Concordia University Department of Computer Science and Software Engineering

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

PLANZILLA DEVELOPEMENT PROCESS DOCUMENT

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	INTRO	DUCTION	1
2	PROJE	CT DESCRIPTION	2
	2.1 S	ystem Description	2
	2.1.1	System Function	2
	2.2 P	roject Goals	2
	2.2.1	Objectives	2
	2.2.2	Deliverables	2
3	GOAL	S AND CONSTRAINTS	4
	3.1 F	unctional Requirements	4
	3.1.1	Use Cases	4
	3.1.2	Use Case Model	8
	3.2	omain 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 C	Quality Standards	11
	3.3.1	Speed	11
	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
	3.3.7	Workflow Resilience in Regards to Network Issues	12
	3.4 P	latform Requirement	13
	3.4.1	Cross Platform Support	13
4	SCOPI	NG	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	l	16
	5.1	Design	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	p	
	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	21
		Gantt Chart	
6	SERV	ER SETUP	24
	6.1	Assembla.com Team Collaboration Space	24
	6.1.1	Team Collaboration	24
	6.1.2	•	
	6.1.3	Webhook	24
	6.2	OpenSolaris Server	25
	6.2.1	Web Stack (AMP)	25

	6.2.2	Development Web Stack (AMP-dev)	25
	6.2.3	Hostname	25
	6.2.4	Individual Test Space	25
7	DEVEL	OPER CLIENT SIDE SETUP	26
	7.1	it	26
	7.1.1	Installation (Windows)	26
	7.1.2	Installation (Linux / Mac OS X)	26
	7.1.3	Initial Configuration	26
	7.1.4	Contributing Changes	28
	7.1.5	Synchronizing Work	29
	7.2 N	letBeans IDE	29
	7.2.1	Installation	29
	7.2.2	Plugins	29
	7.2.3	Initial Configuration	29
	7.2.4	Remote PHP Debugger	30
8	PROG	RAMMING LANGUAGES AND LIBRARIES	31
	8.1 L	anguages	31
	8.1.1	PHP	31
	8.1.2	HTML	31
	8.1.3	CSS	31
	8.1.4	Javascript (a dialect of ECMAScript)	31
	8.2 L	ibraries	32
	8.2.1	jQuery	32
	8.2.2	jQueryUI	32
	8.3 P	Plugins	32
	8.3.1	jQuery-Week-Calendar	32
9	OTHE	R SOFTWARE	33
	9.1 C	Operating Systems	33
	9.1.1	Microsoft Windows	33
	9.1.2	Apple Mac OS X	33
	9.1.3	GNU/Linux	33
	9.2	Office Software	33
	9.2.1	Microsoft Office	33

	9.2.2	2 Di	a	33
	9.2.3	В М	lySQL	34
10	HUN	1AN F	RESOURCES	35
1	0.1	Eric	Chan	35
1	0.2	Séba	astien Parent-Charrette	35
1	0.3	Core	y Clayton	35
1	0.4	Mat	hieu Dumais-Savard	36
1	0.5	Andı	reas Eminidis	36
1	0.6	Phu	ong-Anh-Vu Lai	36
1	0.7	Julia	Lemire	36
1	8.0.	Mar	c-André Moreau	37
11	3-TII	R AR	CHITECTURE	38
1	1.1	Ove	rview	38
12	DEV	ELOP	MENT VIEW	39
1	2.1	Com	ponent Diagram	39
1	2.2	Brief	f Component Diagram Description	40
	12.2	.1	User	40
	12.2	.2	Tabs	40
	12.2	.3	Controls	40
13	PRES	SENT	ATION TIER	41
1	3.1	Tabs	S	41
	13.1	.1	Academic Record	41
	13.1	.2	Sequence Planner	41
	13.1	.3	Schedule Viewer	41
	13.1	.4	Course List Browser	41
1	3.2	Cont	trols	41
	13.2	.1	Core	41
	13.2	.2	Modal Window	42
	13.2	.3	Filter	42
	13.2	.4	Calendar	43
14	LOG	IC TIE	ER	44
1	4.1	Auth	nentication	44
1	4.2	Acad	demic Record	44

