(s):	Student	
Goal/Actor Goals:	Add a course to a generated schedule.	
Description/Summary:	Specific courses can be added to a schedule generated by a user. The user must search for the course and manually add it to their schedule.	
Preconditions:	❖ The user is logged in to their account ❖ A schedule has been generated	
Post-conditions:	The specified course is added to the schedule.	
Minimum Guarantee:	The user's schedule remains unchanged and no new course is added	
Basic Flow:	1. User searches for a course they wish to add. 2. User selects the specific course they wish to add. 3. User tells the system to add the selected course. 4. The system returns a modified schedule.	
Risk assessment:	Medium	
Importance assessment:	3/5	

Use Case ID:	UC11	
Use Case Name:	Save Generated Schedule	
Created By:	Lori Dalkin	Last Updated By: Claudia Della Serra
Date Created:	February 7, 2016	Last Revision Date: February 7, 2016
Actor(s):	Student	
Goal/Actor Goals:	The user wishes to save a schedule they have generated.	
Description/Summary:	A generated schedule can be saved in order to be accessed and viewed later on by the user. This schedule consists of courses the user is satisfied with and wishes to keep on their schedule.	
Preconditions:	❖ The user is logged on to their account ❖ A schedule has been generated containing at least one course.	
Post-conditions:	The system saves the generated schedule to the user's account.	
Minimum Guarantee:	The user's account remains unchanged.	
Basic Flow:	1. The user will generate a schedule 2. When satisfied with the schedule, the user will tell the system to save it. 3. The system will store the generated schedule in the user's file.	
Risk assessment:	Medium	
Importance assessment:	4/5	

Use Case ID:	UC12	
Use Case Name:	View Saved Schedule	
Created By:	Bryce Drewery Schoeler	Last Updated By: Claudia Della Serra
Date Created:	February 7, 2016	Last Revision Date: February 7, 2016
Actor(s):	Student	
Goal/Actor Goals:	A student wants to view generated schedule.	
Description/Summary:	The student wishes to see their class schedule. The student will specify a week to be displayed. The system must display the schedule for that week.	
Preconditions:	❖ The student has been authenticated ❖ A schedule has been generated and saved	
Post-conditions:	The schedule is shown on screen.	
Minimum Guarantee:	The schedule fails to be displayed.	
Basic Flow:	1. Student requests to see the schedule. 2. The system prompts the student to select a week. 3. Student selects a week 4. The systems display the schedule for that week	
Risk assessment:	Medium	
Importance assessment:	3/5	

3.2 Domain Model

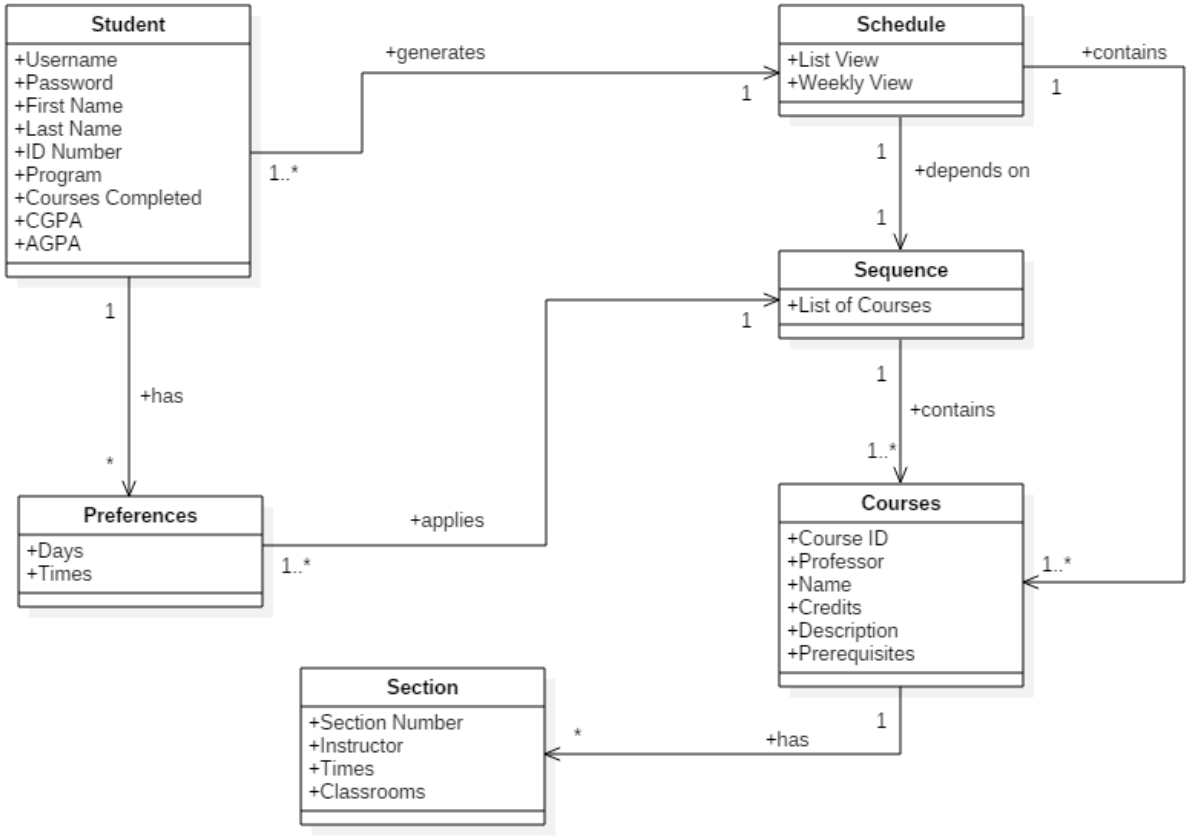


Figure 2. Domain Model

3.3 Constraints and Qualities

3.3.1 Security

Security is a crucial aspect of any software. One of the most important security requirements is to make sure there are no SQL injections that could breach and break our database. In order to assist in preventing SQL injections, we will be using prepared statements prior to query executing which is provided by PHP's internal libraries. The possibility of having the database breached could potentially expose user personal user information or perhaps destroy content. The exposure of passwords is a concern in case of breach as they should be hashed and salted upon entry by using hashing algorithms such as *sha512* and *bcrypt*. This would essentially diminish any chance hackers have to brute force password entries.

3.3.2 Performance

Performance is important especially when working with large data sets. The user must be able to navigate and use the site without having to wait for data generation. The application shall also ensure that database design is robust with the use of proper table indexing and storage engines and cache sizes. This will essentially allow for quicker SQL query executions that are expected to return data within 100 nanoseconds. The performance shall also account for larger data execution, such as the schedule generator with a performance factor speed of no more than 5 seconds.

