System and Software Architecture Description (SSAD)

LEMA Pilot School Integrated Scheduling System

Team No. 12

Name	Primary Role	Secondary Role
David Wiggins	Project Manager	Developer
Aakash Shah	Prototyper	Developer
Kushalpreet Kaur	Developer	Developer
Thammanoon Kawinfruangfukul	Tester	Developer
Eunyoung Hwang	Architect	Developer
Louis Demaria	IIV&V	Developer
Mark Villanueva	QFP	Developer
Sangik Park	Developer	Developer

04/27/2012

Version History

Date	Author	Version	Changes made	Rationale
10/12/11	E.H.	1.0	• Introduction & System Analysis	• Initial draft for LEMA Pilot School Integrated Scheduling System v1.0
10/14/11	E.H.	1.1	• Fix 1.2 and change APSCS term into a scheduler on the diagrams and tables	• Correct bugs detected from IIV&V personnel through Bugzilla
10/17/11	E.H.	2.0	• Edit diagrams and add more use cases specification for 2.1.System Analysis Overview	• Correct diagram and pre-conditions for the process description according to the TA's comment
10/21/11	E.H.	2.1	• Correct diagrams on the System Analysis	• After getting comments from FCR ARB presentation
10/24/11	E.H.	3.0	• Correct diagrams on the system analysis, especially the integration part with the external system, the LEMA Integrated Family Accountability System	• After the meeting with the other team from CSCI577 developing Family Accountability System of the same organization as us, LEMA pilot school
11/21/11	E.H	3.1	• Add Technology-specific system design and Architectural styles, patterns and frameworks	• Draft DC Package for LEMA Integrated Scheduling System
11/29/11	E.H	3.2	• Add a couple of sequence diagram and class diagram	• Prepare for DCR ARB session
12/05/11	E.H	3.3	• Correct a couple of sequence diagrams and 4. Architectural styles, patterns and frameworks	• After the DCR ARB session, it is the final package for the 577a
02/05/12	E.H.	4.0	• Updated ADO.NET framework into Symfony on diagrams	• Updating for the Draft Rebaselined DC Package
02/09/12	E.H	4.1	• Corrected bugs reported on Bugzilla and updated diagrams	• Updating before Rebaselined DCR ARB
02/15/12	E.H	4.2	• Detailed artifacts and information diagram	• Updating for the Rebaselined DC Package
03/26/12	E.H	5.0	• Updated some use-cases and artifacts and information diagram	• Updating after getting comment from TA and for the first Initial Operational Capability Package
04/16/12	E.H	6.0	• Updated overall reflecting current development status	• Updating before Code Review session
04/18/12	EH, TK	6.1	• Updated diagrams and use case descriptions	• Refining for Code Review session
4/27/12	EH	6.2	• Changed data implementation part from Symfony to MySQL connector as a stereotype for diagrams.	• After Code review, reflected comments

Table of Contents

Table of Tables

Table of Figures

1. Introduction

1.1 Purpose of the SSAD

The purpose of the System and Software Architecture Description (SSAD) is to document the object-oriented analysis and design of the system being developed. The developer uses the SSAD as reference to the system architecture and it is to help the maintainer and clients to understand the structure of the system after the proposed system is delivered.

1.2 Status of the SSAD

This document currently contains all the parts for System and Software Architecture Description of the LEMA Integrated Scheduling System. This version is revised after the Code Review session reflecting the 577b staff and it is part of IOC second set with overall the design part for synchronization between development and architecture design.

2. System Analysis

2.1 System Analysis Overview

The main purpose of the LEMA Pilot School Integrated Scheduling System is to enhance the performance for optimal course scheduling. The LEMA Integrated Scheduling system gets course preference input directly from the student and helps a counselor to review students' preference information and confirm them. The system also provides with administrator's course input and teacher course allocation. The system's main feature is to take the constraints defined by LAUSD norms or LEMA disciplines and to produce optimal scheduling based on those constraints and the data of students' course register and teacher capacity. Furthermore, students are able to track their progresses while inputting course preference. Students and teacher finally can view their final schedule through the website.

2.1.1 System Context

Figure 1: System Context Diagram

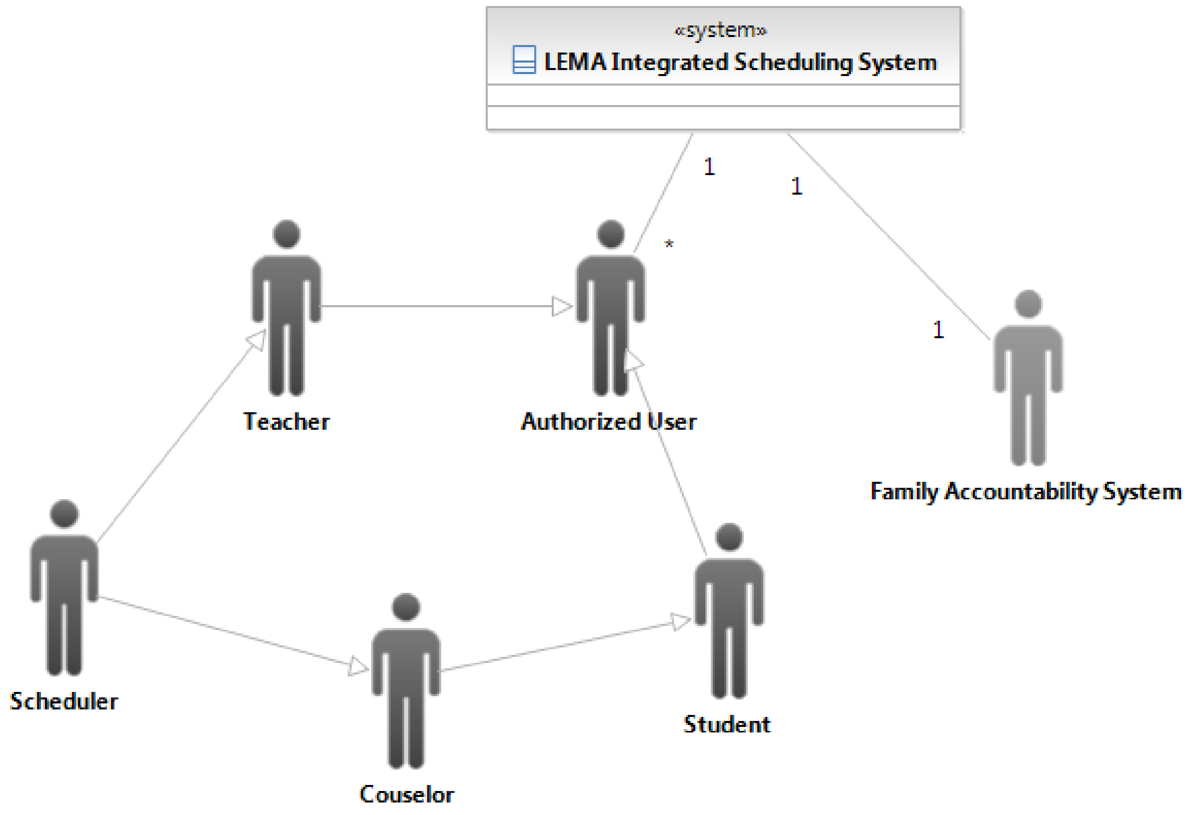

«system»
LEMA Integrated Scheduling System
1
1
*
1
Teacher
Authorized User
Family Accountability System
Scheduler
Student
Couselor

Table 1: Actors Summary

Actor	Description	Responsibilities
Authorized user	Users representing the general user who can be authorized by the system	- Log in and out of the system
student	Users who register courses for the next semester and track their progresses during semester	- Input and update course preference - View student's final schedule - View their progress
teacher	Users who register their teaching capability of courses and follow course schedule	- View teacher's final schedule - View enrolled students
counselor	users reviewing each students registered course preference, checking whether student registered proper courses, and finalizing them	- View submitted students' course preference - Modify them if needed - Approve students' courses preference
Scheduler	A person assigning courses to teachers and setting constraints for scheduling	- Assign teachers to ccourses - Input and update course information - Import district course list - Export teacher, course and activity information - Import the final schedule into our database
LEMA Integrated Family Accountability System	A separate LEMA pilot school system, being built by another CSCI577 team, with which the LEMA Integrated Scheduling System	- Provide each user's authentication information to our system - Provide faculty and student basic information via REST