	14.3	Regi	stration	.44
	14.4	Sche	eduling	.44
	14.5	Cou	rse	.44
15	DAT	A TIE	R	.45
	15.1	Ove	rview	.45
	15.2	User	~S	.45
	15.2	.1	Administrator	.46
	15.2	.2	Student	.46
	15.2	.3	Teacher	.46
	15.3	Cou	rses	.47
	15.3	.1	Course	.47
	15.3	.2	Class	.47
	15.3	.3	Department	.48
	15.3	.4	Faculty	.48
	15.4	Requ	uirements	.48
	15.4	.1	"AND" Requirements	.48
	15.4	.2	"OR" Requirements	.48
	15.4	.3	Combination of "AND" and "OR" Requirements	.49
	15.5	Prog	rams	.49
	15.5	.1	Program Options	.49
	15.5	.2	Course Sequence	.49
	15.6	Fina	l Database Model	.50
16	PRO	CESS	VIEW	.51
	16.1	Activ	vity Diagram	.51
17	PHY:	SICAL	. VIEW	.53
	17.1	Depl	loyment Diagram	.53
	17.1	.1	OpenSolaris Server	.53
	17.1	.2	User Client	.54
18	LOG	ICAL	VIEW	.55
	18.1	Class	s Diagram	.55
	18.1	.1	Unit Descriptions	.55
	18.1	.2	ScheduleViewer	.55
	18.1	.3	CourseBroswer	.56

	18.1.4	SequencePlanner	56
	18.1.5	AcademicRecord	56
	18.1.6	Main	56
	18.1.7	Course	56
	18.1.8	Authentication	56
	18.1.9	StudentRecord	56
	18.1.10	Database	57
19	MODULE	INTERFACE SPECIFICATION	58
1	9.1 Auth	nentication Class	58
	19.1.1	Detailed Description	58
	19.1.2	Public Member Functions	58
	19.1.3	Public Attributes	58
	19.1.4	Member Function Documentation	58
	19.1.5	Member Data Documentation	60
	19.1.6	Source File	60
1	9.2 Cou	rse Class	60
	19.2.1	Detailed Description	60
	19.2.2	Public Member Functions	60
	19.2.3	Member Function Documentation	61
	19.2.4	Source File	62
1	9.3 Data	abase Class	62
	19.3.1	Detailed Description	62
	19.3.2	Public Member Functions	62
	19.3.3	Member Function Documentation	62
	19.3.4	Source File	63
1	9.4 Stud	lentRecord Class	63
	19.4.1	Detailed Description	63
	19.4.2	Public Member Functions	63
	19.4.3	Private Attributes	63
	19.4.4	Member Function Documentation	64
	19.4.5	Member Data Documentation	64
	19.4.6	Source File	65
20	DYNAMIO	C DESIGN SCENARIOS	66