3.3.3 Cross-Browser Compatibility

There are cases where web applications do not necessarily comply with all possible web browsers. For example, certain aspects of CSS and HTML might work for one browser but break in another. This application shall ensure that the CSS and HTML model will be compliant with the browser's boundaries. Ensuring browser compliance, the application shall be able to run on virtually any operating system that has any of the following browsers: Safari, Opera, Internet Explorer, Firefox and Chrome.

3.3.4 Ease of Use

The application shall be straightforward and lightweight in order to ensure that students can start working on their schedule planner immediately. The use of Ajax technology will also minimize content and page refreshes thereby allowing users to navigate without losing the scope of the data they were working on.

4. Resource Evaluation

4.1 Human Resources

Our team of Undergraduate Students is a strong team of 12 individuals with varying skills and strengths. Our team has a diverse background, both socially and in terms of past work experiences. We also possess a good gender ratio of 7 males and 5 females. With both knowledgeable leaders and dedicated team players, we work efficiently together while also complementing each other's strengths and weaknesses. Each member of our team brings their

own outlook towards the project that will ultimately create a well-rounded project that can fit the needs of many. Below are the names of the 12 members along with the following information:

- The different skills and knowledge of languages they possess.
- Their experiences with past occupations, school projects and personal projects. Other relevant experiences that can add insight towards the project.
- Their primary role during the project.
- Their strengths that will help towards the success of the project.
- Their approximate availability per week, including both group meetings and personal work time.

Name	**Daniel Di Corpo**
Skills	C++, JAVA, PROLOGUE, LISP, PHP, HTML
Experience	Technical DEC in Comp. Sci., Image Programing Specialist at Matrox
Role	Project Team Lead, System Architecture and Distribution of Tasks
Strengths	Leadership, Team Player, Management skills, Communication
Availability	7 hours/week

Name	**Dimitri Topaloglou**
Skills	PHP, CSS, C++, C#, HTML, SQL, JAVASCRIPT, JQUERY, JAVA, PROLOGUE, CLOS, ASP.NET
Experience	Seven years of experience in web application development.
Role	Coding team Technical Lead, system designer, frontend + backend developer
Strengths	Advanced coder, quick learner, opened to new ideas
Availability	7 hours/week

Name	Claudia Della Serra
Skills	HTML, CSS, JAVA, JAVASCRIPT, PROLOGUE, LISP, CLOS, ASPECTJ, PYTHON
Experience	Developed 2 Android applications, certified tutor in English reading and writing, multiple years of experience in document editing.
Role	Documentation Team Lead, writing, editing and consolidation of documents
Strengths	Quick learner, team player, leadership and management skills
Availability	7 hours/week

Name	Aline Koftikian
Skills	JAVA, HTML5, CSS3, JAVASCRIPT, C#, PROLOG, ASPECTJ, COMMON LISP, PYTHON
Experience	Developed two mobile (Android) applications w/ teams.
Role	Documentation team member, in charge of diagrams + logo
Strengths	Team player, open minded, organized
Availability	7 hours/week

Name	Kevin Yasmine
Skills	JAVA, HTML5, CSS3, JAVASCRIPT, PROLOG, ASPECTJ, COMMON LISP, CLOS, Photoshop/Paint.Net
Experience	Class projects/assignments, extracurricular Android App Development using Android Studio.
Role	Documentation team member
Strengths	Team player, ability to see the big picture in large projects
Availability	7 hours/week

Name	Ideawin-Bunthy Koun
Skills	JAVA, HTML5, CSS3, PHP, JAVASCRIPT, PROLOG, ASPECTJ, COMMON LISP
Experience	Academic projects/assignments, attended two hackathons where Android Studio was used, developed Android applications.
Role	Coding team member as a Front-End developer
Strengths	Fast learner, eager to learn new technologies, team player
Availability	7 hours/week

Name	Philip Lim
Skills	JAVA, HTML5, CSS3, JAVASCRIPT, PHP, MYSQL, PROLOG, COMMON LISP
Experience	Completed an internship as a programmer analyst: involved Java programming using IntelliJ IDEA and documenting the code Developed an apartment renting website (academic project) Completed various academic assignments/projects in Java
Role	Documentation team member
Strengths	Flexible, active listener, team player
Availability	7 hours/week

Name	Ryan Lee
Skills	JAVA, HTML, CSS, JAVASCRIPT, PHP, PROLOG, LISP, C#, PYTHON, RUBY
Experience	Competed in 2 hackathon and made applications using java and android studio, implementing APIs provided by companies.
Role	Documentation team member
Strengths	Team player, Communicative abilities, problem solving
Availability	7 hours/week

Name	Lori Dalkin
Skills	PHP, CSS, HTML, JAVASCRIPT, JAVA, PROLOGUE, CLOS, ASPECTJ, PYTHON(basic), C#(basic)
Experience	Attended two hackathons where two android apps using java and android studio were developed.
Role	Coding team member. Back end and front end developer.
Strengths	Quick learner, enthusiasm for learning new languages, creative.
Availability	7 hours/week

Name	Bryce Drewery Schoeler
Skills	PHP, CSS, HTML, JAVASCRIPT, JAVA, PROLOGUE, ASPECTJ, PYTHON, C++, C#
Experience	Developed android apps at hackathons.
Role	Coding team member. Back end and database developer.
Strengths	Enthusiastic learner, Good grasp of theoretical computer science. Excellent at solving logical problems.
Availability	7 hours/week

Name	Erin Benderoff
Skills	PHP, CSS, HTML, JAVASCRIPT, JAVA, PROLOG, CLOS, ASPECTJ, PYTHON (beginner), MySQL
Experience	Made two Android applications at hackathons
Role	Coding team member. Mainly front end but will also do a bit of back end.
Strengths	Detail-oriented, motivated, fast learner, easy to get along with
Availability	7 hours/week

Name	Marc-Andre Leclair
Skills	PHP, CSS, HTML, JAVASCRIPT, JAVA, PROLOG, RUBY, C#, Visual Basic.net, MySQL
Experience	Created two android app at two different hackathons
Role	Coding team member. Mostly back end attending to php and the database but also some front-end
Strengths	Love to learn, strong team player, fast learner
Availability	8 hours/week

4.2 Technical Resources

Each developer's computer must have the same XAMPP version installed in order to align the Apache and PHP version as found in the production environment. This is to avoid any conflicts and bugs which can be caused due to incompatible versions of PHP or Apache's server core files.
XAMPP (64-bit environment) includes the following pre-installed software packages:

- Database: MySQL 5
- Web Server: Apache HTTP Server 2.0
- Primary language: PHP 6.0 and Python 2.7
- Database User Interface: phpMyAdmin

The production environment in question has the following infrastructure:

- CPU: Intel i7
- OS: Windows Server 2012
- RAM: 16GB
- DISK DRIVE: 500 GB (Solid State)
- Network Speed: 10 Mbps

The production environment is hosted on a team member's own computer. This decision was made in order to have a better overlook of the server and have complete control of the environment.

