Engi9874 Project: Course Scheduling at a University

Obi Nixon & Inaam Ahmed
MUN# 201472933
MUN# 201692944

Milestones:

Create course schedule, view that schedule with section, room#, instructor and time interval. Manipulate weekly schedule, identify conflicts, enter constraints (hard & Soft), suggest schedule with minimized conflicts.

Deliverable Contents

- **Requirements**
  a) **User Stories/ Functional Requirements**
  b) **Non-Functional Requirements**
  c) **Lexicon**
  d) **Domain Diagram**

Functional requirements

User Stories

R0: As an Administrator, I can create new schedule, open existing schedule from schedules in Database (text files).
R1: As an Administrator, I can create room with name and seating capacity
R2: As an Administrator, I can remove the room
R3: As an Administrator, I can edit/change the room name and capacity
R4: As an Administrator, I can create course with name and course enrollment limit
R5: As an Administrator, I can remove course
R6: As an Administrator, I can edit the course name and enrollment limit
R7: As an Administrator, I can create Sections
R8: As an Administrator, I can create Subsections
R9: As an Administrator, I can remove individual subsections
R10: As an Administrator, I can remove complete Subsection
R11: As an Administrator, I can create set of students leaf
R12: As an Administrator, I can create set of students composite
R13: As an Administrator, I can create variable length time intervals
R14: As an Administrator, I can create instructor name
R15: As an Administrator, I can associate subject to instructor
R16: As an Administrator, I can view the schedule from (section, room#, Instructor and time interval) point of view
R17: As an Administrator, I can export finished schedule to a printable format (e.g. pdf)
R18: As an Administrator, I can edit the schedule
R19: As an Administrator, I can view the unused time intervals

Non-Functional Requirements

N-R1: All of code writing is based on the Java SE platform, version 1.8

Lexicon

Administrator: Person designated for scheduling
Schedule: Time sheet used to synchronize sections with instructors according to room availability at certain time interval
Database: Schedule stored on disk called database (Text Files)
Room: Class room having chairs for students and teaching aids
Seating Capacity: Capacity of students to sit inside room
Course: Subject that instructor is going to teach
Enrollment Limit: Maximum limit of students in one course
Section: Group of students in one category (e.g. Engineering students, Business students etc.)
Subsection: Group of students in one category further divided into subsections (Computer Eng., Electrical Eng., etc.)
Leaf: each student with name and student Number.
Composite: student group in one subsection (MASCE, MEng., etc.)
Time interval: duration of time assigned one session of a course
Instructor: Person designated to teach a course
Instructor Type: Which type of courses instructor going to teach
Subject: same context as course.

Use Case

Use Case “start”
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R0
Precondition: The schedule has not been created.
Typical flow:

1: The Admin requests the Scheduler to open
2: Scheduler asks the Admin to create a new schedule or open an existing schedule.
   i. if the Admin chooses a new schedule, the Scheduler creates a new schedule.
   ii. Else if the user chooses an existing schedule, the Scheduler opens an existing schedule from the database, based on the path provided.

Postcondition: “A schedule is set up”
Alternative path: If the Admin selects cancel in step 2, the use case stops with postcondition “the Admin selected cancel”

Use Case “create, edit, remove room”
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R1, R2, R3
Precondition: The room hasn’t been created yet, or a room has been created and needs to be changed or removed.
Typical flow:

1: The Admin creates a new room name with a seating capacity.
2: Admin can edit/change an existing room assigned to a course
   i. If the number of students are more than the seating capacity.
   ii. If the room doesn’t contain required equipment’s for the course.
3: The Admin can remove a room name with a seating capacity
   i. if a room doesn’t exist in the University

Postcondition: “A room name and capacity has been created”
Alternative path: If the Admin selects cancel in step 1, the use case stops with postcondition “the Admin selected cancel”

Use Case “create, edit, remove course”
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R4, R5, R6
Precondition: The course hasn’t been created yet, or a course has been created and needs to be changed or removed.
Typical flow:

1: The Admin creates a course name with an enrollment limit.
2: Admin can edit/change a name and limit assigned to a course
   i. If the number of students requesting to enroll are more than the limit set.
   ii. If the course name is incorrectly saved in the database.
3: The Admin can remove a course name with its enrollment limit
   i. if the course isn’t offered anymore in the University.

Postcondition: “A course name and enrollment limit has been created”
Alternative path: If the Admin selects cancel in step 1, the use case stops with postcondition “the Admin selected cancel”

Use Case “create sections”
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R7
Precondition: The sections has not been created.
Typical flow:

1: click the create/new button
2: pops up a text-field window
   i. Fill fields and click OK, and create section
   ii. click Cancel, and no section created.