2.1.2 Artifacts & Information

Figure 2: Artifacts and Information Diagram

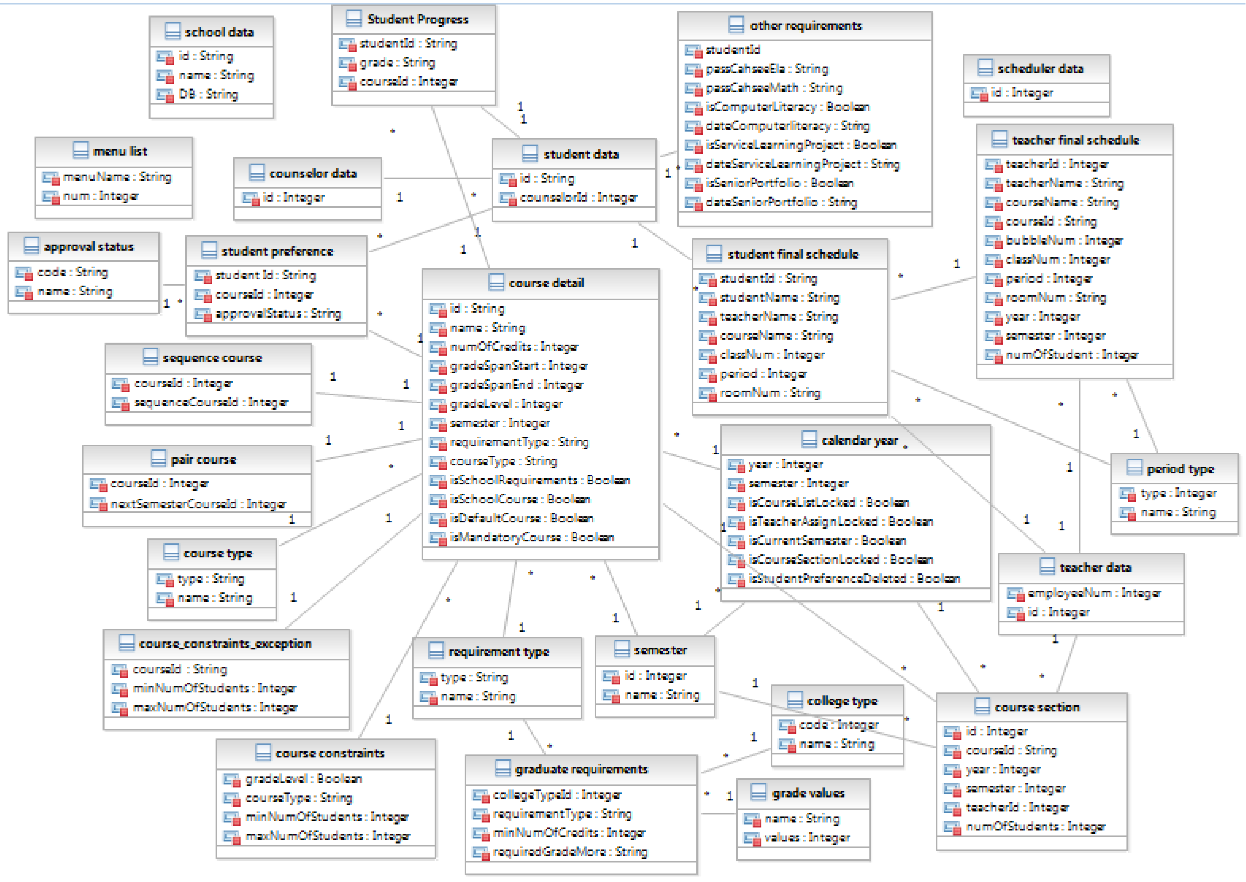


Table 2: Artifacts and Information Summary

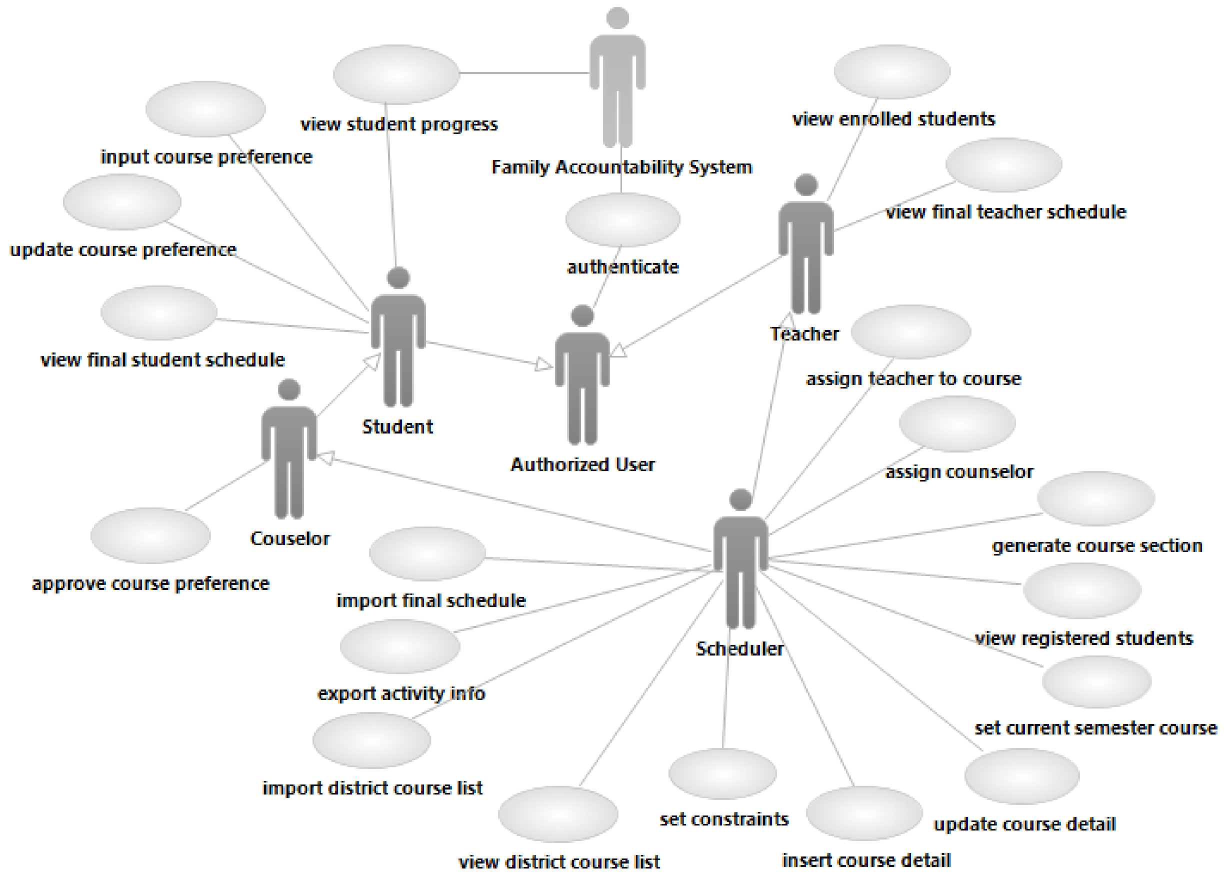
Artifact	Purpose
ATF-1: counselor data	Contains a counselor id from the Accountability System
ATF-2: student data	Contains student id from the Accountability System
ATP-3: teacher data	Contains teacher id from the Accountability System
ATP-4: scheduler data	Contains scheduler id as an administrator from the Accountability System
ATF-5: student course preference	Contains student's course preference information which student entered from the course preference form. However, the final schedule for each student has nothing to be related with the student preference because the final schedule is imported by the scheduler from the Columbia system.
ATP-6: course detail	Contains course detail with course id, name, and number

	of credits etc., which a scheduler entered from the course detail form
ATP-7: other requirements	Contains requirement information to graduate highschool
ATP-8: pair course	Contains current semester's course id and the next semester's course id
ATP-9: course type	Contains name and code of the course type such as mandatory or elective
ATP-10: course constraints	Contains the information about what grade level and what type of course needs how many numbers of students as minimum and maximum.
ATP-11: course section	Contains the course section information which is calculated by the system according to the class constraints. For example, when there are many students over maximum number of that course, the course can have more than one section. This section is formation is to feed the scheduling COTS as an activity data which contains the mapping of the teacher and course.
ATF-12: student final schedule	Contains student and course data created by the .CSV file format final schedule from the Columbia system
ATF-13: student progress	Contains student's course and its grade information to track student's progress
ATF-14: requirement type	Contains requirement type(A to G) code and name
ATF-15: sequence course	Contains course need to be scheduled side by side on the periods
ATF-16: menu list	Contains the menu(use case) list which the Scheduling System provides
ATF-17: graduate requirements	Contains graduate requirements categories, requirement type(A to G), minimum number of credits for each requirement type and required grade to graduate or to go to certain type of college or university
ATF-18:grade values	Contains grades and its values in order to weigh the value of the grades showing grade A is the higher one compared with the grade C
ATF-19:college type	Contains the name and code of the graduate requirement categories for the student progress tracking such as high school graduation, university of California, or community college
ATF-20: calendar year	Contains year and semester information which is entered beginning of the course management
ATF-21: school data	Contains school code and database name for allowing only that school to access the specific database
ATF-22: Semester	Contains semester id and name
ATF-23: Approval status	Contains approval status code and name such as pending, denial, or approved for student's course preference
ATF-24: Period type	Contains period type from 1 to 10 including club and

	advisory period
ATF-25: Teacher final schedule	Contains teachers final schedule which will allow a teacher to view their final schedule and this data come from School system (Columbia) as csv file format
ATF-26: Course constraints exception	Contains min and max number of students for specific courses as exceptions from general course constraints table

2.1.3 Behavior

Figure 3: Use-case Diagram



2.1.3.1 Authentication

2.1.3.1.1 Authenticate

Table 3: Process Description - authenticate

Identifier	UC-1: authenticate
Purpose	Authenticate and login user
Requirements	SR-3 Authentication to the system

Development Risks	Integration to get each user's authentication information from the external system which is LEMA Integrated Family Accountability System(FAS)
Pre-conditions	The system database is properly initialized
Post-conditions	If user is authorized, the user is able to access to appropriate functionality depending on the type of the user and the user's session is generated; otherwise, the user is denied to access to the system.

Table 4: Typical Course of Action – authenticate: successful

Seq#	Actor's Action	System's Response
1	[user] Inputs user name and password	
2	[user]Clicks "LEMA Scheduling System"	
3		Send user name and encrypted password to Family Accountability System
4	[Family Accountability System] return authentication result	
5		Check if the id exists in database and if not insert id into the user table
6		Request the user's permission menu list which is accessible to the user type
7	[Family Accountability System] return the list of the permission menu for that type of the user	
8		Compare the user's menu list from the Family Accountability System with each name at the menu list table from the database
9		Display the verified menu list on the homepage of the Scheduling System

After the user login to the system, every page needs to get the list of the permission menu from Family Accountability System using REST (step 5-6 on table 4).