Main software packages used in the project include:

- Languages: HTML, CSS, PHP, JQuery, JavaScript, SQL
- IDE: PhpStorm 10
- Framework: Yii 1.1.16
- Version Control Systems: GitHub, Gitbash, GitHub Desktop
- Team Collaboration Tools: GitHub, Facebook Chat, Trello

- OS: Windows, Linux
- UML and Diagrams: StarUML
- Documentation tools: Google Drive, Microsoft Word
- Graphical Design Tool: Photoshop

5. Scoping

When the Project was first announced and the groups were formed, the team spent a lot of time discussing what the Scheduler should include. Many features came up, such as administrative privileges, manual manipulation of the sequence, etc. However, after working on the plan and blueprints of the project, it became clear that there are many requirements and constraints that would limit the additional features that we wanted to include.

5.1 Limitations

Taking into consideration section 3 and 4, our biggest limitation is time. Some features would be an asset to have, but given that the project should be developed in only a few weeks, it is impossible to include them all while maintaining Security, Performance, Cross-Browser Compatibility and Ease of Use (see section 3.3).

5.2 Scoped Out

5.2.1 Administrative Privileges

Initially, the team wanted to include Teachers and Administrators that are able to do different tasks, such as change class time, drop students, add students, change teachers, bypass prerequisites, etc., but given our time constraint and our limited experience as students, this aspect was scoped out. Such activities include

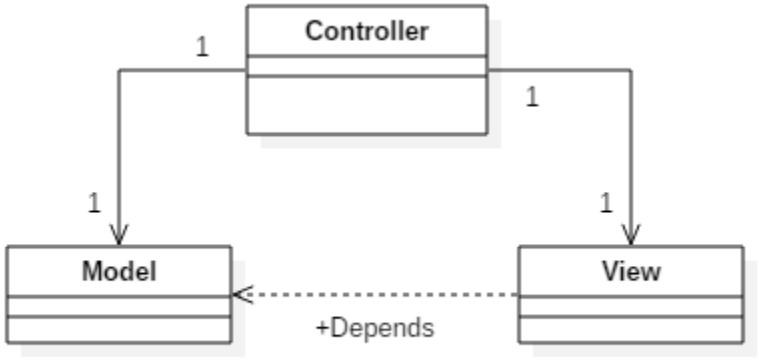
- Managing a Section
- Adding a Course to the Course List
- Changing the Time of a Course
- Etc.

5.2.2 Manual Manipulation of Sequence

Another feature that was scoped out, is to allow students do add and drop individual classes. The team realized that the main function of the Scheduler is to automatically create a full, four-year schedule. We decided to optimize this function so that it runs well, instead of giving multiple – less efficient – ways of adding classes.

6. Solution Sketch

6.1 Architecture



Time Turner's architecture will follow a Model-View-Controller architecture (MVC). This MVC architecture is best suited for this project because it isolates our application to 3 different layers, data, logic and presentation. With these clear separation of concerns, changing any of these components might not affect the other two components. This is a preferred for team based environments since all 3 components can be coded at the same time.

We have chosen to use the Yii Framework for its ease of use, modular components and its exceptional handling of applications with big traffic flow such as portals and content management systems. We have chosen this framework for its high speed of development and work, integration with Javascript, can handle multiple types of databases and convenient and flexible caching system which will help for caching validation.

A general work flow starts off with a user request. This is handles by the controller, which will separate what needs to be sent to the model component and the view component. Anything that needs data retrieval or sequence generation, will be sent to the model component, which is in conjunction with our domain model. Once the data has been retrieved, it is sent back to the controller to be packaged and sent to the viewer. Once the view component has all the information, it will generate, what will be, in the end, the display on the user's web browser. Then, once a new request is sent, the process will start over again.

6.1.1 Controller

This is the data of the MVC. With a web application, the user will make a request for a web page which will be handled by the controller. The controller will evaluate the request, sending any logic operations to the model component. Once the controller has collected all of the necessary information, it will send all its data to the view component, which will generate an HTML output to the user.

6.1.2 Model

This is the logic of the MVC. With the user information given from the controller, and with the use of the underlying logical data structure, such as our domain model, processes such as, generate sequence, can be utilized and sent back to the controller. One controller does not have exclusivity over the model component, since multiple pages may need the same information, many controllers can access the model component.

6.1.3 View

This is the presentation of the MVC. With the proper session handling from the controller and relevant data from the modeller, the viewer can generate an HTML output with CSS and Javascript, which will, in-turn, be viewed on the user's web browser.

6.2 Technologies in Use

6.2.1 Programming Languages

6.2.1.1 Server-side

6.2.1.1.1 PHP

PHP is the main server-side language that will be handling business logic and SQL database queries. PHP was chosen due to the syntax similarities of other major Object-Oriented programming languages, allowing for a quicker learning experience.

6.2.1.2 Client Side

6.2.1.2.1 JavaScript

JavaScript a multi-paradigm scripting language that is supported and natively embedded by all major browsers including Internet Explorer, Firefox, Safari and Chrome. It is mainly used for dynamic client side manipulation of the DOM and allows for a dynamic browsing experience.

6.2.1.2.2 jQuery

jQuery is essentially a JavaScript script library that allows programmers to write less lines of code in order to perform the same functionality in the native JavaScript environment. It allows for quick DOM manipulation with very few lines of code, animations, client side validation, form input handling and quick Ajax calls.

6.2.1.2.3 HTML (Hyper Text Markup Language)

HTML is the primary markup language used to create web layout and provide structure to web content. Of course, it is also natively supported in every major web browser.

6.2.1.2.4 CSS (Cascading Style Sheet)

CSS is used primarily for styling purposes that is supported by all major browsers with some detailed exceptions in portability. The essence of CSS is to separate styling from the markup language, such as HTML.

6.2.2 IDEs

6.2.2.1 PhpStorm 10

PhpStorm is a powerful IDE provided by JetBrains that is integrated with an extensive Git support system and provides direct database management without the use of any other third party application.

6.2.3 Database Management System

6.2.3.1 MySQL

MySQL is an open-source database management system that provides more than enough features required to for the scope of the web application. It also provides InnoDB storage engine access which supports transactions, enabling to undo database query commits.

6.2.4 Web Server

6.2.4.1 XAMPP (v1.8.1)

XAMPP 1.8.1 is a free open-source, cross-platform web server solution package which includes Apache HTTP Server, MySQL and PHP functionalities.

6.2.5 Source Code Revision Management

6.2.5.1 GitHub

GitHub is a popular source code revision management system that allows for developers to clone source code repositories and provide a non-destructive means of working in isolation from the original repository. In turn, this provides a better overview of which developer is working on which portion of the code, thereby providing a flexible version control workflow.

6.2.6 Deployment Software

6.2.6.1 DeployHQ

DeployHQ is an online service that allows automatic and manual deployment of sources code from any type of repository service such as GitHub. It is used to automatically deploy source code directly to the main hosting server from a selected GitHub branch of your choice in a seamless manner. The service is installed via a webhook service, integrated within GitHub and source code deployment occurs upon a commit done in the master branch. It also provides rollback options.