2	0.1	Browse Course List	66
	20.1.	System Sequence Diagram 1 (SSD1) .	66
	20.1.	Operational Contract 1.1 (CO1.1)	66
2	0.2	Generate Schedule (Advanced)	67
	20.2.	System Sequence Diagram 2 (SSD2) .	67
	20.2.	Operational Contract 2.1 (CO2.1)	68
	20.2.	Operational Contract 2.2 (CO2.2)	68
	20.2.	Operational Contract 2.4 (CO2.4)	68
	20.2.	Operational Contract 2.5 (CO2.5)	69
	20.2.	Operational Contract 2.6 (CO2.6)	69
	20.2.	Operational Contract 2.7 (CO2.7)	69
	20.2.	3 Operational Contract 2.8 (CO2.8) (Sc	oped out)69
	20.2.	Operational Contract 2.9 (CO2.9) (Sc	oped out)70
2	0.3	Miscellaneous	70
	20.3.	Operational Contract 3.1 (CO3.1)	70
	20.3.	Operational Contract 3.2 (CO3.2)	70
	20.3.	Operational Contract 3.3 (CO3.3)	70
	20.3.	Operational Contract 3.4 (CO3.4)	71
	20.3.	Operational Contract 3.5 (CO3.5)	71
	20.3.	Operational Contract 3.6 (CO3.6)	71
	20.3.	7 Operational Contract 3.7 (CO3.7)	71
	20.3.	Operational Contract 3.8 (CO3.8)	71
	20.3.	Operational Contract 3.9 (CO3.9)	72
21	REVIS	ION HISTORY	73
22	TESTI	NG REPORT	74
2	2.1	Test coverage	74
	22.1.	L Tested Items	74
	22.1.	2 Untested Items of Interest	75
2	2.2	Fest Cases	76
	22.2.	L Unit Testing	76
	22.2.	Requirement Testing	80
	22.2.	3 Stress testing	85
23	SYSTE	M DELIVERY	86

23.1 Inst	all Manual	86
23.1.1	Installation Instructions	86
23.1.2	Server Verification & Installation	86
23.1.3	PlanZilla Installation	86
23.1.4	PlanZilla Configuration	86
23.1.5	Database Preparation	87
23.1.6	Database Population	87
23.1.7	Deployment	87
23.1.8	Software Downloads	87
23.2 Inst	ruction Manual	88
23.2.1	Accessing Planzilla	88
23.2.2	Login/Logout	88
23.2.3	Navigation Bar	89
23.2.4	View Schedule	89
23.2.5	Browse Course	90
23.2.6	Program Requirement	90
23.2.7	Sequence Planner	91
23.2.8	Academic Record	92
REFERENCES.		93
APPENDIX A -	- Latest Requirement Test Report	94
APPENDIX B -	- Latest Unit Testing Reports	99

LIST OF FIGURES

Figure 2.1 - Overview of the project's five deliverables	3
Figure 3.1 - Use Case Model	9
Figure 3.2 - Domain Model Diagram	10
Figure 5.1 - Gantt chart	
Figure 7.1 - Example of proper git clone	26
Figure 7.2 - Example of a list of git configurations	27
Figure 7.3 - Instructions for configuring the user.name and the user.email settings	27
Figure 7.4 - Instruction for calling the configured settings and a list of the current ones	27
Figure 7.5 - Example of a git status summary	28
Figure 7.6 - Instruction for a git commit including a comment	28
Figure 7.7 - Instruction for pushing changes into the git repository	28
Figure 7.8 - Command for opening an ssh tunnel in a linux terminal	30
Figure 11.1 - 3-tier architecture	38
Figure 12.1 - UML Component Diagram	39
Figure 13.1 - Example of a modal window	42
Figure 13.2 - Example of the calendar	43
Figure 15.1 - Entity-Relationship Diagram	
Figure 15.2 - User credentials and information.	46
Figure 15.3 - Courses in the database	47
Figure 15.4 - Representation of requirements	48
Figure 15.5 - Representation of a program	
Figure 15.6 - Reverse engineered entity-relationship model for the PlanZilla database	50
Figure 16.1 - Activity diagram	
Figure 17.1- Deployment diagram.	53
Figure 18.1 - Class diagram	55
Figure 20.1 - Browse Course List system sequence diagram (SSD1)	
Figure 20.2 - Generate Schedule (Advanced) system sequence diagram (SSD2)	
Figure 23.1 - Login Screen	88
Figure 23.2 - PlanZilla navigation bar	
Figure 23.3 - View schedule screenshot	
Figure 23.4 - Browse course screenshot	
Figure 23.5 - Program Requirement screenshot	
Figure 23.6- Sequence Planner Screenshot	91
Figure 23.7- Academic Record Screenshot	92