Table 5: Alternate Course of Action – authenticate: failure

Seq#	Actor's Action	System's Response
1-4	Refer to typical course of action	
5		If the login status is fail, redirect to the login page with the error message

		(unauthorized user)

Table 6: Exceptional Course of Action – authenticate: failure to get a response from the external system

Seq#	Actor's Action	System's Response
1-3	Refer to typical course of action	
4		Display "No response from the Accountability system" message
5	Click ok button	
6		Redirects the user to the previous page

2.1.3.2 Course management

2.1.3.2.1 Import district course list

Table 7: Process Description – import district course list

Identifier	UC-2: import district course list
Purpose	Import district course list to the database
Requirements	CR-2 Lock in certain classes
Development Risks	none
Pre-conditions	The scheduler have the district course list file
Post-conditions	District course list saved into database

Table 8: Typical Course of Action - import district course list

Seq#	Actor's Action	System's Response
1	Clicks the choose file button	
2		Show dialog for the user to input the district course list file
3	Select the district course list file	
4	Click confirm button on dialog	
5	Click import button	
6		Get the district course list file
7		Record the district course list file to the database
8		Send confirmation message

Table 9: Exceptional Course of Action – import district course list: invalid file type

Seq#	Actor's Action	System's Response
1-5	Refer to typical course of action	
4		Display "Please check the file to import!" message
5	Click ok button	

2.1.3.2.2 View district course list

Table 10: Process Description – view district course list

Identifier	UC-3: view district course list
Purpose	view district course list
Requirements	CR-2 Lock in certain classes
Development Risks	none
Pre-conditions	The district course file is imported to the database.
Post-conditions	Display district course list

Table 11: Typical Course of Action - view district course list

Seq#	Actor's Action	System's Response
1	Click "course profile list" menu	
2		Get the district course list from the database
3		Display the district course list

2.1.3.2.3 Input course detail

Table 12: Process Description – input course detail

Identifier	UC-4: input course detail
Purpose	Get each course's profile from a scheduler
Requirements	CR-2 Lock in certain classes
Development Risks	none
Pre-conditions	A scheduler already has the details of courses list for this semester
Post-conditions	course details saved into database

Table 13: Typical Course of Action - input course detail

Seq#	Actor's Action	System's Response
1	Input course id and name	
2	Select requirement type(A-G)	
3	select the number of credits, min and max number of students	
4	Click "save" button	
5		Insert all data selected or input into course detail at database
6		Send confirmation message

Table 14: Exceptional Course of Action - input course detail : did not input course id or name

Seq#	Actor's Action	System's Response
1-4	Refer to typical course of action	
5		Display an alert message "Please enter course ID" or "Please enter course name"
6	Click ok button	
7		Redirects the input course detail form

2.1.3.2.4 Update course detail

Table 15: Process Description – input course detail

Identifier	UC-5: update course detail
Purpose	A scheduler can update if there is any changes to the each course's detail
Requirements	CR-2 Lock in certain classes
Development Risks	none
Pre-conditions	A scheduler already input course detail form but it is still before the beginning of the student's course preference registration
Post-conditions	course details updated into database

Table 16: Typical Course of Action - input course detail

Seq#	Actor's Action	System's Response
1	Click the course that the user would like to change	
2		Redirect to course profile page
3	Make some changes	
4	Click save button	
5		update changed information into

		course profile and related tables at database
6		display confirmation message

Table 17: Exceptional Course of Action - input course detail: did not input course name

Seq#	Actor's Action	System's Response
1-4	Refer to typical course of action	
5		Display an alert message "Please enter course name"
6	Click ok button	
7		Redirects the input course detail form

2.1.3.2.5 Set current semester course

Table 18: Process Description – set current semester course

Identifier	UC-6: set current semester course
Purpose	A scheduler can set the current semester and available courses for that semester.
Requirements	CR-2 Lock in certain classes
Development Risks	none
Pre-conditions	The district course file is imported to the database.
Post-conditions	current semester courses are recorded into the database

Table 19: Typical Course of Action – set current semester course

Seq#	Actor's Action	System's Response
1	Selects current semester from the combo box	
2	Clicks "Set current semester" button	
3		Display "This semester has been set!" message
4	Clicks check boxes to select courses which will be open	
5		Save selected course into the course section table
6	Clicks "lock" button	
7		Update Lock status into database.
8		display confirmation message

2.1.3.2.6 Set constraints

Table 20: Process Description – set constraints

Identifier	UC-7: set constraints
Purpose	A scheduler can set constraints for generating course segment
Requirements	CR-4: Schedule generation
Development Risks	none
Pre-conditions	The system database is properly initialized
Post-conditions	constraint details updated into database

Table 21: Typical Course of Action - set constraints

Seq#	Actor's Action	System's Response
1	Click constraints management page	
2		Display constraint details
3	Make some changes	
4	Click confirm button	
5		update changed information into course constraints table
6		display updated constraint details

Table 22: Alternate Course of Action - set constraints

Seq#	Actor's Action	System's Response
1-4	Refer to typical course of action	
5	Click cancel button	
6		Redirects the scheduler main page

2.1.3.2.7 Generate course section

Table 23: Process Description – generate course section

Identifier	UC-8: generate course section
Purpose	A scheduler generates course section which can be calculated by the number of students who wants to take the same course based on the class minimum and maximum numbers defined by the LAUSD norm
Requirements	CR-7 map teachers to course sections
Development Risks	none
Pre-conditions	Getting students course preference has been closed and the counselor confirmed them. The maximum and minimum number of student per class on the grade level should be designated

	already.
Post-conditions	Course section data saved into database

Table 24: Typical Course of Action - generate course section

Seq#	Actor's Action	System's Response
1	Click "course list by semester" menu	
2		Calculate the course section data based on the student course preference and the course constraints information
3		Display the list of course sections with the number of students registered
4	Click "Lock" button	
5		Save course section data into the course section with the number of students for each course
6		Save the lock status into calendar year table
7		Send confirmation message

Table 25: Alternate Course of Action - generate course section

Seq#	Actor's Action	System's Response
1-3	Refer to typical course of action	
4	Click "unlock" button	
5		Save the lock status into calendar year table
6		Redirects the course list by semester

2.1.3.2.8 View registered students

Table 26: Process Description – view registered students

Identifier	UC-9: view registered students
Purpose	A scheduler can update if there is any changes to the each course's detail
Requirements	CR-4: Schedule generation
Development Risks	none
Pre-conditions	Getting students course preference has been closed and the counselor confirmed them. The maximum and minimum number of student per class on the grade level should be designated

Post-conditions	Display registered students

Table 27: Typical Course of Action - view registered students

Seq#	Actor's Action	System's Response
1	[User] Click a link of specific course in registered student column	
2		Redirect to Student List on Section Conflicts page
3		Request student's detail data by sending student id
4	[the Family Accountability System] send the student's detail data back to the scheduling system	
5		Display register students' information providing the link to the student registration page

Table 28: Exceptional Course of Action – view registered students: failure to get a response from the external system

Seq#	Actor's Action	System's Response
1-3	Refer to typical course of action	
4		Display "No response from the Accountability system" message

2.1.3.3 Student Course Registration

2.1.3.3.1 Input course preference

Table 29: Process Description - input course preference

Identifier	UC-10: input course preference
Purpose	Get course preference from students
Requirements	CR-1:online application for students subject registration
Development Risks	Integration with the Family Accountability System to get student detail information
Pre-conditions	Courses for this semester have been already registered by a scheduler
Post-conditions	the student course preference saved in the database

Table 30: Typical Course of Action - input course preference

Seq#	Actor's Action	System's Response

1	[User]Click "student course registration" menu	
2		Request student's detail data by sending student id
3	[the Family Accountability System] send the student's detail data back to the scheduling system	
4		Display student's detail data and the default course list by A-G categories
5	[User]select preferred courses for next semester in check box	
6	[User]click confirm button	
7		Insert them into the database
8		Send confirmation message

Table 31: Alternate Course of Action - input course preference

Seq#	Actor's Action	System's Response
1-6	Refer to typical course of action	
7		Display an error message "Your selected courses are over the maximum numbers of courses"
8	Click ok button	
9		Redirects the course preference form

Table 32: Exceptional Course of Action – input course preference: failure to get a response from the external system

Seq#	Actor's Action	System's Response
1-3	Refer to typical course of action	
4		Display "No response from the Accountability system" message

2.1.3.3.2 Update course preference

Table 33: Process Description - update course preference

Identifier	UC-11: update course preference
Purpose	Update course preference by students
Requirements	CR-1:online application for students subject registration
Development Risks	none
Pre-conditions	Student already submitted the preference form but it is still before

	the deadline for the student’s course preference registration
Post-conditions	some student course preference updated in the database

Table 34: Typical Course of Action - update course preference

Seq#	Actor’s Action	System’s Response
1	[User]Click "student course preference" menu	
2		Request student's detail data by sending student id
3	[the Family Accountability System] send the student's detail data back to the scheduling system	
4		Display student's detail data and the preference course list
5	select other courses from the combo box	
6	click confirm button	
7		update them into the database
8		Send confirmation message

Table 35: Exceptional Course of Action – update course preference: failure to get a response from the external system

Seq#	Actor’s Action	System’s Response
1-3	Refer to typical course of action	
4		Display “No response from the Accountability system” message

2.1.3.3.3 Approve course preference

Table 36: Process Description - approve course preference

Identifier	UC-12: approve course preference
Purpose	Approve student’s course preference
Requirements	CR-3: Student course requests are viewed/approved/rejected by counselor
Development Risks	None
Pre-conditions	All students submitted their course preference
Post-conditions	the student course preference updated in the database

Table 37: Typical Course of Action - approve course preference

Seq#	Actor's Action	System's Response
1		display the list of students
2	Click a student's unapproved link on the summary list	
3		display the student's course registration page
4	Review the courses and select approved in the combo box	
5	click confirm button	
6		Update data
7		Redirect to the page for the students list

Table 38: Alternate Course of Action – approve course preference

Seq#	Actor's Action	System's Response
1-3	Refer to typical course of action	
4	Review the courses and select unapproved in the combo box	
5	Click confirm button	
6		Update approval status into the database
7		Redirect to the page for the students list

2.1.3.4 Teacher management

2.1.3.4.1 Assign teacher to course

Table 39: Process Description - assign teacher to course

Identifier	UC-13: assign teacher to course
Purpose	Update teaching courses by scheduler
Requirements	CR-7:Map teachers to course sections
Development Risks	none
Pre-conditions	A scheduler managed the conflict on the course section and course sections have been locked but it is still before exporting acrivity data from the database into the CSV file.
Post-conditions	teacher course assignments updated in the database

