

**Concordia University
Department of Computer Science
and Software Engineering**

**Software Process
SOEN 341/4 --- Winter 2009 --- Section S**

SCOPE, REQUIREMENTS & PLAN

Team members information	
Name	SID
Marc-André Moreau	9347100
Mathieu Dumais-Savard	6095275
Julia Lemire	9402969
Phuong-Anh-Vu Lai	5644399
Sébastien Parent-Charette	9178821
Andreas Eminidis	9377603
Corey Clayton	9349200
Eric Chan	9365079

TABLE OF CONTENTS

1. INTRODUCTION	1
2. PROJECT DESCRIPTION.....	2
2.1 System Description.....	2
2.1.1 System Function	2
2.2 Project Goals	2
2.2.1 Objectives.....	2
2.2.2 Deliverables.....	2
3. GOALS AND CONSTRAINTS	4
3.1 Functional Requirements.....	4
3.1.1 Use Cases.....	4
3.1.1.1 Open the Application (UC1)	4
3.1.1.2 Login (UC2)	5
3.1.1.3 Logout (UC3)	5
3.1.1.4 View Schedule (UC4)	5
3.1.1.5 View a Room or Professor Schedule (UC5)	6
3.1.1.6 Browse Course List (UC6)	6
3.1.1.7 View Program Sequence (UC7).....	6
3.1.1.8 View Academic Record (UC8)	7
3.1.1.9 Generate Schedule (UC9)	7
3.1.1.10 View Activity Log (UC10)	8
3.1.1.11 Generate Schedule (advanced) (UC11)	8
3.1.2 Use Case Model	8
3.2 Domain Model.....	10
3.2.1 Domain Model Diagram	10
3.2.2 DLO Login	10
3.2.3 DLO Student	10
3.2.4 DLO Schedule.....	11
3.2.5 DLO Course List.....	11
3.2.6 DLO Course	11
3.2.7 DLO Planner.....	11
3.2.8 DLO Academic Record.....	11

3.3 Quality Standards	11
3.3.1 Speed	12
3.3.2 Ease of Use	12
3.3.3 Scalability	12
3.3.4 Security.....	12
3.3.5 Accessibility	12
3.3.6 Printer Friendly	12
3.4 Platform Requirements	13
3.4.1 Cross Platform Support.....	13
4. SCOPING	14
4.1 Scope.....	14
4.1.1 Core Component	14
4.1.2 Course Planner	14
4.1.3 Academic Record	14
4.1.4 Browse Course List.....	14
4.1.5 Other Features.....	15
4.2 Priority	15
4.3 Removed Features.....	15
4.4 Excluded Features	15
4.4.1 Administrative Functions	15
5. PLAN	16
5.1 Desgin	16
5.1.1 Initial Discussions.....	16
5.1.2 General Design	16
5.1.3 Architecture and Overall Implementation.....	17
5.1.4 In-depth Implementation Design	18
5.1.5 Final Implementation.....	18
5.2 Presentation Tier	18
5.2.1 Implementation of the Tab Modules.....	19
5.2.2 Implementation of the Control Modules.....	19
5.3 Logic Tier	20
5.4 Data Tier.....	21
5.4.1 Construction of the Course Database.....	21

5.4.2 Implementation of the Database	21
5.5 Working Environment.....	22
5.6 Gantt Chart.....	22

LIST OF FIGURES

Figure 2.2.2.1: List of Deliverables	3
Figure 3.1.2.1: Use Case Model	9
Figure 3.2.1.1: Domain Model Diagram	10
Figure 5.6.1: Gantt Chart	23

LIST OF TABLES

Table 5.1.1.1: Plan of Initial Discussions	16
Table 5.1.2.1: Plan of General Design Discussion	17
Table 5.1.3.1: Plan of Architecture Discussion	17
Table 5.1.4.1: Plan of In-Depth Implementation Discussion	18
Table 5.1.5.1: Plan of Final Implementation	18
Table 5.2.1.1: Plan of Implementation of Tab Modules	19
Table 5.2.2.1: Plan of Implementation of Control Modules	20
Table 5.3.1: Plan of Logic Tier	20
Table 5.4.1.1: Plan of Construction of the Course Database	21
Table 5.4.2.1: Plan of Implementation of Main Database	21
Table 5.5.1: Plan of Working Environment	22

1. INTRODUCTION

This document is meant to cover all of the aspects of the project relating to the requirements, both functional and non-functional. It is also a reference to which features are going to be available in the final stage of the project and which features will have been scoped out. This report contains a plan of the estimated time each step will take to complete.

2. PROJECT DESCRIPTION

The aim of this project is to implement a personal web-based scheduling system, designed for undergraduate students enrolled in Software Engineering, based on certain specifications and requirements.

2.1 – System Description

2.1.1 – System Function

This web application will enable registered students to login to their account using a username and a password. Once authenticated, a conflict-free schedule for the upcoming semester will be displayed. Students will have access to an optimized course sequence. It is up to the student whether or not to follow this sequence. Registered students can view their schedules for the current year and modify them by adding constraints or by changing a course or class time. Every modification requested by the student has to meet a list of requirements. For example, if a student wishes to add a course, all of its prerequisites must have been previously completed.

2.2 – Project Goals

2.2.1 - Objectives

The main goal of this project is to design and implement a software package that automates the process of schedule generating. This is to be done using the tools and skills acquired from the SOEN 341 lectures and tutorials that are designed to teach efficient software development. An important aspect of this is to create an attractive graphical user interface, which will enable the user to generate schedules based on specific criteria he or she sets. A key feature of this user interface is its authentication. At each stage, the web application must be secure and all inputs made by the user have to be verified to screen for potential remote exploits. Another aspect of the design and implementation of this project is the design, creation and maintenance of a database. Lastly, the website is to have a specific framework in order to enforce proper coding standards and so as to make the software easily maintainable.