6.2.7 Team Collaboration

6.2.7.1 Trello

Trello is a free team collaboration tool that organizes task into board spaces. It provides an overview of what tasks need to be done, tracks due dates and assigns tasks to collaborators. It also provides a mobile app version which allows to push notifications to users' mobile when the project board was updated, allowing collaborators to respond to tasks accordingly.

6.2.8 Framework

6.2.8.1 Yii

Yii is a highly extensible, open source framework designed for web applications and is based on an MVC architecture. It shortens development time with the use of Gii, an incorporated tool that quickly creates on-the-fly templates for models, controllers, forms, modules, extensions and CRUD control actions and views. This allows developers to minimize repetitive tasks and enforces the MVC architecture according to Yii's specifications. It also comes with a pre-built

web template to allow unfamiliar developers to get a quick start in learning how to properly create models, views and controllers.

7. Plan

7.1 Activities

Activity	Purpose/Description	Artifact(s) produced
7.1.1 Form and divide team	Forming a team allows the project to start. Dividing the team into a coding and documentation sub-teams allows members to focus on particular tasks. Such division requires the evaluation of the strengths and weaknesses of all members to determine their role.	1.4.1 Human Resources
7.1.2 Complete the system overview	The system overview consists of finding a name for the system, determining the main purposes and elements involved in it, and drawing a domain model of the system along with the description of each entity found in the domain model.	Deliverable 0 1.3.2 Domain Model
7.1.3 Write a project description	The project description introduces the entire document. It informs the reader about its goals, the information to be found in it, as well as the purposes and objectives of the project.	1.2 Project Description
7.1.4 Determine the functional requirements and draw use case diagrams accordingly	Determine the requirements that the system should have, including those that may later be scoped out, their relative importance and relative difficulty. Define the actions that must take place in the software and illustrate each requirement by the mean of a use case diagram.	1.3.1 Functional Requirements
7.1.5 Review and correct domain model	Make any necessary changes according to the feedbacks of Deliverable 0.	1.3.2 Domain Model

7.1.6 Find and describe constraints and qualities	Describe any design constraints, qualities and non-functional requirements that the system should meet.	1.3.3 Constraints and Qualities
7.1.7 Review and format human resources document	After the profile of each team member has been determined, the document should be revised and formatted correctly.	1.4.1 Human Resources
7.1.8 Determine technical resources	List the computer resources and tools available to complete the project.	1.4.2 Technical Resources
7.1.9 Reduce the amount of features to be realistic according to time and resources	Outline the scope of the software. List all features, goals and qualities that are scoped out, and provide an explanation for each element that is scoped out.	1.5 Scoping
7.1.10 Choose the architecture of the system	Explain the high-level architecture of the system, and give a design rationale. A diagram should accompany the explanation.	1.6.1 Architecture
7.1.11 Decide which technologies are relevant to the project development	List technologies that will be used in the project and give a rationale for each of them.	1.6.2 Technologies in Use
7.1.12 List and describe the main activities involved in the project	List all activities that produce at least one artifact and include a description for each activity.	1.7.1 Activities
7.1.13 List all artifacts	List and describe all artifacts that need to be produced for the project.	1.7.2 Artifacts

7.1.14 Estimate the cost and time of production	Estimate the cost and time of production of each artifact and sum them up.	1.7.3 Project Estimates
7.1.15 Delegate activities to different team members	Assign each activities of the project to a team member or to a group of team members.	1.7.4 Activities Assignments
7.1.16 Produce a schedule for the project	Create a schedule/timeline of all activities for the project using a Gantt chart. Start dates, due dates, names of activities, and participants are all information to be included in the schedule.	1.7.5 Schedule
7.1.17 Determine the risks of the project	Outline and explain the risks associated and presented by this project.	1.7.6 Risk
7.1.18 Produce a prototype report	Describe the work done in the development of the prototype. Explain how the chosen technologies are appropriate, and how the team members are comfortable using such technologies.	1.8 Prototyping
7.1.19 Ensure professionalism of the document	Assemble all parts of deliverable 1. Ensure the document is professional looking and well-organized.	Deliverable 1
7.1.20 Update the introduction for Deliverable 2	Update the introduction to give the reader an overview of the content found in Deliverable 2: Architecture and Design	2.2 Introduction to Part 2
7.1.21 Produce an architecture diagram	Produce a 4+1 Architectural View. Include a rationale for the design. Explain any differences between the old design and this one.	2.3.1 Architecture Diagram

7.1.22 Specify the interactions between the components of the software	Describe the interactions between the components of the software through their interfaces. Include the function calls, the description of the parameters and the range of accepted values of those parameters.	2.3.2 Subsystem Interfaces Specifications
7.1.23 Illustrate the internal structure of the system	Provide a graphical representation of the structure of each subsystem by means of a UML class diagram. A description of each class should be included.	2.4.1 Detailed Design Diagram
7.1.24 Describe each class in the subsystem	Provide the programmers with the descriptions of each class in the UML diagram along with any necessary detailed relevant to the development.	2.4.2 Unit Description
7.1.25 Produce dynamic design scenarios	Draw the dynamic design of two use cases.	2.5 Dynamic Design Scenario
7.1.26 Update project estimates	Update the cost of the project for each module and include the cost of integration, testing and documentation.	2.6 Estimation
7.1.27 Produce a report about the prototype and risk	List and describe all elements that have been implemented and describe the effect that those implementations had on the design decisions, risks, estimate and scope.	2.7 Rapid Prototyping and Risk
7.1.28 Ensure professionalism of the document	Assemble all parts of deliverable 2. Ensure the document is professional looking and well-organized.	Deliverable 2
7.1.29 Update introduction	Update the introduction so that it includes an overview of the content presented in deliverable 3.	3.2 Introduction
7.1.30 List all tested items	Create a list of all items that have been tested, the test cases that were used, and a rationale for the test.	3.3.1.1 Tested Items
7.1.31 List all untested items	Create a list of items that are to be tested, an explanation of why those items should be tested, and how they could be tested.	3.3.1.2 Untested Items
7.1.32 Unit testing report	Describe the test cases, the technique used, the code used and the result of the testing for two units.	3.3.2.1 Unit Testing

7.1.33 Requirements testing report	Provide a list of test cases for all tested requirements by means of scenario of system usage and system reaction.	3.3.2.2 Requirements Testing
7.1.34 Stress testing report	Describe situations of extreme system usage, the tests designed to evaluate the performance of the system under such condition, and the test results.	3.3.2.3 Stress Testing
7.1.35 Security testing report	Describe tests performed to protect the system against security threats such as SQL injections and automated tools.	3.3.2.4 Security Testing
7.1.36 Installation manual redaction	Guide any administrators to install and execute the software.	3.4.1 Installation Manual
7.1.37 User's manual redaction	Guide any users to use the system with all of its features.	3.4.2 User's Manual
7.1.38 Update cost estimate	Update the table of costs with all components of all phases.	3.5 Final Cost Estimate
7.1.39 Ensure professionalism of the document	Assemble all parts of deliverable 2. Ensure the document is professional looking and well-organized.	Deliverable 3
7.1.40 Correction of various part of the deliverables	Make all necessary changes in the deliverables for the final report.	Corrected deliverables
7.1.41 Assemble all deliverables	Put all corrected/updated deliverables together for the final report. Ensure professionalism of the document.	Final report

7.2 Artifacts

7.2.1 1.2. Project Description

The project description is an introduction to the entire project. It's a brief that allows the reader and the user of the system to understand the goal and purpose of the project. This document must be presented as an opening to the rest of the manual, and must include the idea, objective, background, approach, and result of the entire project.