LIST OF TABLES

Table 5.1 - Plan of the initial discussions of the project	16
Table 5.2 - Plan of the general design of the project	17
Table 5.3 - Plan for the architecture and overall implementation decisions	17
Table 5.4 - Plan for the in-depth look at the implementation of the project	18
Table 5.5 - Plan for the final implantation of the complete project	18
Table 5.6 - Plan for the implementation of the tab modules of the UI	19
Table 5.7 - Plan for the implementation of the control modules of the UI	20
Table 5.8 - Plan for the logic tier of the application	20
Table 5.9 - Plan for the construction of the course database	21
Table 5.10 - Plan for the design and implementation of the main database of the system	21
Table 5.11 - Plan for setting up and maintaining the working environment	22
Table 21.1 - Revision History	73

LIST OF USE CASES

Use Case 1	Open the Application (UC1)	4
	Login (UC2)	
	Logout (UC3)	
Use Case 4	View Schedule (UC4)	5
Use Case 5	View a Room or Professor Schedule (UC5) (Scoped out)	6
Use Case 6	Browse Course List (UC6)	6
Use Case 7	View Program Sequence (UC7)	6
Use Case 8	View Academic Record (UC8)	7
Use Case 9	Generate Schedule (UC9)	7
Use Case 10	View Activity Log (UC10) (Scoped out)	7
Use Case 11	Generate Schedule (advanced) (UC11)	8

1 INTRODUCTION

During university course registration, Concordia ENCS students are required to choose their own courses according to their program, prerequisites, time availabilities and preferences. This task can become very complex. However, there is currently no official application that help student organize their time tables. Students are actually forced to do so manually and use trial and error with the Concordia registration system. This creates extra traffic in the registration system and consumes precious studying time.

PlanZilla is web-based application that is designed to help students organize their schedule prior to using the Concordia Registration system. It helps students find the optimal schedule based on time constraints, preferences, program requirements and course sequence. The application takes care of finding all possible schedule combination for the student. All the student needs to do is give time constraints.

This document describes the software development process of PlanZilla. It contains the following steps: Identifying requirements, planning, scoping, designing, implementing and testing. Each section of this document describes a specific development process.

Currently, PlanZilla has finished its development phase with the release of version 1.4111. Future revisions are only maintenance releases.

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, the current official schedule for the upcoming semesters will be displayed. Students will have access to an official course sequence. It is up to the student whether or not to follow this sequence. Students can change their schedule by using an elaborated schedule generator that takes into account the courses to be taken (selected from the official sequence by default) and various time-related constraint to accommodate students with special needs. 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 2.1 shows a list of the objectives of each document.

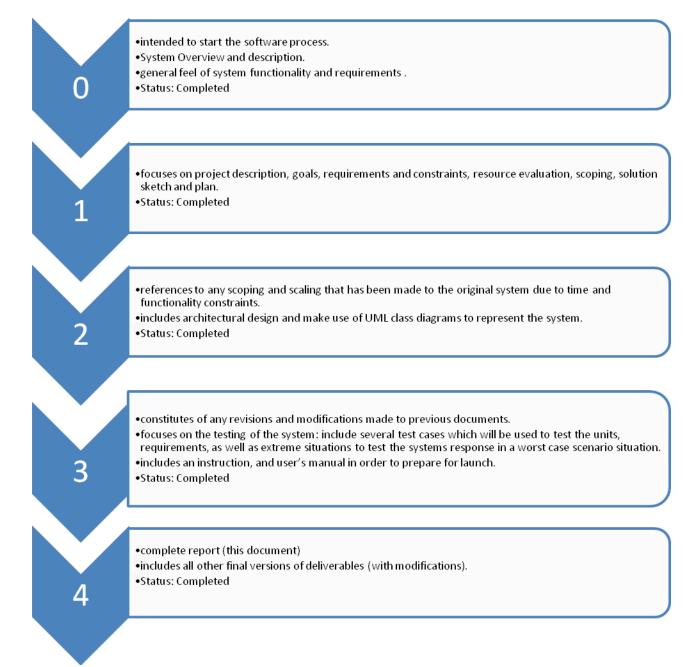


Figure 2.1 - Overview of the project's five deliverables

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 descriptions 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 preconditions 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.