2.2.2 - Deliverables

Throughout the semester, the progress of the project will be demonstrated by delivering five documents. Figure 2.2.2.1 shows a list of the objectives of each document.

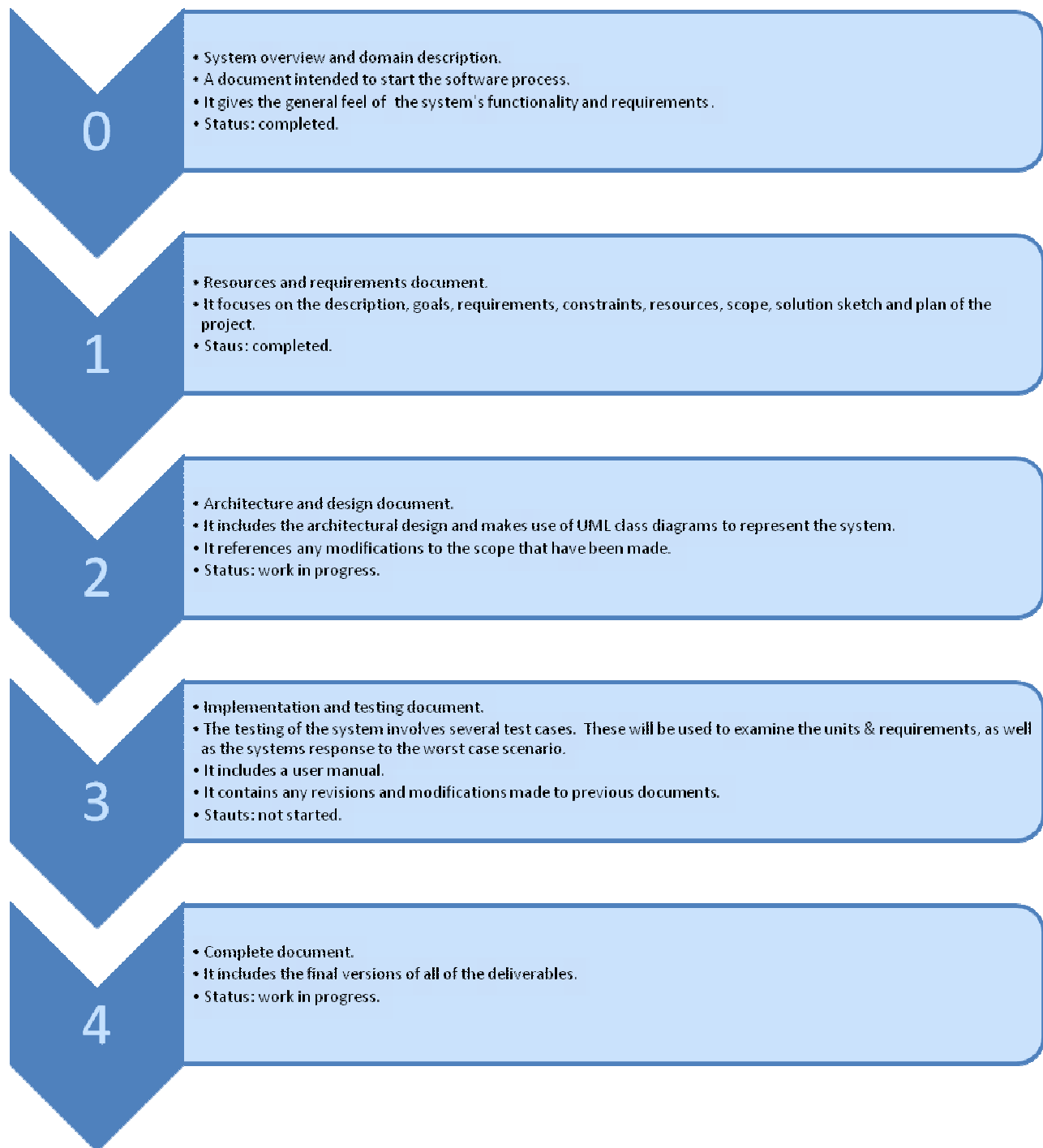


Figure 2.2.2.1 – Overview of the five deliverables that constitute the documentation portion of this project.

Each subsequent document should build and expand upon the previous documents. This includes any modifications that have been made to the previous topics covered. It should be noted that along with the complete report, the final version of the software should be completed.

3. GOALS AND CONSTRAINTS

The objective of this project is to design and implement a personal scheduler for undergraduate Software Engineering students. This section describes the requirements, constraints and quality standards the system is to meet.

3.1 Functional Requirements

3.1.1 – Use Cases

The functional requirements are defined as individual use cases. The following twelve use cases have been outlined for this project. Each one has been assigned an importance and a difficulty level ranging from one to five, with one being considered the lowest importance or difficulty and five being considered the highest importance or difficulty. When a use case has pre-conditions listed, these must be met before the use case can be accessed.

There are two types of actors: a registered user and an unregistered user. A registered user is one with an account they can access with a username and password. This is usually a student. An unregistered user is one who does not have an account. This can be a teacher, a student without an account or someone wishing to view course related information.