7.2.2 1.3.1 Functional Requirements

Requirements are the tasks that are necessary for the completion of the system. They are the components of the system that must be documented, measured and tested in order to ensure the ideal conclusion to the project in question. The requirements that need to be specified for this system are functional and non-functional requirements. The functional requirements are the tasks that define what the system is trying to accomplish. These are the elementary actions that constitute the main goal of the system. They include the user viewing weekly schedule, adding classes, dropping classes, swapping, etc.

Each individual requirement must be presented as a use case. A use case is a description of the behavior of a system that explains the process and results obtained through interactions of the user with the system. The functional requirements must be known before writing use cases, because they are the details that compose the main actions and tasks of the system. The use cases are present in both list form and diagram form.

- Use Cases: The list-based use cases are full formal documents that list all use cases of the system in detail, with each property of the use case explained in full detail. These include the name of the use case, the general description, the actors, preconditions, postconditions, etc.
- Use Case Diagram: Use case diagrams are used to depict a visual representation of the use cases listed. The actors, which typically include the user or an external entity, are depicted using stick figures. The use cases themselves are drawn as circles with a name that describes the title of the action that can be initiated by the actors. There are lines connected from users to the use cases called connections, which describe the actor's interactions with those use cases.

7.2.3 1.3.2. Domain Model

The Domain Model is the model that describes all core concepts and an overview of the system at hand. Its main purpose is to list the objects of interest that form the system, and presents their attributes, constraints, and relationships between them. The model is designed in UML format, with arrows describing relationships from one object to another, and ensuring that there are multiplicity values at both ends of the arrow denoting multiplicity factors of the relationship. The finalized domain model will come with a description that will provide a more detailed explanation of the model as well as each object present in the model.

7.2.4 1.3.3. Constraints and Qualities

The constraints and qualities segment describes the main standards of quality and non-functional requirements that are expected to be met by the system. The non-functional requirements are the general properties of the system. They are the constraints and qualities that will make up the system when in use. These include response times, maintenance times, security, etc. It's important that the constraints and qualities are as specific as possible, which means they must have concrete metric specifications wherever possible.

7.2.5 1.4.1. Human Resources

The human resources of a project is the group of individuals who compose the main workforce of the project. They are the source of labor of a project, and in the case of large projects, are typically divided into smaller groups for more organized work.

7.2.6 1.4.2. Technical Resources

The technical resources of a project are the computer resources that will be used to complete the project. This includes the computer programs, software, libraries, and websites that are available for use.

7.2.7 1.5. Scoping

The scoping section describes all elements, features, qualities, and goals that have been scoped out from the end result. Each scoped element must come with a description of what it is, in addition to a paragraph that describes why it was chosen to be scoped out. These scoped elements have been typically scoped out in the early phases of a project, where the requirements are being planned and discussed in between team members.

7.2.8 1.6.1. Architecture

The architecture of a project is the high-level system architecture that is planned for the complete project. It must be presented with a UML class diagram that describes the details and relationships between different components of the system, along with a detailed description of the mindset behind the architecture (also known as a design rationale). In addition, each module of the architecture must be accompanied with its own design rationale and responsibilities.

7.2.9 1.6.2. Technologies in Use

This segment must provide a technical description of all the technologies that will be used to build the project. These include programming languages, libraries, servers, databases, IDE and compilers, etc. For each technology listed, there must be a short description that explains the rational for the choice and use of the specific tool.

7.2.10 1.7.1. Activities

The activities section is a section that will list a step-by-step process of how the separate artifacts of the project will be produced in order to reach the completion of the project. Each activity must be presented with a clear purpose, description, and must produce at least one artifact in consequence.

7.2.11 1.7.2. Artifacts

An artifact is a by-product that is produced as a consequence of the development of a software project. The artifacts segment will list each document that will be produced in the project. Each artifact listed must come with a description that provides details about the role of that specific document in the completion of the project.

7.2.12 1.7.3. Project Estimates

The project estimates section is for providing a realistic estimate for the cost, as well as the schedule for the length of the project. This is done by summing the evaluated cost of each artifact that's produced throughout the conception of the project. Each cost must be backed up by a basis of the estimation.

7.2.13 1.7.4. Activities Assignments

The activities assignments are the specific assignments of tasks and activities that are made to each team member based on the members' skills, requirements, capacities, limitations and schedules.

7.2.14 1.7.5. Schedule

The schedule section must present all the project's deadlines, milestones, sprints, iterations and other target dates in the form of a diagram or table. A Gantt chart is the ideal way to present a schedule as it outlines the dates all important events from start to finish in a clean and organized fashion.

7.2.15 1.7.6. Risk

The risk section will present all elements of risk of the project, including both early factors and late factors of risk. Each risk must be accompanied by a description and a consequence should the risk occur. They may be presented in a table or list form.

7.2.16 1.8. Prototyping

The prototype section is a description of what work has been completed so far in terms of developing a prototype of the project. A prototype is an early model that is built before the final release of a system or device. The goal of this section is to decide whether the tools and technologies chosen for the completion of the project are appropriate for the final release.

7.2.17 2.2. Introduction to Part 2

The second introduction will introduce the user to the second large deliverable of the project. It is simply a brief description of what information is included in the entire document.

7.2.18 2.3.1. Architecture Diagram

The architecture diagram section is an improved and more detailed version of the architecture diagram presented in the previous deliverable. The system must be divided into at least two systems for this diagram. It must include a high-level description of the architecture of the system in addition to descriptions for each separate module included in the system, accompanied with the visual diagram.

7.2.19 2.3.2. Subsystem Interfaces Specification

The subsystem interfaces specification section is the section that will describe the specifications of the interactions between the software interfaces of the components of the project. These are most likely presented as the function calls and message passing that occur. They must be accompanied with descriptions of the parameters passed of the function calls, including possible valid and invalid ranges of values.

7.2.20 2.4.1. Detailed Design Diagram

The detailed design diagram is a diagram that describes the internal structure of the subsystems depicted in the project. This must be presented as a UML diagram, and must include descriptions that explain the rationale of the designs, in addition to any extra information that is not present in the diagram.