Use Case 1 Open the Application (UC1)

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

Use Case 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	 ✓ The user has not logged into the system. ✓ The front-end is loaded and a login view is activated. 		
Step	1. User identifies itself to the system with its credentials.		
(success path)	2. System acknowledges		
Post-condition (success)	✓ A session is opened on the system for that user		
Failure end conditions	Wrong Credentials: × Message displayed in login window asking for the user to retry.		
Related Use Case	Open the application (UC1)		

Use Case 3 Logout (UC3)

Importance	5/5	Difficulty	1/5
Description	Action of signing out of the system.		
Actors	Registered user		
Goal	Actor terminates the se	ession so that it can no lo	onger be modified.
Pre-conditions	✓ A session is opened	l	
Step (success path)	 The user activates the sign out feature OR does not perform any action for a significant amount of time. Sign out request is acknowledged by the system. 		
Post-condition	✓ Session is closed on the server.✓ All front-end data is cleared.		
Failure End Condition	Failure End Condition × 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)		

Use Case 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	✓ A session is opened		
Step	1. User requests the "View Schedule" feature.		
(success path)	2. The system presents the user's schedule		
Post-condition			
Failure End Condition	× No schedule is available for current actor.		
Related Use Case	Login (UC2)		

Use Case 5 View a Room or Professor Schedule (UC5) (Scoped out)

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 consu	t the schedule of a roon	n or professor.
Pre-conditions			
	1. User requests the "View Schedule" feature.		
Step	2. The system presents the user's schedule.		
(success path)	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			

Use Case 6 Browse Course List (UC6)

Importance	5/5	Difficulty	3/5
Description	Course calendar can be b	prowsed 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)	 User activates the "Browse Course List" feature. System prompts about which types of courses to display. User indicates department and/or course number and/or professor and asks for the course list. System presents a list of corresponding courses. User selects a course in that list. System displays detailed information about the selected course. 		
Post-condition			
Failure End Condition			
Related Use Case			

Use Case 7 View Program Sequence (UC7)

Importance	2/5	Difficulty	1/5
Description	View the complete course sequence of the program in which the		
Description	student is enrolled.		
Actors	Registered user		
Goal	To inform the user about	their progression in the	e program's sequence.
Pre-conditions	✓ A session is opened		
Step	1. The user requests the "View Program Sequence" feature.		
(success path)	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		
ranure End Condition	current student. Maybe the student is not enrolled in a program.		
Related Use Case	Login (UC2)		

Use Case 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)	 The user activates the "View Academic Record" feature. 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)		

Use Case 9 Generate Schedule (UC9)

Importance	5/5	Difficulty	5/5
Description	In order to register for courses, the user needs to generate a schedule.		
Description	They can do so by selecti	ng courses.	
Actors	Registered user		
Goal	Student desiring to regis	ter for courses must gen	erate their schedule.
Pre-conditions	✓ A session is opened.		
	1. User activates the "Ge	nerate Schedule" featur	e.
	2. The system responds with a list of possible schedules.		
Ston	3. The user adopts one of the proposed schedules.		
Step (success path)	4. The system prompts for confirmation, specifying any currently		
(Success patil)	— 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		nester are dropped.	
Failure End Condition	× Displays error if no schedule can be made using selected courses.		
Related Use Case	Login (UC2)		

Use Case 10 View Activity Log (UC10) (Scoped out)

Importance	0/5	Difficulty	2/5	
Description	To access a detailed log of	of a user's account		
Actors	Registered user			
Goal	The user wishes to check	details about the occur	rence of events.	
Pre-conditions	← A session is opened	✓—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)			