3.1.1.1 – Open the Application (UC1)

Importance	5/5	Difficulty	1/5
Description	The trivial step of accessing the system.		
Actors	Any user (public)		
Goal	Actor wants to access the system.		
Pre-conditions			
Step (success path)	1. Actor accesses the entry page of the system.		
Post-condition (success)	The front-end is displayed on the user's screen and the default view is activated.		
Failure end conditions	User uses a non-optimal browser: Warning message is displayed on the screen but application is still accessible to the user. User uses a non-supported browser: Error message is displayed and application is not accessible to the user.		
Related Use Case			

3.1.1.2 – Login (UC2)

Importance	5/5	Difficulty	2/5
Description	Action of signing in into the system.		
Actors	Any user (public)		
Goal	Actor gives credentials to the system to confirm it has access to certain features.		
Pre-conditions	<ul style="list-style-type: none"> ✓ The user has not logged into the system. ✓ The front-end is loaded and a login view is activated. 		
Step (success path)	<ol style="list-style-type: none"> 1. User identifies itself to the system with its credentials. 2. System acknowledges 		
Post-condition (success)	<ul style="list-style-type: none"> ✓ A session is opened on the system for that user 		
Failure end conditions	Wrong Credentials: <ul style="list-style-type: none"> ✗ Message displayed in login window asking for the user to retry. 		
Related Use Case	Open the application (UC1)		

3.1.1.3 – Logout (UC3)

Importance	5/5	Difficulty	1/5
Description	Action of signing out of the system.		
Actors	Registered user		
Goal	Actor terminates the session so that it can no longer be modified.		
Pre-conditions	<ul style="list-style-type: none"> ✓ A session is opened 		
Step (success path)	<ol style="list-style-type: none"> 1. The user activates the sign out feature OR does not perform any action for a significant amount of time. 2. Sign out request is acknowledged by the system. 		
Post-condition	<ul style="list-style-type: none"> ✓ Session is closed on the server. ✓ All front-end data is cleared. 		
Failure End Condition	<ul style="list-style-type: none"> ✗ The system displays an error message stating the error's cause. ✗ The user will not be logged into the system. 		
Related Use Case	Open the application (UC1), Login (UC2)		

3.1.1.4 – View Schedule (UC4)

Importance	3/5	Difficulty	3/5
Description	Default view presented to the user upon login.		
Actors	Registered user		
Goal	Actor wants to consult their schedule.		
Pre-conditions	<ul style="list-style-type: none"> ✓ A session is opened 		
Step (success path)	<ol style="list-style-type: none"> 1. User requests the “View Schedule” feature. 2. The system presents the user's schedule 		
Post-condition			
Failure End Condition	<ul style="list-style-type: none"> ✗ No schedule is available for current actor. 		
Related Use Case	Login (UC2)		

3.1.1.5 – View a Room or Professor Schedule (UC5)

Importance	3/5	Difficulty	3/5
Description	A user accesses the schedule of a given room or professor.		
Actors	Any User (public)		
Goal	A user wishes to consult the schedule of a room or professor.		
Pre-conditions			
Step (success path)	1. User requests the “View Schedule” feature. 2. The system presents the user’s schedule. 3. User requests the schedule for a specific room or professor. 4. The system presents the schedule of the selected entity.		
Post-condition			
Failure End Condition	✕ No schedule is available for the given entity.		
Related Use Case			

3.1.1.6 – Browse Course List (UC6)

Importance	5/5	Difficulty	3/5
Description	Course calendar can be browsed at will.		
Actors	Any user (public)		
Goal	A user who wants more information about a course accesses it in the browse course list section.		
Pre-conditions			
Step (success path)	1. User activates the “Browse Course List” feature. 2. System prompts about which types of courses to display. 3. User indicates department and/or course number and/or professor and asks for the course list. 4. System presents a list of corresponding courses. 5. User selects a course in that list. 6. System displays detailed information about the selected course.		
Post-condition			
Failure End Condition			
Related Use Case			

3.1.1.7 – View Program Sequence (UC7)

Importance	2/5	Difficulty	1/5
Description	View the complete course sequence of the program in which the student is enrolled.		
Actors	Registered user		
Goal	To inform the user about their progression in the program’s sequence.		
Pre-conditions	✓ A session is opened		
Step (success path)	1. The user requests the “View Program Sequence” feature. 2. System reports on the sequence by listing courses and their status.		
Post-condition			
Failure End Condition	✕ The system warns there is no sequence data available for the current student. Maybe the student is not enrolled in a program.		
Related Use Case	Login (UC2)		

3.1.1.8 – View Academic Record (UC8)

Importance	1/5	Difficulty	1/5
Description	To display the current academic information of a student.		
Actors	Registered user		
Goal	Student wants to access their academic information pertaining to either previously taken or current courses.		
Pre-conditions	✓ A session is opened		
Step (success path)	1. The user activates the “View Academic Record” feature. 2. System presents the user’s academic record, stating courses taken with their grade and courses currently registered in.		
Post-condition			
Failure End Condition	✗ No academic record found for the current student.		
Related Use Case	Login (UC2)		

3.1.1.9 – Generate Schedule (UC9)

Importance	5/5	Difficulty	5/5
Description	In order to register for courses, the user needs to generate a schedule. They can do so by selecting courses.		
Actors	Registered user		
Goal	Student desiring to register for courses must generate their schedule.		
Pre-conditions	✓ A session is opened.		
Step (success path)	1. User activates the “Generate Schedule” feature. 2. The system responds with a list of possible schedules. 3. The user adopts one of the proposed schedules. 4. The system prompts for confirmation, specifying any currently registered courses that might be dropped or replaced in the process. 5. User confirms the selection. 6. System confirms the transaction.		
Post-condition	✓ A new scheduled is saved and the selected courses are registered. ✓ Alternate courses (if any) for the present semester are dropped.		
Failure End Condition	✗ System displays an error if no schedule can be generated based on the selected courses.		
Related Use Case	Login (UC2)		

3.1.1.10 – View Activity Log (UC10)

Importance	0/5	Difficulty	2/5
Description	To access a detailed log of a user's account		
Actors	Registered user		
Goal	The user wishes to check details about the occurrence of events.		
Pre-conditions	✓ A session is opened		
Step (success path)	1. User activates the "View Activity Log" feature. 2. The system responds with a list of login/logout transactions and detailed information all events and their origin.		
Post-condition			
Failure End Condition	✗ No log found for the current user.		
Related Use Case	Login (UC2)		