7.2.21 2.4.2. Unit Description

The unit description section is the section that lists each class that's present in the subsystem and accompanies them with a description of the purpose and function of the class. In addition, they are accompanied by all attributes and methods of the class. They may also be notes and reminders that are there to give any extra information for the programmers who are implementing the classes.

7.2.22 2.5. Dynamic Design Scenario

The dynamic design scenario is a list of two vital and substantial use cases of the system. These use cases must have at least 3 system operations, and the scenarios listed must include system sequence diagrams and operational contracts.

7.2.23 2.6. Estimation

The estimation section is a list of all modules that have been identified so far, accompanied by a new estimate for each one. The revision must also include the newly estimated costs of integration, testing, and additional documentation.

7.2.24 2.7. Rapid Prototyping and Risk

The rapid prototyping and risk section lists the prototypes of the system completed so far. The list must be accompanied by all information pertaining to the models, including classes, modules, drivers, frameworks, database, and other technologies used. Each prototype has a description of the effect of its model, in addition to the effects on the risks previously mentioned, the effects on the estimate, and the changes made in the scope because of them.

7.2.25 3.2. Introduction to Part 3

The third introduction will introduce the user to the third large deliverable of the project. It is simply a brief description of what information is included in the entire document.

7.2.26 3.3.1.1. Tested Items

The tested items section is a list of all items and components that have been tested so far, including the test cases that were applied during the process. Each tested item is accompanied with a description that explains the test case along with why the testing was necessary.

7.2.27 3.3.1.2. Untested Items

The untested items section is a list of all items and components that have yet to be tested, but are deemed necessary to be tested. They are accompanied with an explanation that highlights why it should be tested, along with how it would be tested.

7.2.28 3.3.2.1. Unit Testing

The unit testing section includes two mid-level testable units of the system, which could be modules, classes, or subsystems. Each test unit is accompanied by a list of test cases and the code for the drivers used. In addition, an explanation of the techniques and description of the test are provided, along with the results of the testing.

7.2.29 3.3.2.2. Requirements Testing

The requirements testing section is a list of test cases that accompany each tested requirement of the system. This includes a realistic scenario of the system usage during the test, along with the expected reaction of the system under the conditions of the test.

7.2.30 3.3.2.3. Stress Testing

The stress testing section is a description of the potential extreme situations of system usage. This is necessary to view the behavior of the system in unlikely but extreme conditions to ensure stability of the system as a whole. A design of tests is provided along with the description that verifies how the system performs under those extreme conditions.

7.2.31 3.3.2.4. Security Testing

The security testing section is a description of the tests that have been undergone to ensure maximum security of the system. This includes SQL injection attack resistance to avoid breach of user information.

7.2.32 3.4.1. Installation Manual

The installation manual is a report that is aimed for a competent administrator. It is a precise description of the steps to take to install (i.e. from a compressed file or disk) and execute the software. The reader is expected to install the system simply using the code and this manual provided.

7.2.33 3.4.2. User's Manual

The installation manual is a document that is aimed for the general user. It is a precise description of how to use the system, and it provides guidelines for all usable components present. All features of the system must be included and described in this document.

7.2.34 3.5. Final Cost Estimate

The final cost estimate is a table that neatly lists all components that are and have been present in the completion of the project. These estimates include documentation, design, implementation, data acquisition, and testing costs.

7.3 Project Estimates

7.3.1 Deliverable 0 Estimation

Task Names	Cost (Hours)
Domain Model	6
Identification of Technologies	12
Team Members	0.5
Implementation Discussion	5
DMO Descriptions	6
Total	**29.5**

7.3.2 Deliverable 1 Estimation

Task Names	Cost (Hours)
Requirements	10
Use Case Diagrams/Models	8
Resources	5
Scope	5
Architecture	7
Total	**35**

7.3.3 Deliverable 2 Estimation

Task Names	Cost (Hours)
Programming Architecture	100
Use Case Implementation	50
Design Discussion/Implementation	25
Writing up Test Cases	15
Total	**190**

7.3.4 Deliverable 3 Estimation

Task Names	Cost (Hours)
Test the Test Cases	30
Debugging/Optimizing code based on test cases	15
User Performance Test	20
Debugging/Optimizing code based on user tests	10
Total	**75**

7.3.5 Deliverable 4 Estimation

Task Names	Cost (Hours)
Final Optimizations	20
Compiling data into one file	30
Final Test on System	10
Total	**60**

7.3.6 Final Estimations

Deliverable Names	Cost (Hours)
Deliverable 0	29.5
Deliverable 1	35
Deliverable 2	190
Deliverable 3	75
Deliverable 4	60
Total	**389.5**