Table 40: Typical Course of Action – update teacher course

Seq#	Actor's Action	System's Response
1	select teachers for the course from the combo box	
2	click save button	
3	Click lock button	
4		update them into the database
5		Send confirmation message
6		Combo box has been deactivated

Table 41: Alternate Course of Action – update teacher course

Seq#	Actor's Action	System's Response
1-2	Refer to typical course of action	
3	Click unlock button	
4		update lock status into the database
5		Send confirmation message
6		Combo box has been activated

2.1.3.5 Schedule Management

2.1.3.5.1 Export activity information

Table 42: Process Description - export activity information

Identifier	UC-14: export activity information
Purpose	Generate activity(course section and teacher assignment) information to feed the off-the-shelf scheduling application
Requirements	CR-4: Schedule generation
Development Risks	need module to create text file(.csv) from database with the synchronized prototype
Pre-conditions	All activity information should already be saved on database
Post-conditions	CSV files of activities from the Database created

Table 43: Typical Course of Action - export activity information

Seq#	Actor's Action	System's Response
1	Click "export" button	
2		Generate CSV file for activity which is the assignment as activities between teacher and course information from the

		database
3		Save it into the download folder of a scheduler's pc

2.1.3.5.2 Import final schedule

Table 44: Process Description – import final schedule

Identifier	UC-15: import final schedule
Purpose	Save the final schedule created by the Columbia system into the database
Requirements	CR-4: Schedule generation
Development Risks	Format of the final schedule text file and the database schema need to match exactly
Pre-conditions	The Columbia system at LEMA from the LAUSD already generated the final schedule and imported it into the text file(.csv)
Post-conditions	The final schedule saved in the database

Table 45: Typical Course of Action – import final schedule

Seq#	Actor's Action	System's Response
1	Click "browse" button	
2	Select the CSV file to import	
3	Click student final schedule "import" button	
4		Read the students final schedule from the text file
5		Insert the students final schedule data into the database
6		Display confirmation message

Table 46: Alternative Course of Action – import final schedule

Seq#	Actor's Action	System's Response
1-2	Refer to typical course of action	
3	Click teacher final schedule "import" button	Display an error message "An error occurred during importing the final teacher schedule"
4		Read the teachers final schedule from the text file
5		Insert the teachers final schedule data into the database

6		Display confirmation message

Table 47: Exceptional Course of Action – import final schedule: failure to import csv file

Seq#	Actor's Action	System's Response
1-3	Refer to typical course of action	
4		Display "An error occurred during importing the final teacher schedule!" message

2.1.3.5.3 View final student schedule

Table 48: Process Description – view final student schedule

Identifier	UC-16: view final student schedule
Purpose	Provide a student to view the final course schedule
Requirements	CR-4: Schedule generation
Development Risks	None
Pre-conditions	The final schedule has been already imported into the database
Post-conditions	Display the student's final schedule

Table 49: Typical Course of Action - view final student schedule

Seq#	Actor's Action	System's Response
1	[user]Click "final schedule" button	
2		Request student's detail data by sending student id
3	[the Family Accountability System] send the student's detail data back to the scheduling system	
4		Display student's detail and final schedule

Table 50: Exceptional Course of Action – input course preference: failure to get a response from the external system

Seq#	Actor's Action	System's Response
1-3	Refer to typical course of action	
4		Display "No response from the Accountability system" message

2.1.3.5.4 View final teacher schedule

Table 51: Process Description – view final teacher schedule

Identifier	UC-17: view final teacher schedule
Purpose	Provide a teacher to view the final teaching course schedule
Requirements	CR-4: Schedule generation
Development Risks	none
Pre-conditions	The final schedule has been already imported into the database
Post-conditions	Display the teacher's final teaching schedule

Table 52: Typical Course of Action - view final teacher schedule

Seq#	Actor's Action	System's Response
1	Select year and semester from the combo box	
2		Request teacher's detail data by sending student id
3	[the Family Accountability System] send the teacher's detail data back to the scheduling system	
4		Display teacher's detail and final teaching schedule

Table 53: Exceptional Course of Action – input course preference: failure to get a response from the external system

Seq#	Actor's Action	System's Response
1-3	Refer to typical course of action	
4		Display "No response from the Accountability system" message

2.1.3.5.5 View enrolled students

Table 54: Process Description – view enrolled students

Identifier	UC-18: view enrolled students
Purpose	Provide a teacher to and scheduler to view students who enrolled in the class
Requirements	CR-4: Schedule generation
Development Risks	none
Pre-conditions	The final schedule has been already imported into the database

Post-conditions	Display the teacher’s final teaching schedule

Table 55: Typical Course of Action - view enrolled students

Seq#	Actor’s Action	System’s Response
1	[User] Clicks the course link to view enrolled students	
2		Get enrolled student lists
3		Request student's detail data by sending student id
4	[the Family Accountability System] send the student's detail data back to the scheduling system	
5		Display enrolled students’ information

Table 56: Exceptional Course of Action – view enrolled students: failure to get a response from the external system

Seq#	Actor’s Action	System’s Response
1-4	Refer to typical course of action	
5		Display “No response from the Accountability system” message

2.1.3.6 View student progress

Table 57: Process Description – view student progress

Identifier	UC-19: view student progress
Purpose	Provide students progress for them to realize where they are standing to graduate or to go to the college while inputting student course preference
Requirements	CR-5: track the progress of the particular student
Development Risks	none
Pre-conditions	Other system has the past and current semester's grade for each course of the user
Post-conditions	displays the grade and the analyzed data

Table 58: Typical Course of Action – view student progress

Seq#	Actor's Action	System's Response
1	Click student preference menu	
2		Display student progress tracking page showing data from the student progress table and graphical view for the level of they have been achieved

2.1.3.7 Counselor management

2.1.3.7.1 Assign counselor

Table 59: Process Description - assign counselor

Identifier	UC-20: assign counselor
Purpose	Allocate counselor to the student
Requirements	CR-3 Student course requests are viewed/approved/rejected by counselor
Development Risks	none
Pre-conditions	The system already have a counselor id and students id
Post-conditions	each student's assigned counselor id saved in the student data

Table 60: Typical Course of Action - input counselor data

Seq#	Actor's Action	System's Response
1	Click "counselor-student assignment" menu	
2		Show the list of the student by grade level
3	Select for the student group by grade level and check students	
4	Select counselor in combo box	
5	Click "assign students" button	
6		Allocate the counselor to the student by updating the student table's counselor id in database
7		Display updated counselor assignment information on the student list

2.1.4 Modes of Operation

The LEMA Pilot School Integrated Scheduling System, as we see it, will operate only in a single mode. Therefore, we do not need to say anything more about modes of operation.

2.2 System Analysis Rationale

Based on the requirements captured by the analysis of the system and how users will be interacting with the proposed system, the team has identified the following courses of users: *Note: Such users connect to the application form webpage over the Internet, with different permissions and functionalities based on the type of user.*

1. Scheduler:
   Scheduler is the administrator at the LEMA Pilot School who is responsible for allocating teacher courses, entering the course list, import the final schedule and exporting the teacher, course, and activity information from the proposed database. A scheduler's job is to use the off-the-shelf product, scheduling system. This product is not considered as the part of the system we are going to develop ,however, a scheduler use it to generate schedule with some actions; import previously generated teacher, course, activity's text file into the off-the-shelf scheduling system, input some constraints, click generate button to see the final generated schedule. The scheduler is also responsible for importing the final schedule generated by legacy system, Columbia. The scheduler can access the off-the-shelf scheduling program only through workstations at LEMA Pilot School.

2. Student:
   The students at LEMA Pilot School are responsible for entering the course preferences and able to view the final schedule.

3. Teachers:
   The teachers are the faculty members at LEMA Pilot School. They are able to view the final schedule from the website.

4. Counselor:
   The counselor is responsible for reviewing and editing the course preferences for each student based on whether the pre-requisites are met for particular course or not. The counselor can either approve or disapprove the course request.

There is one external system actors that interface with the LEMA Integrated Scheduling System. It is:

1. LEMA Integrated Family Accountability System – provides student and teacher data to the LEMA Integrated Scheduling System. In the other way around, the LEMA Integrated Scheduling System provides the course data and the final schedule information.

The LEMA Integrated Family Accountability System is being built by other CSCI577 teams. The clients have requested for the project to be integrated and for relevant modules and data to be shared. The LEMA Integrated Scheduling System is, therefore, being designed to share data provided by this other system.