3.1.1.11 – Generate Schedule (advanced) (UC11)

Importance	5/5	Difficulty	5/5
Description	In order to register for courses, the user needs to generate a schedule. They can do so by selecting courses and constraints.		
Actors	Registered user		
Goal	Student desiring to register for courses must generate their schedule with some conditions.		
Pre-conditions	✓ A session is opened.		
Step (success path)	1. User activates the "Generate Schedule" feature. 2. The system responds with a list of possible schedules. 3. User specifies which courses they want to take by rejecting the ones offered by the system and specifying new ones. 4. User specifies time constraints during which they do not want class. 5. System presents a new set of possible schedules. 6. User adopts one of the presented schedules. 7. The system prompt for confirmation, specifying any currently registered courses that might be dropped or replaced in the process. 8. User confirms the selection. 9. System confirms the transaction.		
Post-condition	✓ A new scheduled is saved and selected courses are registered. ✓ Alternate courses (if any) for the present semester are dropped.		
Failure End Condition	✗ System displays an error if no schedule can be generated from the data input by the user. This may be due to time conflicts between selected coursed or too many constraints.		
Related Use Case	Login (UC2)		

3.1.2 – Use Case Model

Figure 3.1.2.1 shows the use case model for this application. As previously mentioned, there are two actors: a registered user and an unregistered user. The registered user has access to all of the use cases. The unregistered user is limited to the public use cases. More specifically these are opening the application (UC1), logging in (UC2), viewing the schedule of a

given room or teacher (UC5) and browsing the course list (UC6). It is important to note that should an unregistered user attempt to login, this should result in a failure end condition since an unregistered user does not have an account.

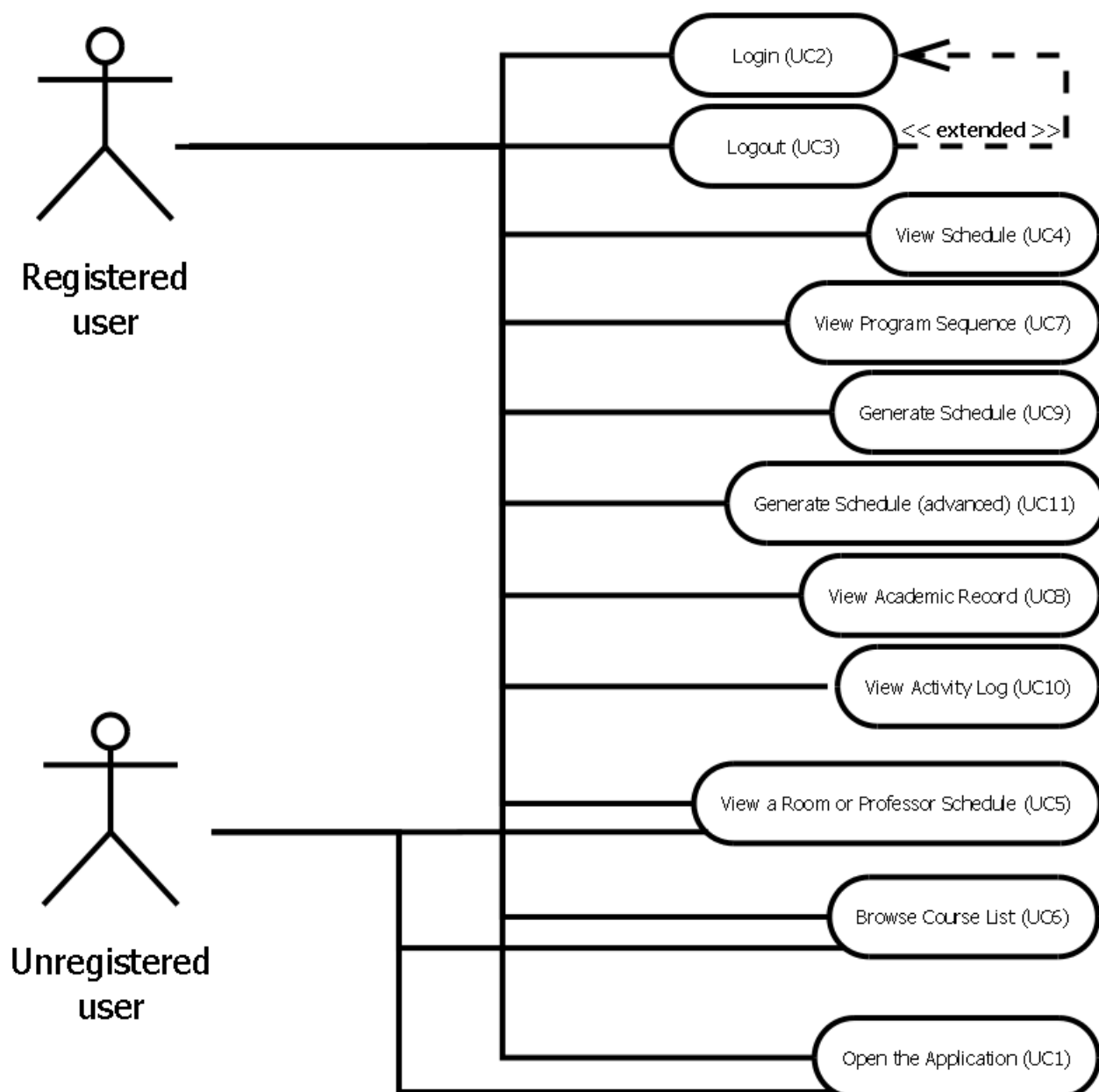


Figure 3.1.2.1 – Use case model depicting the relationship between the actors and the use cases.

3.2 – Domain Model

3.2.1 – Domain Model Diagram

Figure 3.2.1.1 depicts the domain model of the project. There are a total of seven domain level objects (DLO).