7.4 Activities Assignments

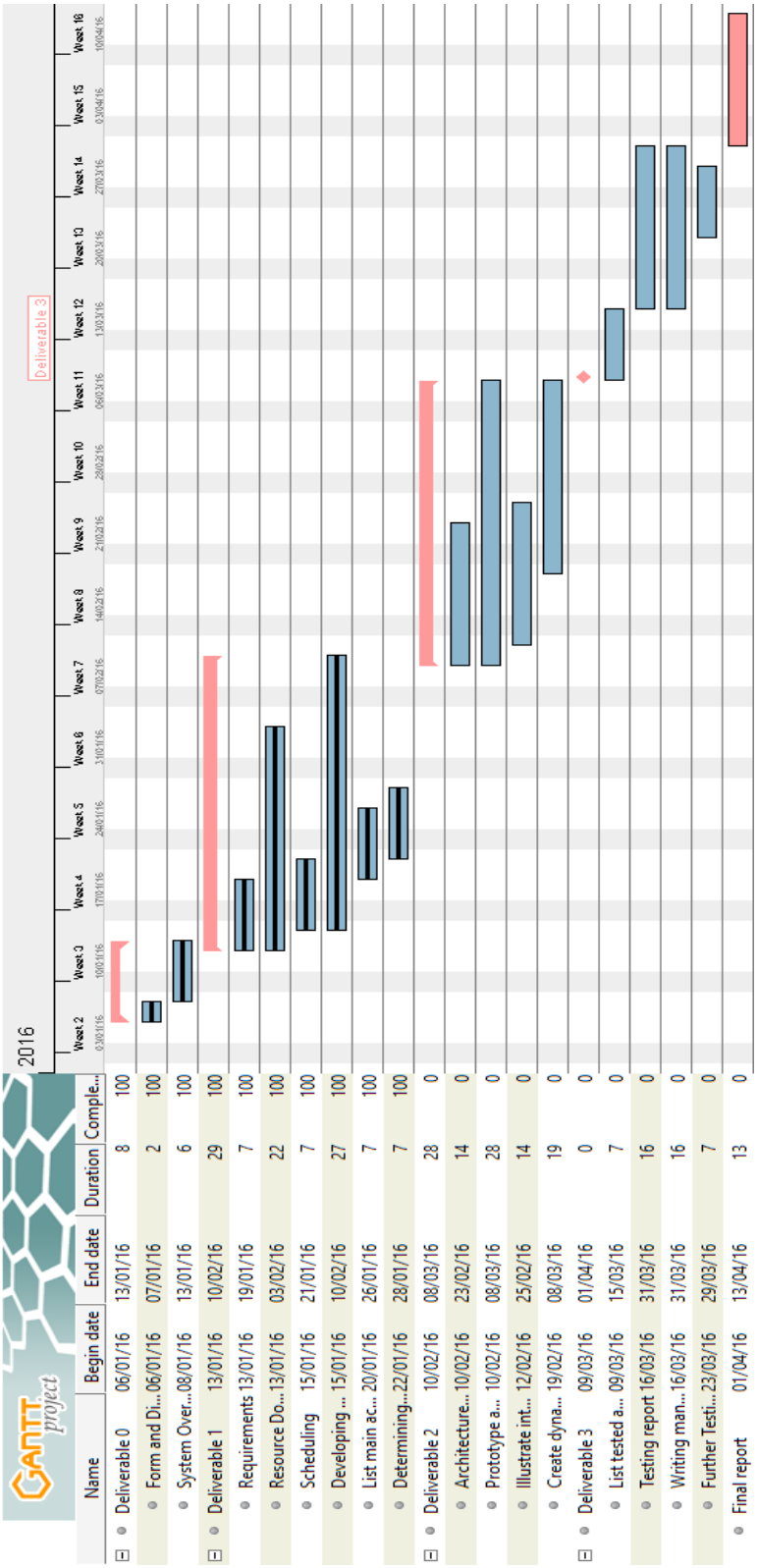
Activity	Assigned member(s)
7.1.1 **Form and divide team**	Claudia Della Serra Dimitri Topaloglou Philip Lim Ryan Lee Daniel Di Corpo Erin Benderoff Aline Koftikian Ideawin-Bunthy Koun Kevin Yasmine Marc-Andre Leclair Lori Dalkin Bryce Drewery-Schoeler
7.1.2 **Complete the system overview**	Claudia Della Serra Dimitri Topaloglou Philip Lim Ryan Lee Daniel Di Corpo Erin Benderoff Aline Koftikian Ideawin-Bunthy Koun Kevin Yasmine Marc-Andre Leclair Lori Dalkin Bryce Drewery-Schoeler

7.1.3 **Write a project description**	Ryan Lee
7.1.4 **Determine the functional requirements and draw use case diagrams accordingly**	Dimitri Topaloglou Bryce Drewery-Schoeler Ideawin-Bunthy Koun Erin Benderoff Lori Dalkin Marc Leclair Claudia Della Serra Daniel Di Corpo
7.1.5 **Review and correct domain model**	Aline Koftikian
7.1.6 **Find and describe constraints and qualities**	Dimitri Topaloglou
7.1.7 **Review and format human resources document**	Lori Dalkin
7.1.8 **Determine technical resources**	Dimitri Topaloglou
7.1.9 **Reduce the amount of features to be realistic according to time and resources**	Kevin Yasmine
7.1.10 **Choose the architecture of the system**	Daniel Di Corpo
7.1.11 **Decide which technologies are relevant to the project development**	Dimitri Topaloglou
7.1.12 **List and describe the main activities involved in the project**	Philip Lim
7.1.13 **List all artifacts**	Aline Koftikian
7.1.14 **Estimate the cost and time of production**	Daniel Di Corpo
7.1.15 **Delegate activities to different team members**	Philip Lim
7.1.16 **Produce a schedule for the project**	Ryan Lee
7.1.17 **Determine the risks of the project**	Dimitri Topaloglou Kevin Yasmine

7.1.18 **Produce a prototype report**	Lori Dalkin
7.1.19 **Ensure professionalism of the document**	Claudia Della Serra
7.1.20 **Update the introduction for Deliverable 2**	Ryan Lee
7.1.21 **Produce an architecture diagram**	Aline Koftikian Dimitri Topaloglou Daniel Di Corpo
7.1.22 **Specify the interactions between the components of the software**	Ideawin-Bunthy Koun Marc-Andre Leclair Philip Lim
7.1.23 **Illustrate the internal structure of the system**	Erin Benderoff Aline Koftikian
7.1.24 **Describe each class in the subsystem**	Kevin Yasmine Lori Dalkin Bryce Drewery-Schoeler
7.1.25 **Produce dynamic design scenarios**	Ryan Lee Claudia Della Serra
7.1.26 **Update project estimates**	Daniel Di Corpo
7.1.27 **Produce a report about the prototype and risk**	Dimitri Topaloglou
7.1.28 **Ensure professionalism of the document**	Claudia Della Serra
7.1.29 **Update introduction**	Ryan Lee
7.1.30 **List all tested items**	Claudia Della Serra
7.1.31 **List all untested items**	Ryan Lee
7.1.32 **Unit testing report**	Lori Dalkin Aline Koftikian
7.1.33 **Requirements testing report**	Erin Benderoff Philip Lim
7.1.34 **Stress testing report**	Ideawin-Bunthy Koun Kevin Yasmine
7.1.35 **Security testing report**	Bryce Drewery-Schoeler Marc-Andre Leclair

7.1.36 **Installation manual redaction**	Claudia Della Serra Dimitri Topaloglou Philip Lim Ryan Lee Aline Koftikian Kevin Yasmine
7.1.37 **User's manual redaction**	Daniel Di Corpo Claudia Della Serra Philip Lim Ryan Lee Aline Koftikian Kevin Yasmine
7.1.38 **Update cost estimate**	Daniel Di Corpo
7.1.39 **Ensure professionalism of the document**	Claudia Della Serra
7.1.40 **Correction of various part of the deliverables**	Claudia Della Serra Dimitri Topaloglou Philip Lim Ryan Lee Daniel Di Corpo Erin Benderoff Aline Koftikian Ideawin-Bunthy Koun Kevin Yasmine Marc-Andre Leclair Lori Dalkin Bryce Drewery-Schoeler
7.1.41 **Assemble all deliverables**	Claudia Della Serra

7.5 Schedule


GANTT project
2016
Deliverable 3
Name
Begin date
End date
Duration
Comple...
Deliverable 0 06/01/16 13/01/16 8 100
Form and Di... 06/01/16 07/01/16 2 100
System Over... 08/01/16 13/01/16 6 100
Deliverable 1 13/01/16 10/02/16 29 100
Requirements 13/01/16 19/01/16 7 100
Resource Do... 13/01/16 03/02/16 22 100
Scheduling 15/01/16 21/01/16 7 100
Developing ... 15/01/16 10/02/16 27 100
List main ac... 20/01/16 26/01/16 7 100
Determining... 22/01/16 28/01/16 7 100
Deliverable 2 10/02/16 08/03/16 28 0
Architecture... 10/02/16 23/02/16 14 0
Prototype a... 10/02/16 08/03/16 28 0
Illustrate int... 12/02/16 25/02/16 14 0
Create dyna... 19/02/16 08/03/16 19 0
Deliverable 3 09/03/16 01/04/16 0 0
List tested a... 09/03/16 15/03/16 7 0
Testing report 16/03/16 31/03/16 16 0
Writing man... 16/03/16 31/03/16 16 0
Further Testi... 23/03/16 29/03/16 7 0
Final report 01/04/16 13/04/16 13 0
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
Week 10
Week 11
Week 12
Week 13
Week 14
Week 15
Week 16

7.6 Risk

7.6.1 Language Familiarities

The main languages used in this project revolve around PHP, SQL, JavaScript and jQuery; not all team members may have much experience using such languages.The language unfamiliarity risks causing further deadline delays due to the learning curves involved in learning a new language. Debugging issues can also arise due to potential inexperience with a certain programming language.