3. Technology-Specific System Design

3.1 Design Overview

3.1.1 System Structure

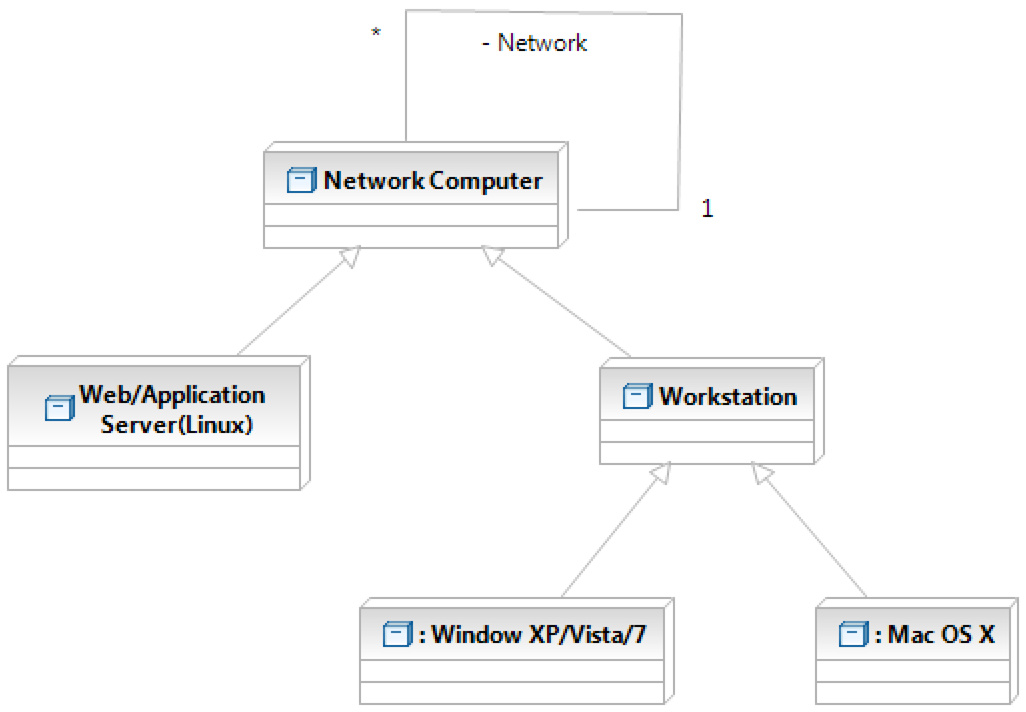


Figure 4: Hardware Component Course Diagram

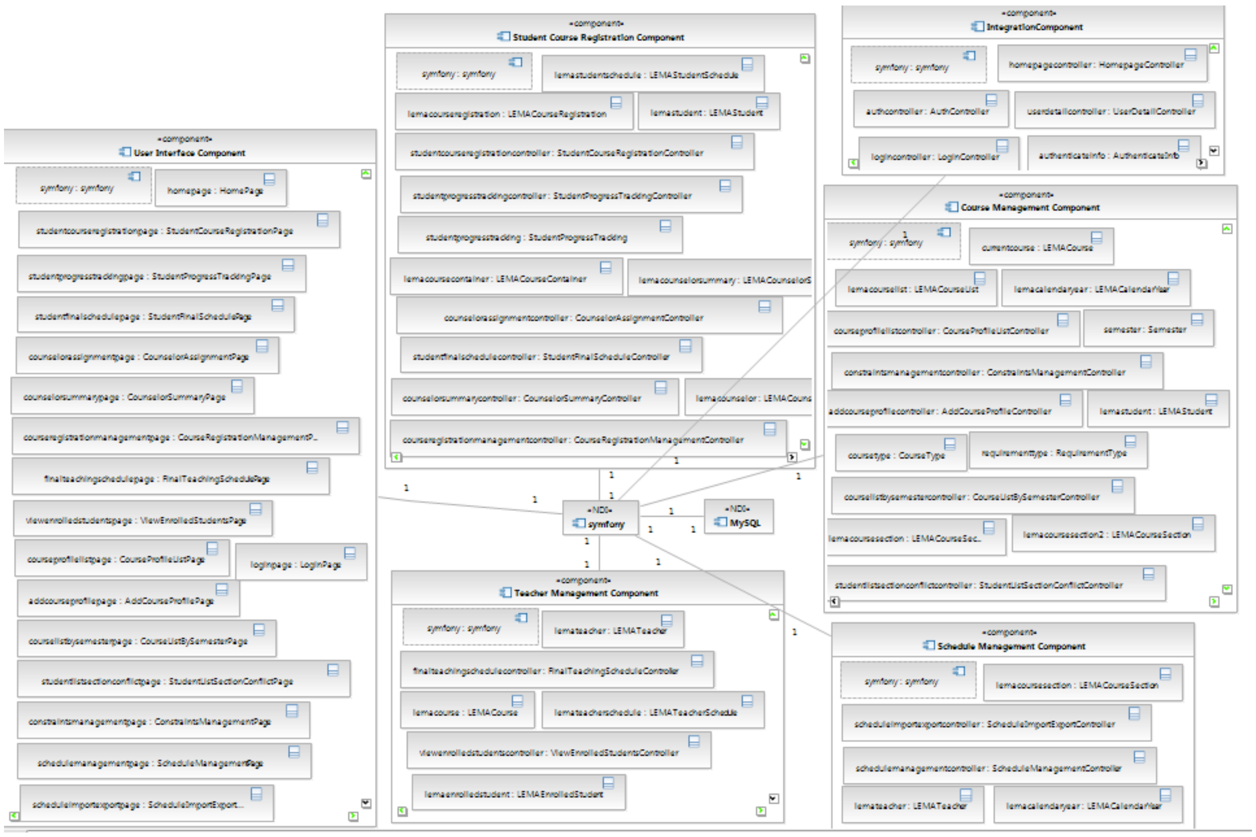

«component»
User Interface Component
symfony : symfony
homepage : HomePage
studentcourseregistrationpage : StudentCourseRegistrationPage
studentprogresstrackingpage : StudentProgressTrackingPage
studentfinalschedulepage : StudentFinalSchedulePage
counselorassignmentpage : CounselorAssignmentPage
counselorsummarypage : CounselorSummaryPage
courseregistrationmanagementpage : CourseRegistrationManagementP...
finalteachingschedulepage : FinalTeachingSchedulePage
viewenrolledstudentspage : ViewEnrolledStudentsPage
courseprofilelistpage : CourseProfileListPage
loginpage : LoginPage
addcourseprofilepage : AddCourseProfilePage
courselistbysemesterpage : CourseListBySemesterPage
studentlistsectionconflictpage : StudentListSectionConflictPage
constraintsmanagementpage : ConstraintsManagementPage
schedulemanagementpage : ScheduleManagementPage
scheduleimportexportpage : ScheduleImportExport...
«component»
Student Course Registration Component
symfony : symfony
lemastudentschedule : LEMAStudentSchedule
lemacourseregistration : LEMACourseRegistration
lemastudent : LEMAStudent
studentcourseregistrationcontroller : StudentCourseRegistrationController
studentprogresstrackingcontroller : StudentProgressTrackingController
studentprogresstracking : StudentProgressTracking
lemacoursecontainer : LEMACourseContainer
lemacounselorsummary : LEMACounselorS
counselorassignmentcontroller : CounselorAssignmentController
studentfinalschedulecontroller : StudentFinalScheduleController
counselorsummarycontroller : CounselorSummaryController
lemacounselor : LEMACouns
courseregistrationmanagementcontroller : CourseRegistrationManagementController
«component»
IntegrationComponent
symfony : symfony
homepagecontroller : HomepageController
authcontroller : AuthController
userdetailcontroller : UserDetailController
logincontroller : LoginController
authenticateInfo : AuthenticateInfo
«component»
Course Management Component
symfony : symfony
currentcourse : LEMACourse
lemacourselist : LEMACourseList
lemacalendaryear : LEMACalendarYear
courseprofilelistcontroller : CourseProfileListController
semester : Semester
constraintsmanagementcontroller : ConstraintsManagementController
addcourseprofilecontroller : AddCourseProfileController
lemastudent : LEMAStudent
coursetype : CourseType
requirementtype : RequirementType
courselistbysemestercontroller : CourseListBySemesterController
lemacoursesection : LEMACourseSec...
lemacoursesection2 : LEMACourseSection
studentlistsectionconflictcontroller : StudentListSectionConflictController
«NDI»
symfony
«NDI»
MySQL
1
«component»
Teacher Management Component
symfony : symfony
lemateacher : LEMATeacher
finalteachingschedulecontroller : FinalTeachingScheduleController
lemacourse : LEMACourse
lemateacherschedule : LEMATeacherSchedule
viewenrolledstudentscontroller : ViewEnrolledStudentsController
lemaenrolledstudent : LEMAEnrolledStudent
«component»
Schedule Management Component
symfony : symfony
lemacoursesection : LEMACourseSection
scheduleimportexportcontroller : ScheduleImportExportController
schedulemanagementcontroller : ScheduleManagementController
lemateacher : LEMATeacher
lemacalendaryear : LEMACalendarYear