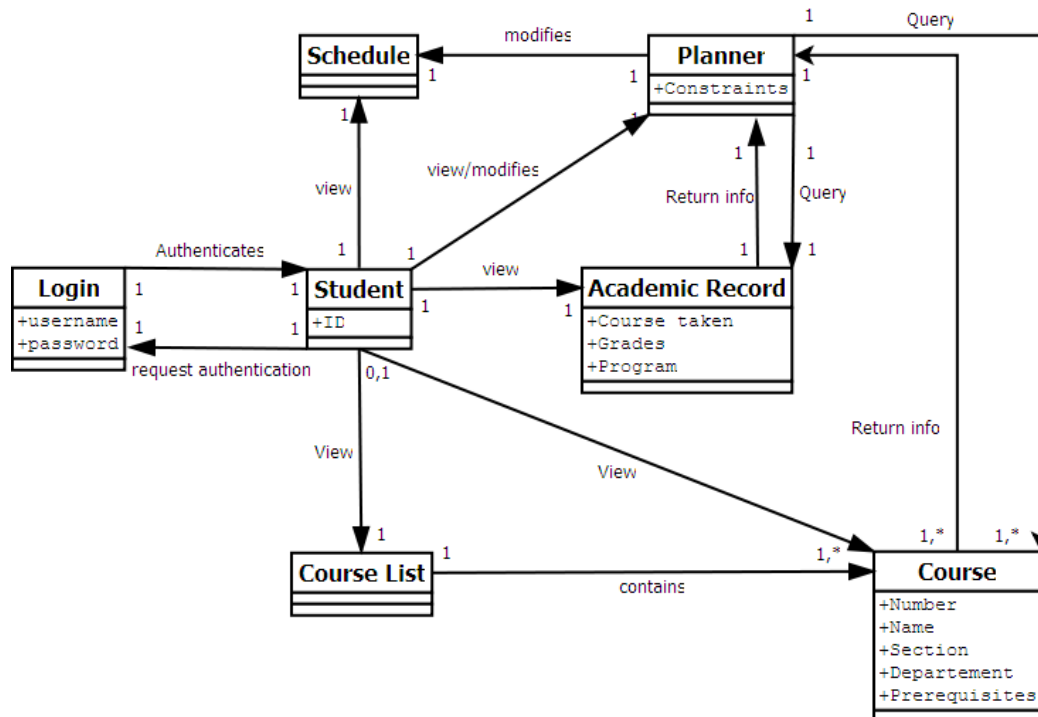


Figure 3.2.1.1 – Domain model diagram.

3.2.2 – DLO Login

Login has two attributes, they being a username and a password. It is connected to the student domain level object. Login authenticates student if its two attributes are valid. This is a one-to-one relationship.

3.2.3 – DLO Student

Student has one attribute – an identification number. It is connected to the login schedule, course list, academic record, course and planner domain level objects. It requests authentication from login. It can view schedule, course list, course and academic record. The student-course list relationship can either be one-to-one or zero-to-one since course list can also be accessed by a user that is not logged in. The student-schedule, student-academic record and student-planner relationships are all one-to-one. The student-course relationship can be

one-to-one, zero-to-one, one-to-many or zero-to-many. Once again, a user can access the course DLO without being logged in. A user may also access more than one course at a time. Student can both view and modify planner. This is a one-to-one relationship.

3.2.4 – DLO Schedule

Schedule has no attributes and does not connect to other domain level objects. This is not to say that other domain level objects do not connect to schedule.

3.2.5 – DLO Course List

Course list has no attributes. It is connected to course by either a one-to-one or a one-to-many relationship. More than one course may be contained by course list at a given time.

3.2.6 - DLO Course

Course can have five attributes – a name, a number, a section, a department and a prerequisite. It is connected to planner by a one-to-one or a many-to-one relationship. Course returns information to planner.

3.2.7 – DLO Planner

Planner can have one attribute – a constraint. It is connected to course, academic record and schedule. Planner queries data from course and academic record. Planner-course can either be a one-to-one or a one-to-many relationship since more than one course may be queried at a given time. Planner-academic record is a one-to-one relationship. Planner can modify the schedule of a given user. This is a one-to-one relationship.

3.2.8 – DLO Academic Record

Academic record can have three attributes – courses taken, grades and the student's program. It is connected to the planner domain level object in that it returns a student's information. This is a one-to-one relationship.

3.3 –Quality Standards

This section covers all of the non-functional requirements of the system, as well as the quality standards the system must meet.

3.3.1 – Speed

Due to the nature of the system, i.e. a web application, it is expected to respond quickly. More specifically, all general requests are expected to take less than two seconds. The schedule generating request is expected to take less than five seconds.

3.3.2 – Ease of Use

The system must be user-friendly. Since this is being made for students, there are a lot of different people with various backgrounds who will use it. Any user to access the domain should be able to easily find their way around without a formal training session. To ensure this, it is important that the most frequently used features of the application are intuitive. This could be tested by having untrained users experiment with the application to see how long it takes to generate a schedule. Also, features common to web-browsers should be integrated into the user interface. An example would be to have a functional back button that is intuitive.

3.3.3 – Scalability

The system should be scalable and should support a minimum of 5000 concurrent users. It should be feasible for this minimum number of users to be connected simultaneously to the system. Should this not be possible, a fallback could be to display a warning message to a user at login when there are a large number of users already using the system.

3.3.4 - Security

The system might be used in a public kiosk environment, for example in a computer lab. In this environment any user can log in using a monitored workstation. The private data should in no way be accessible to anyone else but the authenticated user.

3.3.5 – Accessibility

Since the application is web-based, it should be easily accessible to anyone with an internet connection and a web browser. Users with disabilities like visual impairment should be able to use their browser's accessibility features, like page zooming or text reading, without disrupting the application.