Use Case 11 Generate Schedule (advanced) (UC11)

Importance	5/5	Difficulty	5/5
Doccrintion	In order to register for co	In order to register for courses, the user needs to generate a schedule.	
Description	They can do so by selecting courses and constraints.		
Actors	Registered user		
0 1	Student desiring to regist	ter for courses must gen	erate their schedule
Goal	with some conditions.		
Pre-conditions	✓ A session is opened.		
	1. User activates the "Ge	nerate Schedule" featur	e.
	2. The system responds v	with a list of possible sch	edules.
	3. User specifies which co	ourses they want to take	by rejecting the ones
	offered by the system and specifying new ones.		
Chan	4. User specifies time constraints during which they do not want class.		
Step	5. System presents a new set of possible schedules.		
(success path)	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 s	aved and selected cours	es are registered.
Post-condition	✓ Alternate courses (if any) for the present semester are dropped.		
	× System displays an e	rror if no schedule can b	e generated from the
Failure End Condition	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 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.

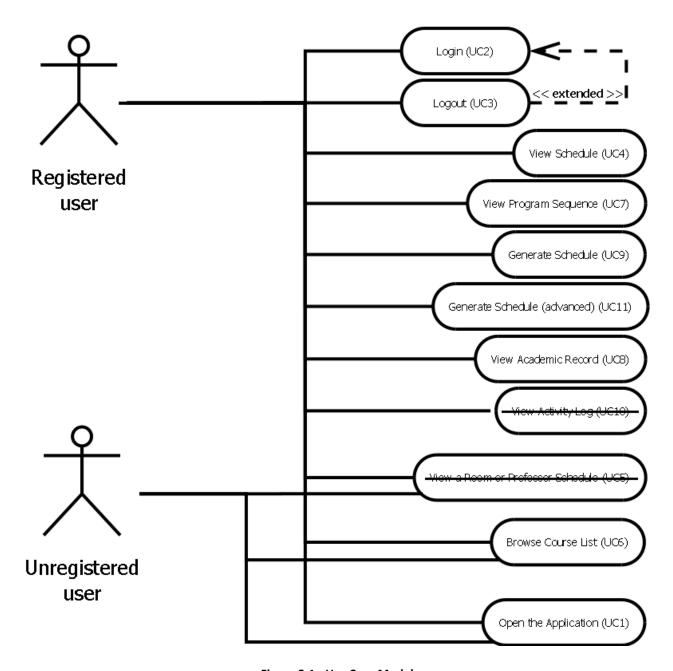


Figure 3.1 - Use Case ModelUse case model depicting the relationship between the actors and the use cases.

3.2 Domain Model

3.2.1 Domain Model Diagram

Figure 3.2 depicts the domain model of the project. There are a total of seven domain level objects (DLO).

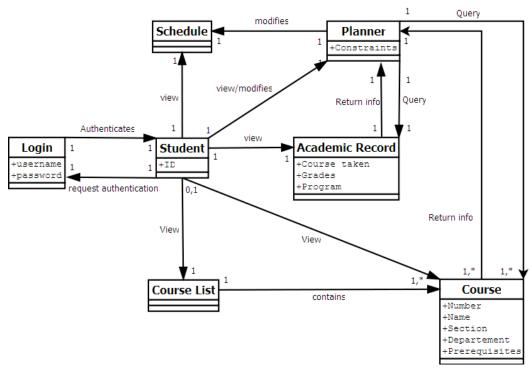


Figure 3.2 - Domain Model Diagram

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 it 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.

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 artefacts should be neglected and only the specific content of the page should be printed.

3.3.7 Workflow Resilience in Regards to Network Issues

A lot of client these days connect to the Internet wirelessly. When using a wireless connection, the signal might often be dropped due to various issues. When an application does not take this into account, random runtime error can occur. Users then have to refresh the page losing all inputted data and resetting the state of the application. Often this implies users have to restart the workflow from scratch. We need our application to properly handle such situations.