Figure 5: Software Component Course Diagram

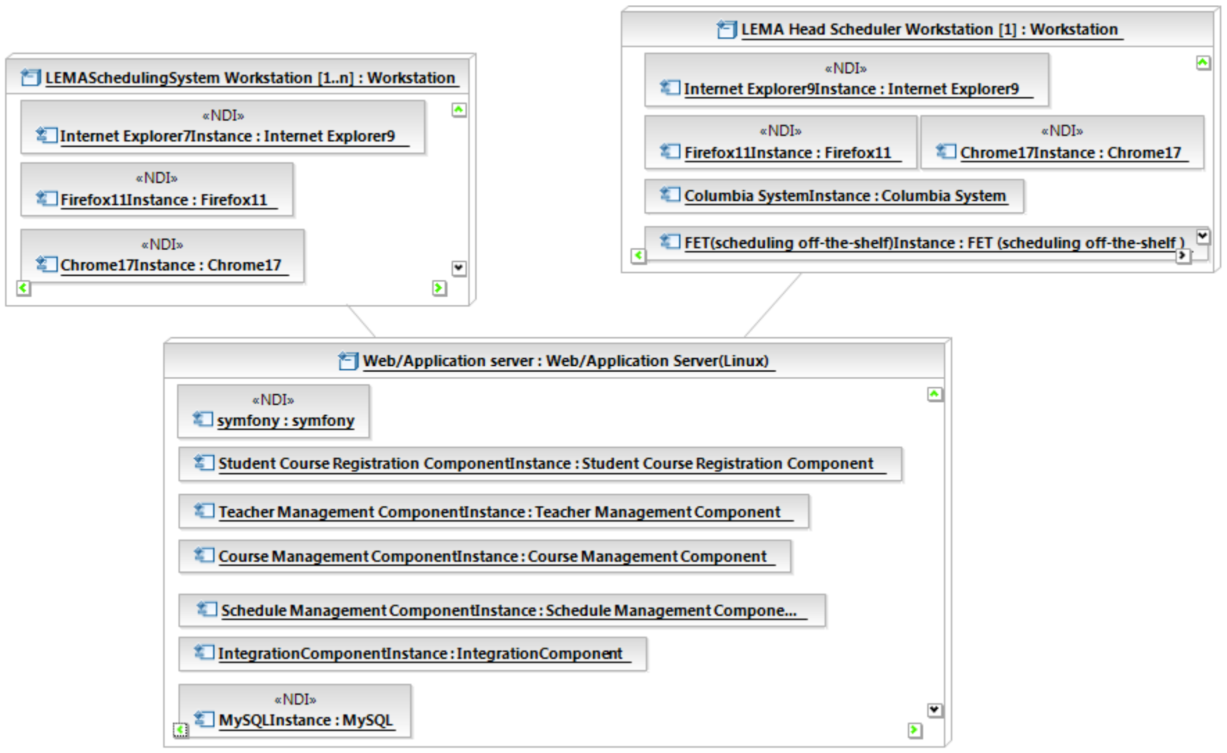

LEMASchedulingSystem Workstation [1..n] : Workstation
«NDI»
Internet Explorer7Instance : Internet Explorer9
«NDI»
Firefox11Instance : Firefox11
«NDI»
Chrome17Instance : Chrome17
LEMA Head Scheduler Workstation [1] : Workstation
«NDI»
Internet Explorer9Instance : Internet Explorer9
«NDI»
Firefox11Instance : Firefox11
«NDI»
Chrome17Instance : Chrome17
Columbia SystemInstance : Columbia System
FET(scheduling off-the-shelf)Instance : FET (scheduling off-the-shelf)
Web/Application server : Web/Application Server(Linux)
«NDI»
symfony : symfony
Student Course Registration ComponentInstance : Student Course Registration Component
Teacher Management ComponentInstance : Teacher Management Component
Course Management ComponentInstance : Course Management Component
Schedule Management ComponentInstance : Schedule Management Compone...
IntegrationComponentInstance : IntegrationComponent
«NDI»
MySQLInstance : MySQL

Figure 6: Deployment Diagram

Table 61: Hardware Component Description

Hardware Component	Description
Workstation	A workstation is the user's machine that the user connects to the Scheduling System over the Internet connection. The platform that the user uses is Microsoft Windows and the Mac OS with the internet browsers such as Firefox or Internet Explorer.
Web/Application Server(Linux)	A web server and the application server are on the same web hosting server. A web server accepts connections from the workstation and the Scheduling system's web application resides on the application server. The application server communicates with the web server and provides database accessibility and business logic.

Table 62: Software Component Description

Software Component	Description
User Interface Component	This component contains Scheduling System web pages for use by all Scheduling System user depending on their authentication. The primary component is the Student course allocation and Scheduling Management.

Student Course Registration Component	This component performs functions involved in allocating courses to students. Main functionalities are getting student course preference, providing student's available courses and showing student progress tracking
Teacher Management Component	This component contains that a scheduler allocate the teacher to the specific courses they can teach
Course Management Component	This component manage course information providing from the class constraints to the open class for this semester
Schedule Management Component	This component performs two different kinds of functionalities. To provide teacher and course data to the scheduling COTS, this component support to export these data by creating CSV files. To get the final schedule from the Columbia system. it supports to import the final schedule into the database system.
Integration Component	This is for the integration part with the other CSCI577 team 4 developing the Family Accountability System.
Database Component	MySQL is used as a database to maintain the course and scheduling details for the project

3.1.2 Design Courses

3.1.2.1 Integration with the external system class

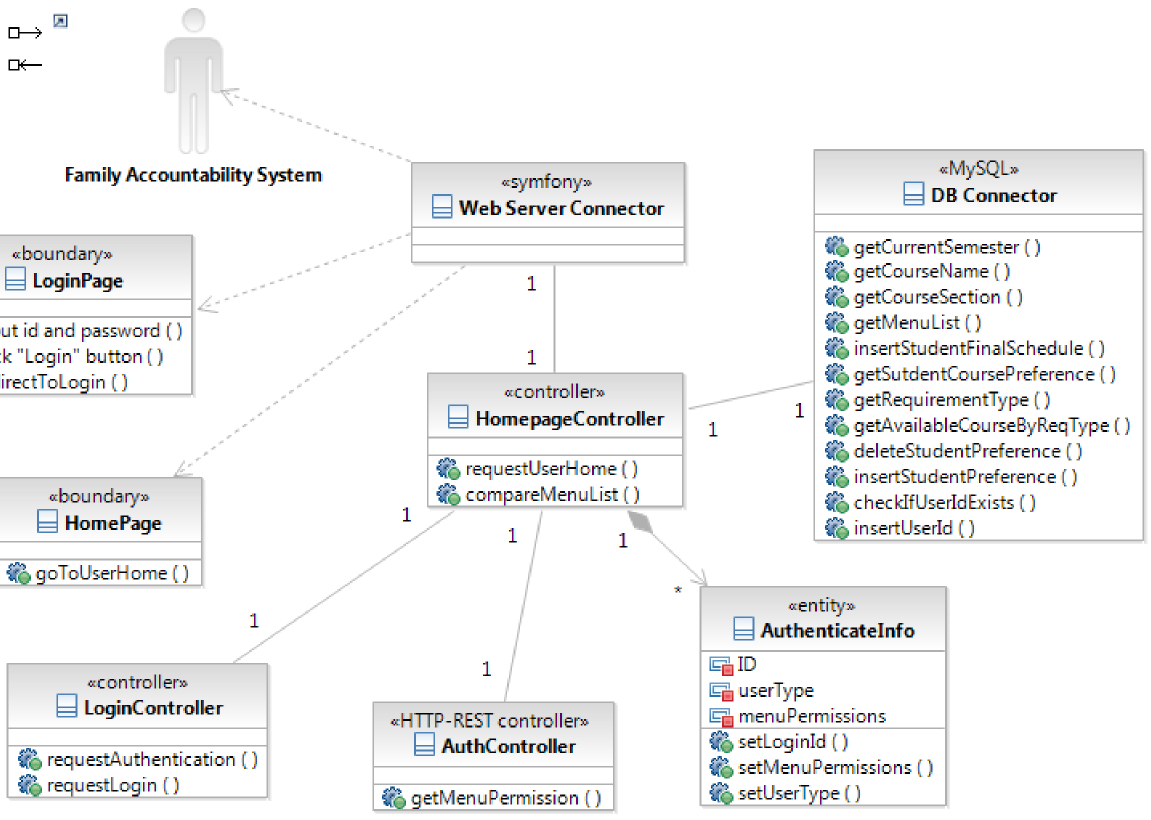


Figure 7: Integration with the external system Class Diagram

Table 63: Integration with the external system class diagram Description

Course	Type	Description
Family Accountability System	Actor	The external system integrating with our system. LoginController sends id and pass word to FAS and get return if the user is authenticated or not. We also get permitted menu list for the user type through REST API
AuthenticateInfo	Entity	contains user's id, type, permitted menu list
LoginPage	Entity	Getting id and password from the user
LoginController	controller	Request authentication to FAS. If it is successful, check user table and if we do not initiate that user, insert the id into the user

		table.
Homepage	boundary	Request HomepageController for the proper home page for the user and display it
HomepageController	controller	Request menu permission to FAS through REST and if the page is allowed to access then display the menu depending on the user menu permission
AuthController	HTTP-REST controller	Requests for the user's menu permission information to the Accountability System
DB Connector	MySQL Connector	Provides easier access to the Database

3.1.2.2 Student course preference class

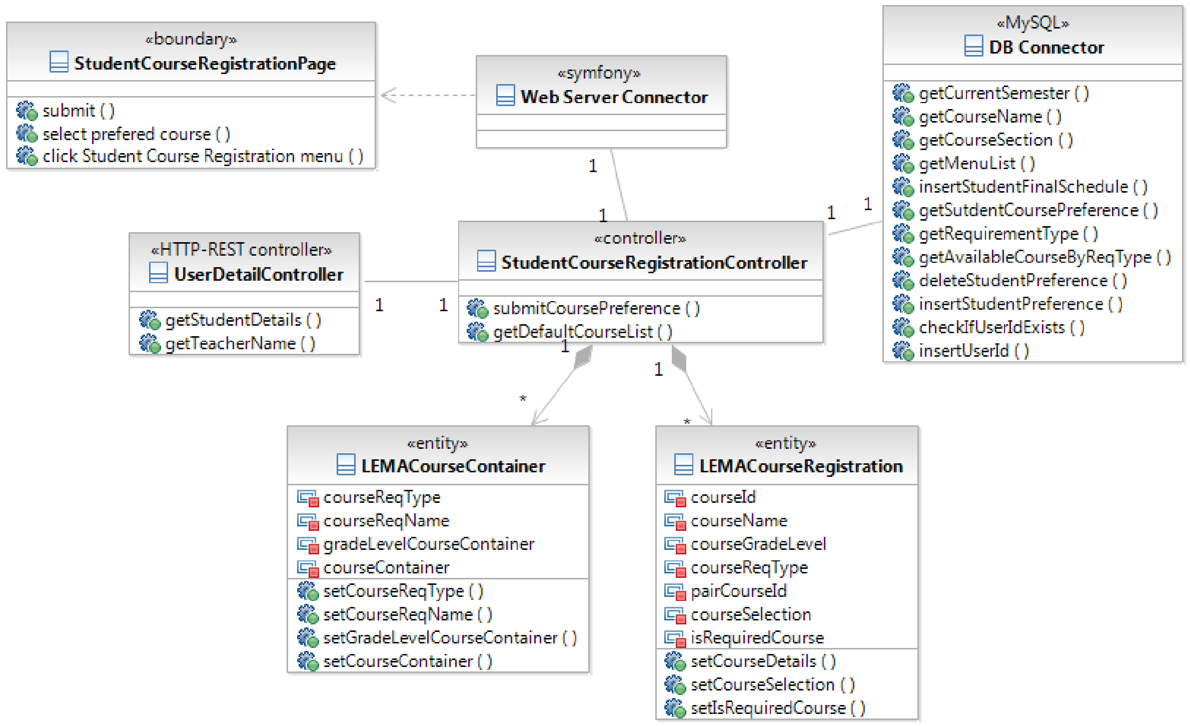


Figure 8: student course preference Class Diagram

Table 64: student course preference class diagram Description

Course	Type	Description
StudentCourseRegistrationPage	boundary	Contains student's default course lists and be able to submit student's course preference