3.3.6 – Printer Friendly

If the user so chooses, they should be able to print any pertinent information from the application using the native print feature of their browser. More specifically, the layout and other artifacts should be neglected and only the specific content of the page should be printed.

3.4 – Platform Requirement

3.4.1 – Cross Platform Support

The basic requirement for the system is a web browser that can function on the client side. Any HTML 4.01 compliant browser with JavaScript support should be able to operate it. This means the system should work on any device with an operating system, including a phone as long as the web browser it uses is HTML 4.01 and JavaScript compliant.

4. SCOPING

4.1 - Scope

This project aims to create a web application in which a Software Engineering student can plan his or her schedule for a given semester. The system has a core unit which must be completed, as well as extra features that would make the application more flexible.

4.1.1 – Core Component

The core component of the application includes the user login, the basic schedule generation and the schedule display. The user must login using a valid username and password. Upon successful authentication, the default display of the current schedule is shown. If there is no schedule available, one will be generated on the spot.

4.1.2 – Course Planner

Along with the core component, other features hope to be included in the application. One of these includes extra features to be added to the component that generates the schedule. This falls under the Course Planner module. It is through this feature that user will be able to plan their schedule. This is done by selecting courses that will be displayed in the schedule. Constraints such as unavailable time slots or other time preferences can be added by the user to help optimize the schedule to the student's specific needs. The planner will also be able to suggest courses to be taken according to the course sequence and academic record information.

4.1.3 – Academic Record

Students will have access to their academic record. This will include all courses that have been previously taken, along with their corresponding grade as well as any other academia related information. This data is used by other features in the application.

4.1.4 – Browse Course List

Students will have access to a complete course list. This list includes the different sections for a course, the time slots for the classes, their prerequisites, their room location, the teacher assigned to the class and a description of the course. It will also display the course sequence of the student's program. It is to differentiate the courses that have already been completed from the ones yet to be taken.

4.1.5 – Other Features

An activity log which keeps track of any and all transactions made by the student is to be included as a feature in the application. The ability to save a generated schedule, to view the schedule of a specific room or teacher, and to register for courses directly from a generated schedule are also featured to be included in this application.

4.2 - Priority

The core component of the application has the highest priority in terms of implementation. The course planner and the academic record modules are second in terms of priority since many modules are dependent on them. The course list browser is third in terms of implementation since it is mostly outputting data from the database to the user interface. Other features such as the activity log, the save and registration capabilities and the room or teacher schedules have the lowest priority implementation wise. When looking to scope out features, the ones with the lowest priority are likely to go first.

4.3. – Removed Features

No features have been scoped out of the project to-date.

4.4 – Excluded Features

4.4.1 – Administrative Functions

Administrative capabilities for the application and the database management, such as enabling or disabling tabs and modifying user permissions, are currently not part of the scope of this project and therefore, will not be implemented. This is mainly due to the fact that these features are currently not needed in order to achieve the goal of this project. Also, according to the planning of the implementation of this application, there is simply not enough time nor resources available to complete the project with this added functionality.

5. PLAN

This section is to act as a guide for the implementation of the project. It contains a detailed schedule of all of the activities and artefacts to be completed as well as the estimated amount of time each one will take to complete. The activities and artefacts of the project have been divided up into five categories: design, implementation, logic, database and other. For the sake of convenience, each module is represented by a table consisting of all of the artefacts and activities it encompasses and their estimated cost in terms of hours.

5.1 - Design

5.1.1 – Initial Discussions

The initial discussions of the project encompass everything from what features the application will have to what the user interface will look like. It is very likely that the end result of the project will be drastically different from the initial plans. Table 5.1.1.1 details this.

Table 5.1.1.1 – Details of the plan of the initial discussions of the project.

Activity	Initial project discussions : Domain, functionalities, Interface
Assigned To	Corey, Mathieu (Matt), Phuong-Anh-Vu (JB), Sébastien (Seb), Andreas, Eric, Julia, Marc-André (Marc)
Due date	Week 2, Friday
❖ Establishing the scope of the project and the various features.	
Artefact	System Overview Document
Costs	20
➤ Representation of the overall design of the system	
Artefact	Block Diagram
Costs	10
➤ Basic representation of the modules.	
Artefact	User interface mock-up draft
Costs	10
➤ Basic representation of the various parts of the interface and how they are linked.	

5.1.2 – General Design

The general design for the project includes all of the requirements and scope of the application, the plan, the UML diagram and the database diagram. Table 5.1.2.1 details this.

Table 5.1.2.1 – Details of the plan of the general design of the project.

Activity	General design discussion
Assigned To	Corey, Matt, JB, Seb, Andreas, Eric, Julia, Marc
Due date	Week 5, Friday
❖	Initiate discussion between team members on the various aspect of the design and establish the software process and various communication protocols to be used by the team.
Artefact	Requirement, scope and plan document
Costs	30
➤	Documentation of the requirements, the scope and the plan of the project.
Artefact	UML Diagram
Costs	10
➤	Documentation of the software process.
Artefact	Use Case Diagram
Costs	10
➤	Documentation about the actors, their possible actions and results.
Artefact	Project Schedule
Costs	5
➤	Establishes the various deadlines of each part of the process to be completed / delivered.
Artefact	Database diagram
Costs	10
➤	Documentation about the database and its structure and relationships.

5.1.3 – Architecture and Overall Implementation

The architecture and overall implementation discussions involved deciding the structure of the application so that it would be straightforward to code and easy to maintain. Table 5.1.3.1 details this.

Table 5.1.3.1 – Details of the plan for the architecture and overall implementation decisions.