All requests to the server should be resumed once the connection has been re-established in the event of a network failure. The user should be notified of this.

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

A few low priority features have been removed from the application in order to meet expected release deadline. These features are:

- Warning users if their browser is not supported by the application (parts of use case 1).
- Displaying Professor or Class schedules under view schedule (use case 5).
- Accessing an Activity Log (Use case 10).
- The application will no longer save/register a schedule for the user (parts of Use Case 9 and 11). User should not need to access an already made schedule if already registered to ENCS registration system. This feature will be implemented in a future revision.

These features can be identified in this document as crossed out text.

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 sack 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 details this.

Activity Initial project discussions: Domain, functionalities, Interface Corey, Mathieu (Mat), Phuong-Anh-Vu (JB), Sébastien (Seb), Andreas, Eric, **Assigned To** 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 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.

Table 5.1 - Plan of the initial discussions of the project.

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 5.2 details this.

Table 5.2 - Plan of the general design of the project.

	Activity	General design discussion	
	Assigned To	Corey, Mat, JB, Seb, Andreas, Eric, Julia, Marc	
	Due date	Week 5, Friday	
*	Initiate discussion betw	veen team members on the various aspect of the design and establish the	
	software process and v	arious communication protocols to be used by the team.	
	Artefact	Requirement, scope and plan document	
	Costs	30	
>	Documentation of the I	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 t	he 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 5.3 details this.

Table 5.3 - Plan for the architecture and overall implementation decisions.

	Activity	Architecture / implementation discussion
	Assigned To	Corey, Mat, JB, Seb, Andreas, Eric, Julia, Marc
	Due date	Week 9, Friday
*	Establish the various pa	arts of the general architecture and implementation.
	Artefact	Architecture and design document
	Costs	30
>	Documentation of the a	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 5.4 details this.

Table 5.4 - Plan for the in-depth look at the implementation of the project.

	Activity	In depth implementation discussions	
	Assigned To	Corey, Mat, 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 5.5 details the plan for this.

Table 5.5 - Plan for the final implantation of the complete project.

	Activity	Project completion and final implementations
	Assigned To	Corey, Mat, 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 5.6 looks at the plan for the implementation of these tabs.

Table 5.6 - Plan for the implementation of the tab modules of the UI.

	Table 5.5 - I fail for the implementation of the tab modules of the of.		
	Activity	Coding and testing tab modules	
	Assigned To	UI Team (consisting of Mat & Julia)	
	Due date	Week 12, Friday	
*		uired to make a tab styled navigation interface using JavaScript and HTML. brough 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 modules.	page containing the course schedule for a student. It is visible from all other	
	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 5.7 shows the plan for the implementation of these control modules.

Table 5.7 - Plan for the implementation of the control modules of the UI.

	Activity	Coding and testing of control modules
	Assigned To	UI Team (consisting of Mat & Julia)
	Due date	Week 12, Friday
*	The actual implementa	tion – 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 construc	ct 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 5.8.

Table 5.8 - Plan for the logic tier of the application

Activity	Coding and testing of Logical Modules	
Assigned To	Corey, Marc, JB, Seb, Andreas, Eric	
Due date	Week 12, Friday	
All the underlying imp	lementation of the logic required to present a schedule to a user based on	
selected courses and t	emporary filters.	
Artefact	Academic record module	
Costs	30	
> The module responsib	le 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 w	ith the objects representing courses.	
Artefact	Registration	
Costs	25	
Validates necessary re	quirements 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 5.9.

Table 5.9 - 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.	,g and regional of the state of the
	databases.	Course Database Entries
	databases.	, , , , ,

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 5.10.

Table 5.10 - 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 5.11 shows the plan for these tasks.

Table 5.11 - 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	> 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 5.6.1-1 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.