DBConnector	MySQL Connector	Provides easier access to the Database
UserDetailController	HTTP-REST controller	Gets each user's detail data from the accountability system
StudentCourseRegistrationController	controller	Contains the logic how to display default course list for student to register and submit student's course registration information
LEMACourseContainer	entity	Contains requirement type and name with the arrays of student's grade level and optional course list
LEMACourseRegistration	entity	Contains the course information for default course list for the student

3.1.2.3 Schedule Management class

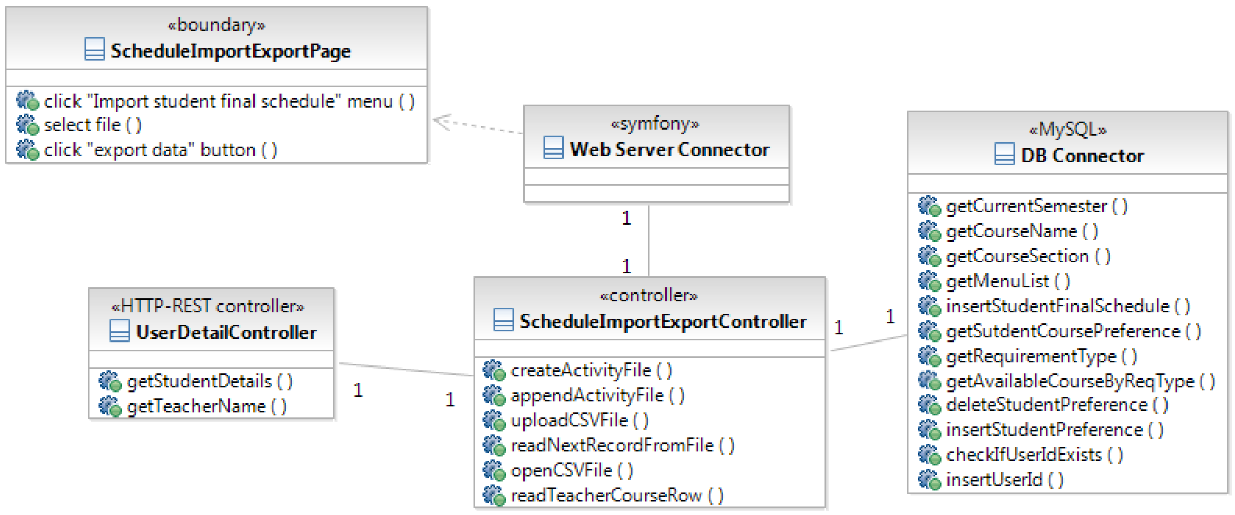


Figure 9: Export Activity Info & Import Final Schedule Class Diagram

Table 65: Export Activity Info & Import Final Schedule Class Diagram Description

Course	Type	Description
ScheduleImportExportPage	boundary	Contains schedule management menus such as export and import schedule or course information into or from our database
DBConnector	MySQL Connector	Provides easier access to the Database
UserDetailController	HTTP-REST controller	Gets each user's name data from the accountability system

ScheduleImportExportController	controller	Contains how to export data into activity CSV files to feed them into the FET application and the logic how to import the final schedule file into the database in the Scheduling System