Mitigation of such a risk will involve the use of extensive research on languages, as well as the aid of other team members in learning and passing on knowledge of programming languages.

7.6.2 Server Uptime

This production environment of this project is set up at a team member's own home. This could potentially cause a problem should a power outage occur or the server computer crashing. This can hinder deployment and may ultimately cause certain downtime until the issue is resolved.

7.6.3 Control Version System

The control version system in use is GitHub. However, GitHub requires an extensive learning curve to master and have full control over. This may cause unforeseen delays should there be an issue with the repository.

Mitigation of this risk will involve all team members becoming familiar with GitHub early on in the project. Working with their own repositories, learning how to add, commit, push, branch, and make pull requests on their own time, and learning how issues and milestones work will aid in reducing the potential of the risk occurring.

7.6.4 Time

One of the main concerns with a project this size is time constraint. With the given timeframe, it is uncertain whether there will be adequate time to complete the project in its entirety without the possibility of lacking certain functionality. Essentially, assessing some of the aforementioned risks, time is a common concern.

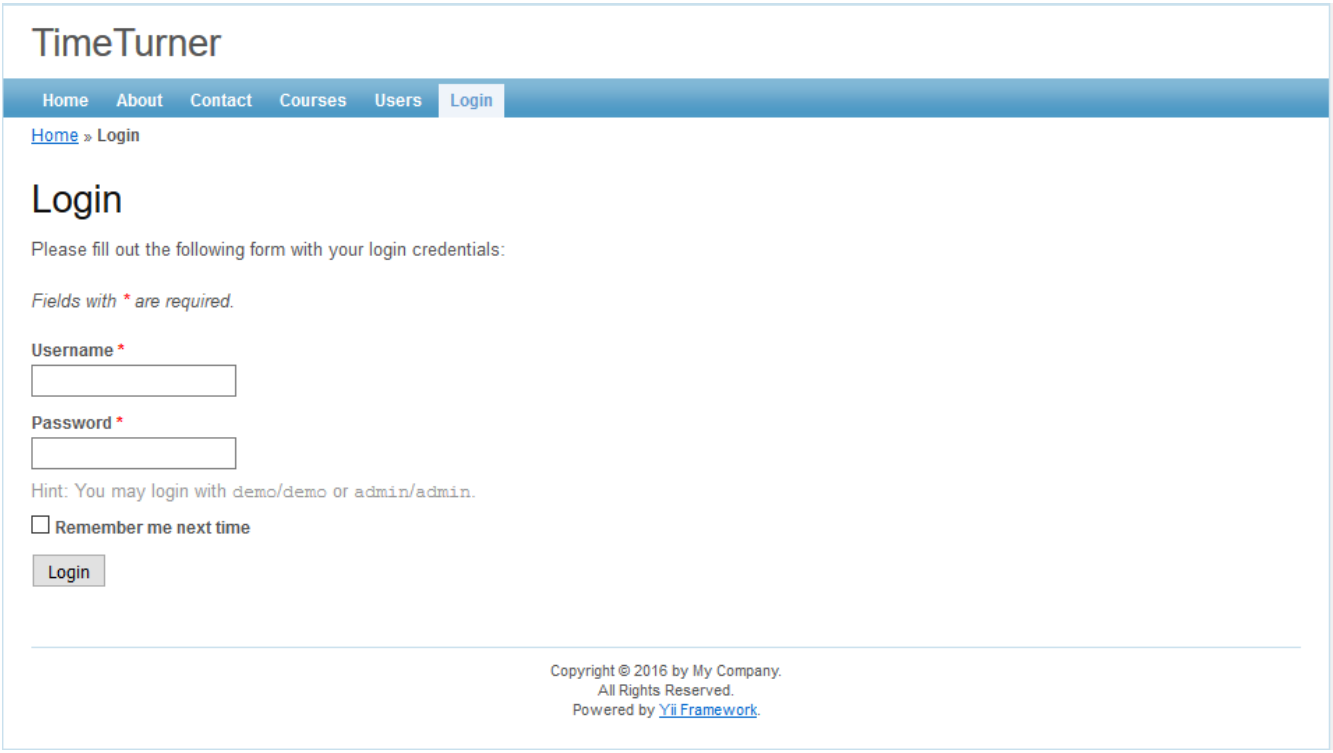
In order to attempt to mitigate such risks, tasks will be divided heavily amongst group members; given that the group consists of 12 members, dividing tasks up will serve to maximize time spent working on the project and time to completion of tasks.

Risk Table				
Risk Summary	Risk Category	Probability	Impact (1-4)	RMMM
Server Uptime	Technology	Medium	1	**7.6.2**
Time	Estimation	High	1	**7.6.4**
Control Version System	People	High	2	**7.6.3**
Language Familiarities	People	Low	3	**7.6.1**

8. Prototyping

The prototype has three functioning pages, the login, the courses and the users page. The log in page allows the user to log in to the website only if their credentials check out. The website sends a request to the database to check if the username and password combination is valid. The current working pair of credentials is: Username= admin, Password= 1234. The update user page allows the user to change their information. This change is transmitted to the database and modifies the user's table. The course page allows the user to view a list of courses from the database. Currently all core class needed for the Software Engineering program are available to see on this list. The user may see information about ten classes at a time and can press the next and previous arrow to move through the list. In this prototype, all users created are admins. In a later iteration, the general user will have more restricted access to the database.

The Yii framework is used to construct this prototype. No members of the group were familiar with this framework thus building the prototype allowed the group members to learn the basics of this framework. This framework follows the MVC architecture format. Yii greatly simplified the processes of making classes to retrieve and modify information from the database by generating the model, view and controller class code in php. This prototype demonstrates that this framework is capable of performing all the specified project requirements.


TimeTurner
Home
About
Contact
Courses
Users
Login
Home » Login
Login
Please fill out the following form with your login credentials:
Fields with * are required.
Username *
Password *
Hint: You may login with demo/demo or admin/admin.
Remember me next time
Login
Copyright © 2016 by My Company.
All Rights Reserved.
Powered by Yii Framework.