Activity	Architecture / implementation discussion
Assigned To	Corey, Matt, JB, Seb, Andreas, Eric, Julia, Marc
Due date	Week 9, Friday
❖	Establish the various parts of the general architecture and implementation.
Artefact	Architecture and design document
Costs	30
➤	Documentation of the architecture of the software and it's actual implementation.
Artefact	In depth UML diagram
Costs	15
➤	In-depth documentation representing an abstraction of the software processes.
Artefact	In depth database diagram
Costs	15
➤	Documentation of the database and its various components (tables, relationships, etc).

5.1.4 – In-Depth Implementation Design

The in-depth design of the implementation includes the documentation of all of the decisions made and executed as well as the test cases designed from the use cases of the project. Table 5.1.4.1 details this.

Table 5.1.4.1 – Details the plan for the in-depth look at the implementation of the project.

Activity	In depth implementation discussions
Assigned To	Corey, Matt, JB, Seb, Andreas, Eric, Julia, Marc
Due date	Week 12, Friday
❖	General discussion between team members outlining the various aspects of the implementation and establishing the actual implementation methods to be used.
Artefact	Implementation documents
Costs	60
➤	Establishes the methods used to implement the various modules of the software.
Artefact	Testing specifications
Costs	40
➤	Designing testing methods for each module based on the use cases and other requirements.

5.1.5 – Final Implementation

The final result of the project is to include all of the documentation describing the process of designing, implementing and testing the application as well as a functional application. Table 5.1.5.1 details the plan for this.

Table 5.1.5.1 – Details the plan for the final implantation of the complete project.

Activity	Project completion and final implementations
Assigned To	Corey, Matt, JB, Seb, Andreas, Eric, Julia, Marc
Due date	Week 13, Friday
❖	The completion of the entire project (documents, implementation, testing, etc).
Artefact	Complete Report
Costs	80
➤	The final report composed of all documents related to design, architecture, implementation and testing documenting the entire software process for this project from beginning to end.

5.2 – Presentation Tier

This section looks at the implementation of the presentation tier of the project in greater detail. It breaks down the design of the project into different modules based on the architecture of the application. For an in-depth description of the architecture outlined for this system, look at the System Design document.

5.2.1 – Implementation of the Tab Modules

The user interface of the application is to be broken down into tabs, with each one serving a specific purpose. Table 5.2.1.1 looks at the plan for the implementation of these tabs.

Table 5.2.1.1 – Details of the plan for the implementation of the tab modules of the UI.

Activity	Coding and testing tab modules
Assigned To	UI Team (consisting of Matt & Julia)
Due date	Week 12, Friday
❖ The programming required to make a tab styled navigation interface using JavaScript and HTML. This is followed by thorough testing from a user's point of view of each module.	
Artefact	Academic record module
Costs	10
➤ Tab used to access the page presenting a student's record. It is visible from all other modules.	
Artefact	Sequence planner
Costs	30
➤ Tab used to access the page presenting the sequence planner for courses. It is visible from all other modules.	
Artefact	Schedule view module
Costs	10
➤ Tab used to access the page containing the course schedule for a student. It is visible from all other modules.	
Artefact	Core Module
Costs	50
➤ The default module that provides the grants the user access to key components of the application upon successful login. It is also the basis for the other modules in the presentation tier.	
Artefact	Course Browser
Costs	25
➤ Tab used to access the page containing the list of courses offered by the university. It is visible from all other modules. It is also a public resource.	

5.2.2 – Implementation of the Control Modules

The user interface will frequently include specific controls in its implementation. In order to reduce repetition of code, these controls have been separated into modules. Table 5.2.2.1 shows the plan for the implementation of these control modules.

Table 5.2.2.1 – Details the plan for the implementation of the control modules of the UI.

Activity	Coding and testing of control modules
Assigned To	UI Team (consisting of Matt & Julia)
Due date	Week 12, Friday
❖ The actual implementation – the code – of the logic required to generate the objects and data used in the presentation layer as well as the testing of this implementation.	
Artefact	Model window module
Costs	25
➤ The module to construct objects needed for UI windows.	
Artefact	Filter module
Costs	25
➤ The module for parsing courses based on the users specifications.	
Artefact	Calendar Module
Costs	25
➤ The module responsible for creating the objects to be rendered in a calendar. These objects will consist mostly of courses. Filters may also be visualized.	

5.3 – Logic Tier

This section looks at the plan for the different modules within the logic tier of the application's architecture. This can be seen in Table 5.3.1.

Table 5.3.1 – Details of the plan for the logic tier of the application's architecture.

Activity	Coding and testing of Logical Modules
Assigned To	Corey, Marc, JB, Seb, Andreas, Eric
Due date	Week 12, Friday
❖ All the underlying implementation of the logic required to present a schedule to a user based on selected courses and temporary filters.	
Artefact	Academic record module
Costs	30
➤ The module responsible for the generation of the completed courses and grades of student.	
Artefact	Authentication
Costs	30
➤ The module handling the secure login of users, and ensuring the security of transactions.	
Artefact	Course module
Costs	50
➤ The module dealing with the objects representing courses.	
Artefact	Registration
Costs	25
➤ Validates necessary requirements and registers the student.	
Artefact	Scheduling module
Costs	60
➤ This module handles the generation of data structures that represent valid schedules.	

5.4 – Data Tier

This section examines the plan for the data used by the application. All of the data is to be stored in a database created by team7.

5.4.1 – Construction of the Course Database

The database containing all of the data pertaining to courses the student can select when generating their schedule must be created by team7. The plan for this can be seen in Table 5.4.1.1.

Table 5.4.1.1 – Details the plan for the construction of the course database.

Activity	Course database construction
Assigned To	Marc, Corey
Due date	Week 11, Friday
❖ Make a dataset either by mining the registrar’s website or by dumping previously compiled databases.	
Artefact	Course Database Entries
Costs	15
➤ Uses the obtained course data to populate rows in the relational database.	