3.1.3 Process Realization

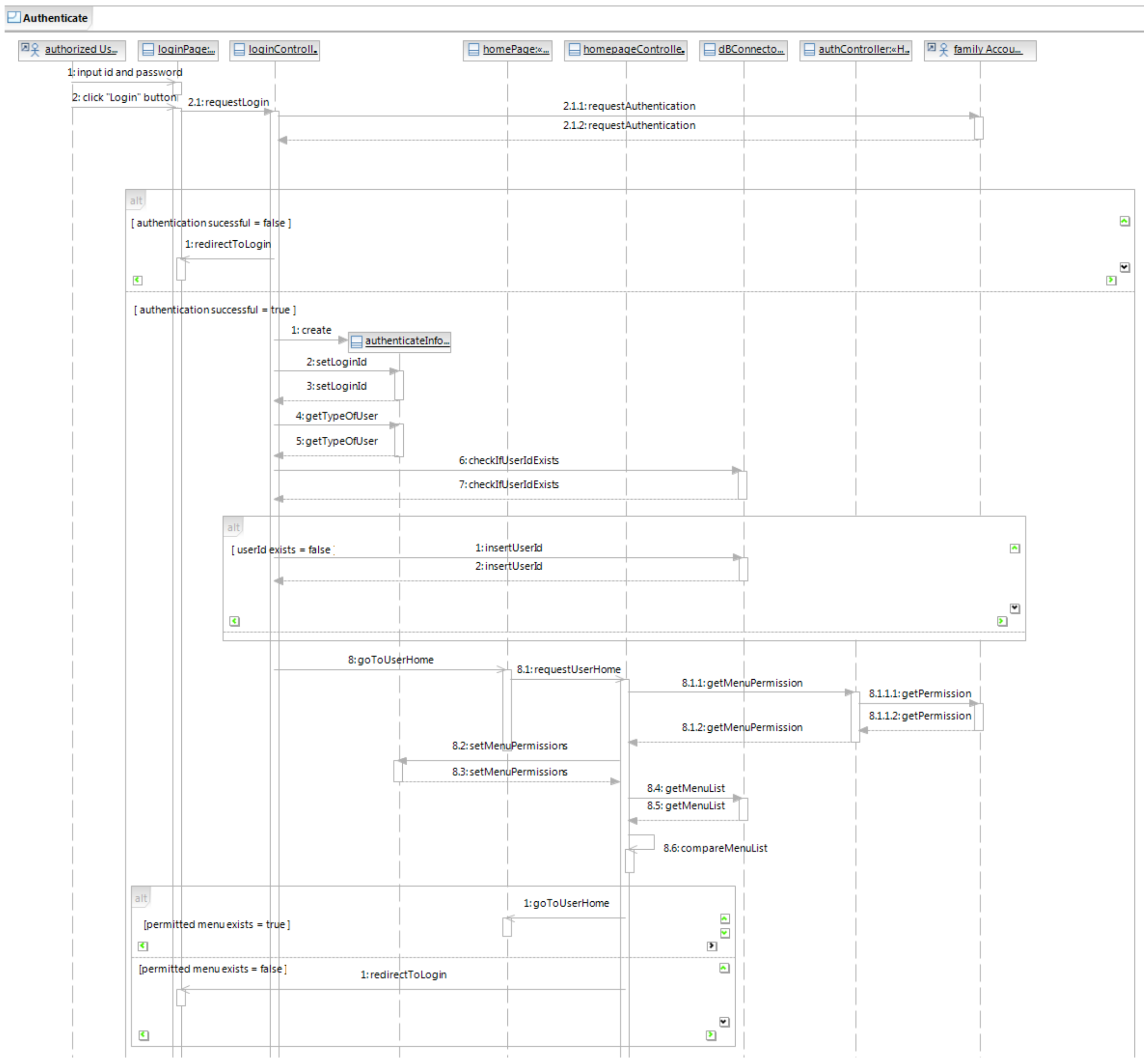


Figure 10: Authenticate Sequence Diagram

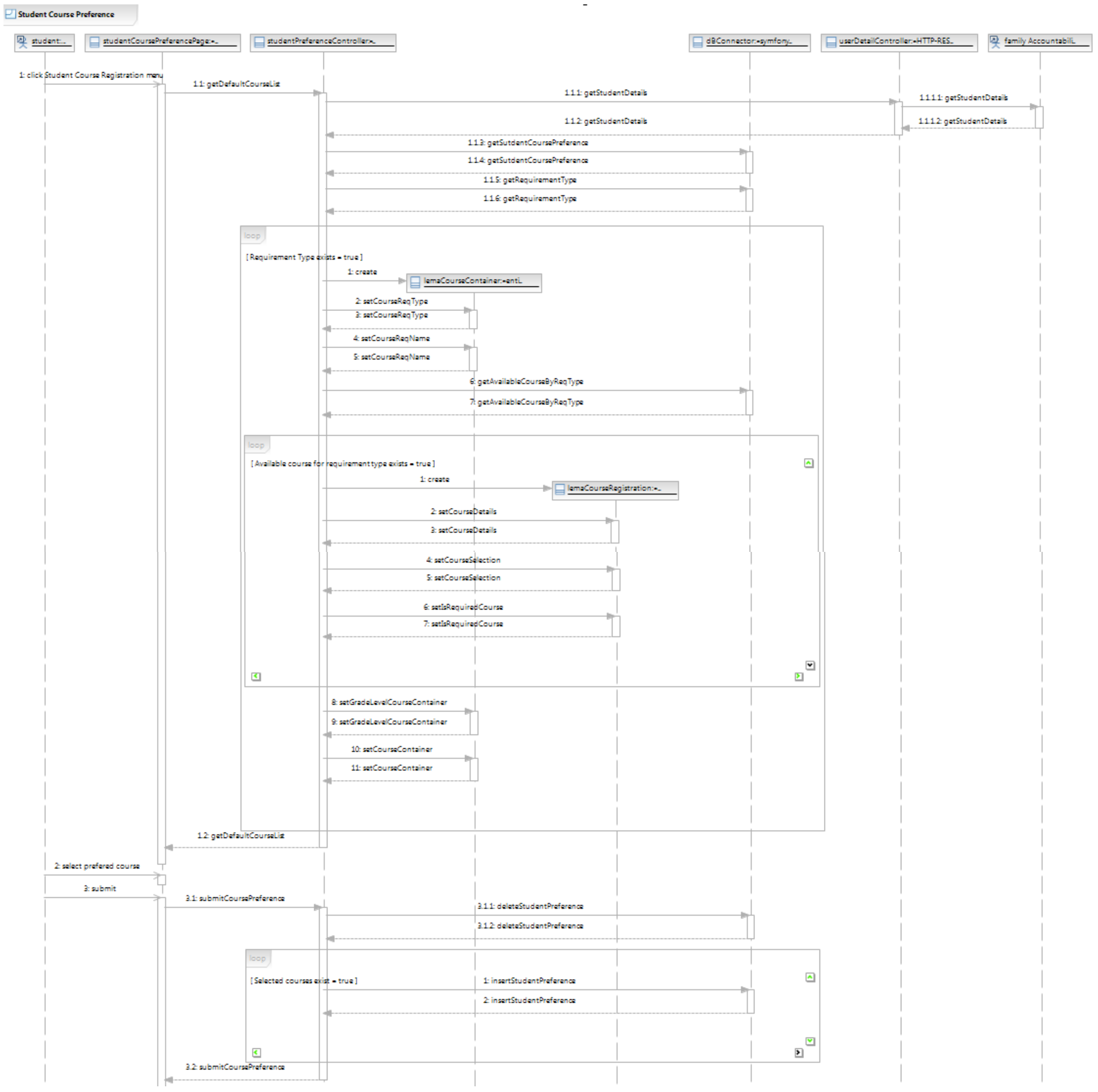

Student Course Preference
student:...
studentCoursePreferencePage:«...
studentPreferenceController:«...
dBConnector:«symfony...
userDetailController:«HTTP-RES...
family Accountabili...
1: click Student Course Registration menu
1.1: getDefaultCourseList
1.1.1: getStudentDetails
1.1.1.1: getStudentDetails
1.1.2: getStudentDetails
1.1.1.2: getStudentDetails
1.1.3: getSutdentCoursePreference
1.1.4: getSutdentCoursePreference
1.1.5: getRequirementType
1.1.6: getRequirementType
loop
[Requirement Type exists = true]
1: create
IemaCourseContainer:«enti...
2: setCourseReqType
3: setCourseReqType
4: setCourseReqName
5: setCourseReqName
6: getAvailableCourseByReqType
7: getAvailableCourseByReqType
loop
[Available course for requirement type exists = true]
1: create
IemaCourseRegistration:«...
2: setCourseDetails
3: setCourseDetails
4: setCourseSelection
5: setCourseSelection
6: setIsRequiredCourse
7: setIsRequiredCourse
8: setGradeLevelCourseContainer
9: setGradeLevelCourseContainer
10: setCourseContainer
11: setCourseContainer
1.2: getDefaultCourseList
2: select prefered course
3: submit
3.1: submitCoursePreference
3.1.1: deleteStudentPreference
3.1.2: deleteStudentPreference
loop
[Selected courses exist = true]
1: insertStudentPreference
2: insertStudentPreference
3.2: submitCoursePreference

Figure 11: Student Course Preference Sequence Diagram

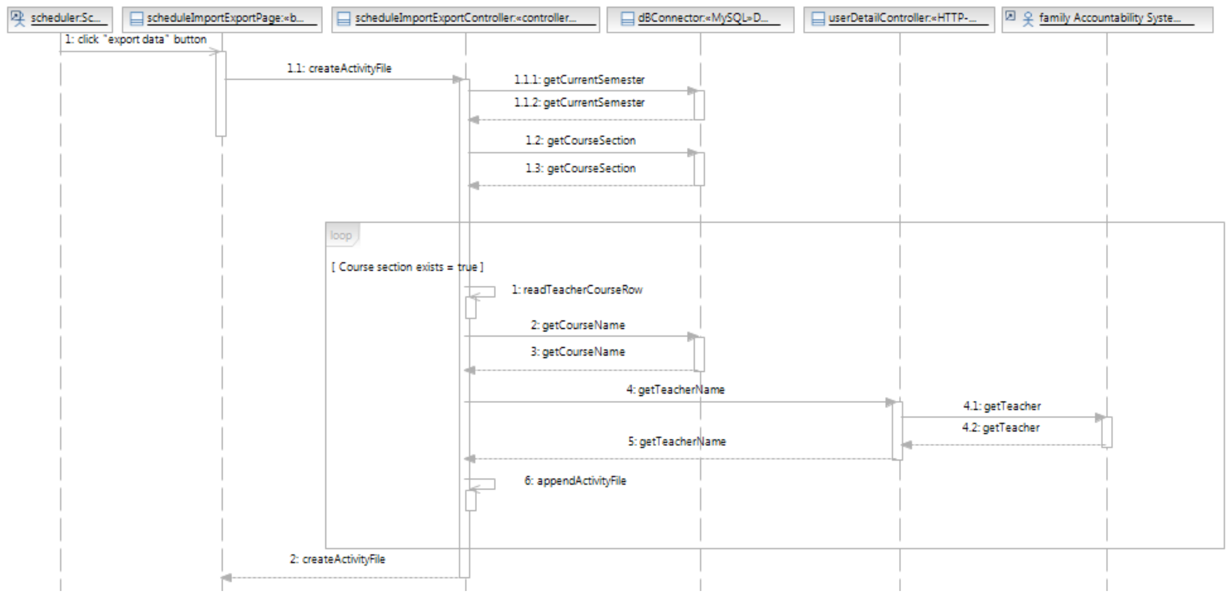


Figure 12: Export Activity Information Sequence Diagram

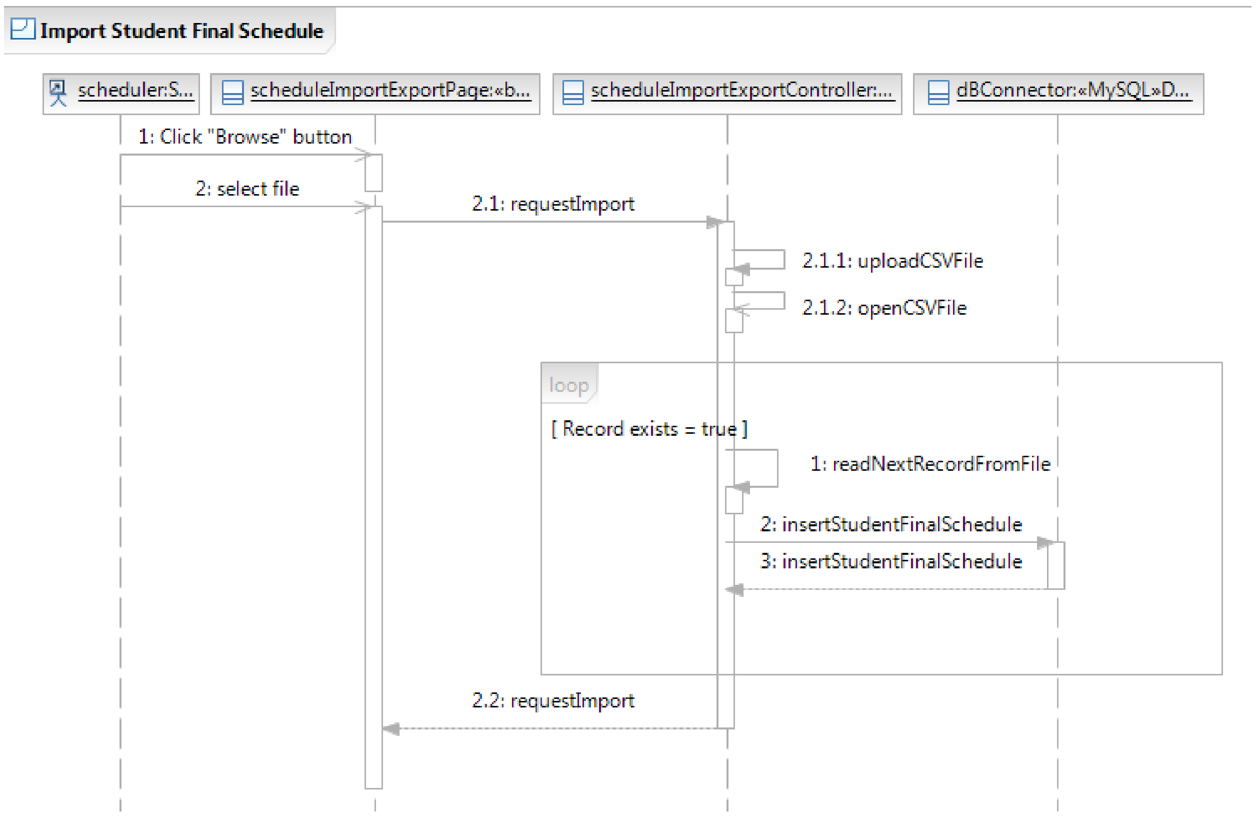


Figure 13: Import Student Final Schedule Sequence Diagram

3.2 Design Rationale

A user input id and password in LEMA Scheduling System, then get authentication result from FAS. If it fails, redirect to the log in page. If it is successful, check if the id exists in our database. If it is not initiated, insert id into the user table. The homepage displays the proper menu for the right user by comparing the permitted menu list between FAS and LEMA scheduling system database.

In the Student course preference, permitted student user click the menu and then the page gets the default course list by getting the student detail from the FAS first and then get each requirement type's course list by grade level. After displaying the default course list, student selects preferred courses then the controller deletes the previous course preferences of the student and inserts selected course preference into database.

To export activity information from our database, a scheduler click activity export button on the webpage. Then the schedule management page requests creating the CSV file to the controller. The controller also gets the name of the courses from the database. The controller gets teachers id from the database and get teacher name through the REST controller from the FAS until there is no teacher data exists in our database. After getting teacher and course name, the controller appends the name to the activity CSV file. It creates the activity entity until there is no more course section information in the database.

In order to import student final schedule from the CSV file, a scheduler clicks the Import menu and the Schedule Import Export Page requests import to the controller. The controller reads record from the file and inserts them into the database row by row until it is the end of the file.

4. Architectural Styles, Patterns and Frameworks

Table 66: Architectural Styles, Patterns, and Frameworks

Name	Description	Benefits, Costs, and Limitations
Three-Tier Architecture(MVC) with Symfony	We will be defining Entity classes to separate Business logic from the Controllers. • We followed MVC architectural pattern, but we implemented using MySQL connector instead of Data Doctrine provided in Symfony.	Separation of concerns and isolation of changes for a scalable environment and for development.
REST	Simple XML based on web services that can be produced and consumed without any dependencies on any languages or platforms	It can speak with other systems only such as the Family Accountability System that is required for sharing data.