5.4.2 – Implementation of the Database

The main database of the system contains the data relating to the academic history of a student, user accounts, course lists, generated schedules, etc. The plan for this is shown in Table 5.4.2.1.

Table 5.4.2.1 – Details of the plan for the design and implementation of the main database of the system.

Activity	Coding and testing of the database
Assigned To	Seb, Marc
Due date	Week 11, Friday
❖ The design and implementation of the relational database to facilitate the operation of the entire system.	
Artefact	Main MySQL database
Costs	30
➤ The database for storing all of the states involved in the system. This includes academic records, user accounts, course lists, etc.	

5.5 – Working Environment

This section includes all other work related to the project that is not directly linked to the design and implementation of the application. This includes taking care of the web server, the git repository and the mailing list. Table 5.5.1 shows the plan for these tasks.

Table 5.5.1 – Details the plan for setting up and maintaining the working environment.

Activity	Setting up the server and working environment
Assigned To	Marc
Due date	Week 3, Friday
❖	Deploying and maintaining the back-end services that act as a platform for the project.
Artefact	Web Server
Costs	15
➤	Provides hyper-text document to web clients. Platform on which the system will run.
Artefact	Git repository
Costs	5
➤	The distributed version control system used for team members to share resources and track modifications to documents and code.
Artefact	Mailing List
Costs	5
➤	For keeping team members in regular communication with each other.

5.6 – Gantt Chart

Figure 5.6.1 is a Gantt chart depicting the expected progress of the project. It contains all of the components outlined in the plan. Each component includes a start date, an end date, the expected duration time, a percentage representing what has been completed to-date and a visual representation of the expected time the component will take to complete. The blue bar graph indicates the time a component is expected to take before it is completed. The pink bar graph indicates the actual amount of time spent working on a component.

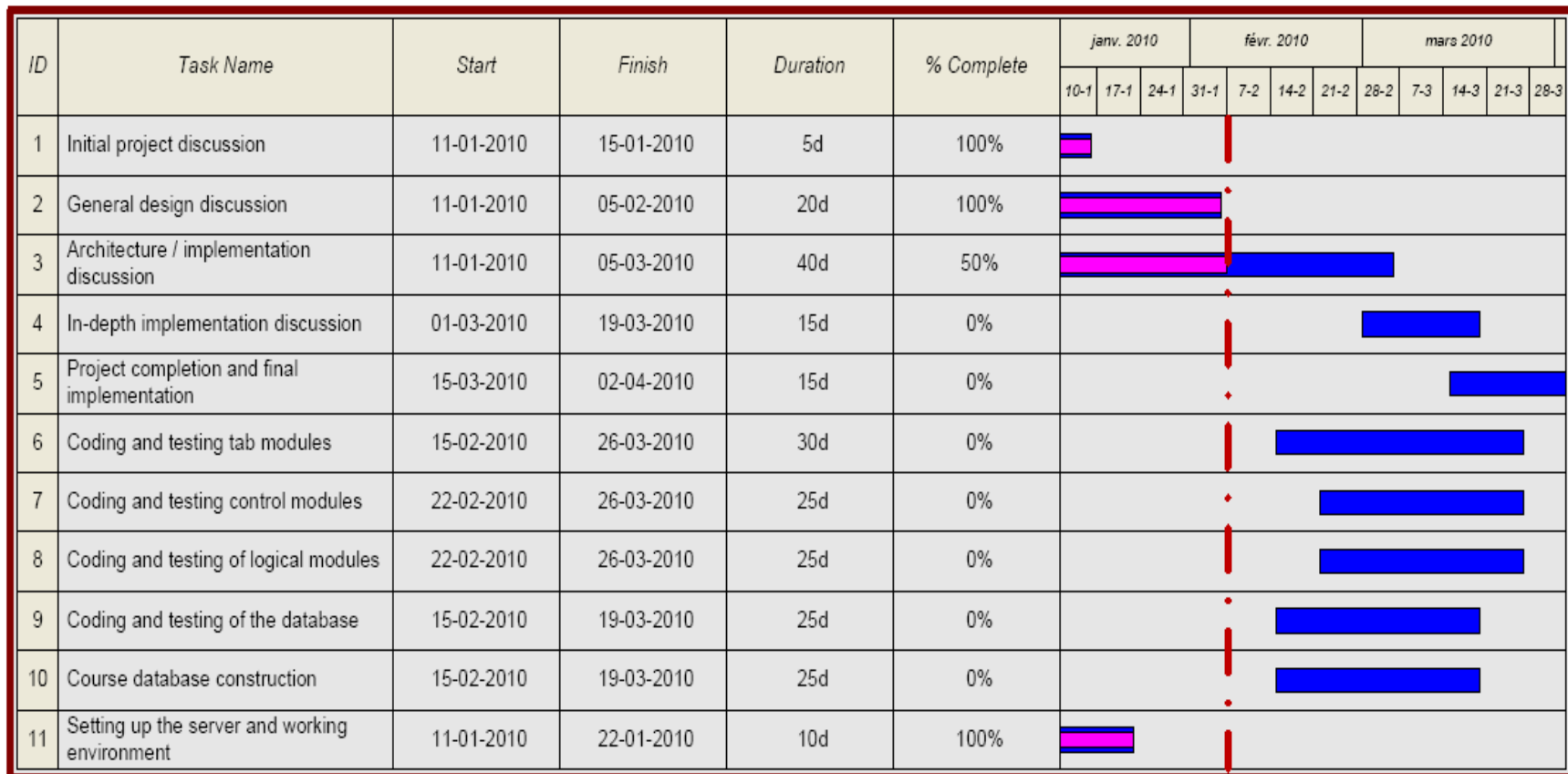


Figure 5.6.1 – Gantt chart representing the estimated progress of work for the project.