Postcondition: “a section is created”

Use Case “create Subsections”
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R8

Precondition: The sections has not been created and same section has not created
Typical flow:
1: select a section and select new Subsection
2: pops up a text-field window
i. Fill fields and click OK,
ii. click Cancel, and no Subsection created.
Postcondition: “a Subsection is created”

Use Case “remove individual or complete Subsections”
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R9, R10
Precondition: The Subsections has been created.
Typical flow:
1: select an individual Subsection and click the delete button
2: pops up a confirmation box
i. click OK, and individual subsection deleted
ii. click Cancel, and individual subsection is kept
3: select Subsection and click the delete button
4: pops up a confirmation box
i. click OK, and subsection deleted
ii. click Cancel, and subsection is kept
Postcondition: “a Subsection is removed”

Use Case “create students leaf or student composite”
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R11, R12
Precondition: The same leaf or composite has not been created.
Typical flow:
1: click the create/new button
2: pops up a text-field window
i. Fill fields and click OK,
ii. click Cancel, and no section created.
Postcondition: “The leaf or composite is created”

Use Case “create variable length time interval”
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R13
Precondition: The same time slots has not been created.
Typical flow:
1: click the new time slot button
2: add new time intervals and click OK.

Postcondition: “Variable time slot is created”
Use Case “create instructor”
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R14

Precondition: The same instructor hasn't been created.
Typical flow:

1: click the new user button

2: add instructor name, ID, list of courses.

i. Click OK to save or
ii. Cancel to exit.

Postcondition: "instructor with associated ID is created"

Use Case "associate subject to instructor"
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R15
Precondition: The association hasn't been created.
Typical flow:

1: get list of subjects from the database

2: add list to an instructor

i. Click confirm to save, or
ii. Cancel to exit.

Postcondition: "association between course and instructor is created"

Use Case "view schedule"
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R16
Precondition: View hasn't been generated and schedule is done
Typical flow:

1: get course name, instructor, section and time slot

2: Display schedule

Postcondition: "view schedule is created"

Use Case "Export schedule in PDF"
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R17
Precondition: "View is open"
Typical flow:

1: get course name, instructor, section and time slot

2: Display exporting navigation slider

3: Export schedule

i. If user selects the "Export to PDF" button

Postcondition: "schedule has been exported to specific path in PDF file format"

Use Case "edit schedule"
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R18
Precondition: schedule been created.
Typical flow:

1: Admin can edit schedule

i. if schedule doesn’t reflect user(instructor) requirements.
ii. If any of section, time slot, instructor etc. given incorrectly.

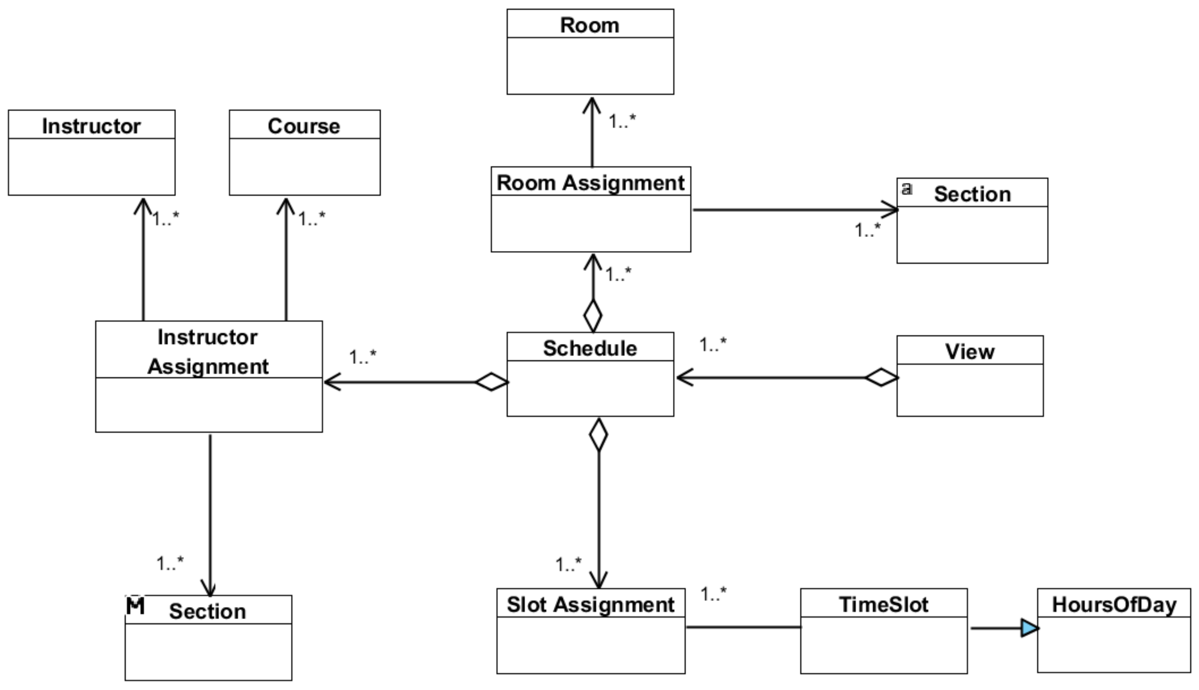
Postcondition: “schedule is edited”

Use Case “view unused time interval”
Subject: Course Scheduler
Actors involved: The Admin, the Scheduler
Requirements addressed: R19
Precondition: time intervals created.
Typical flow:

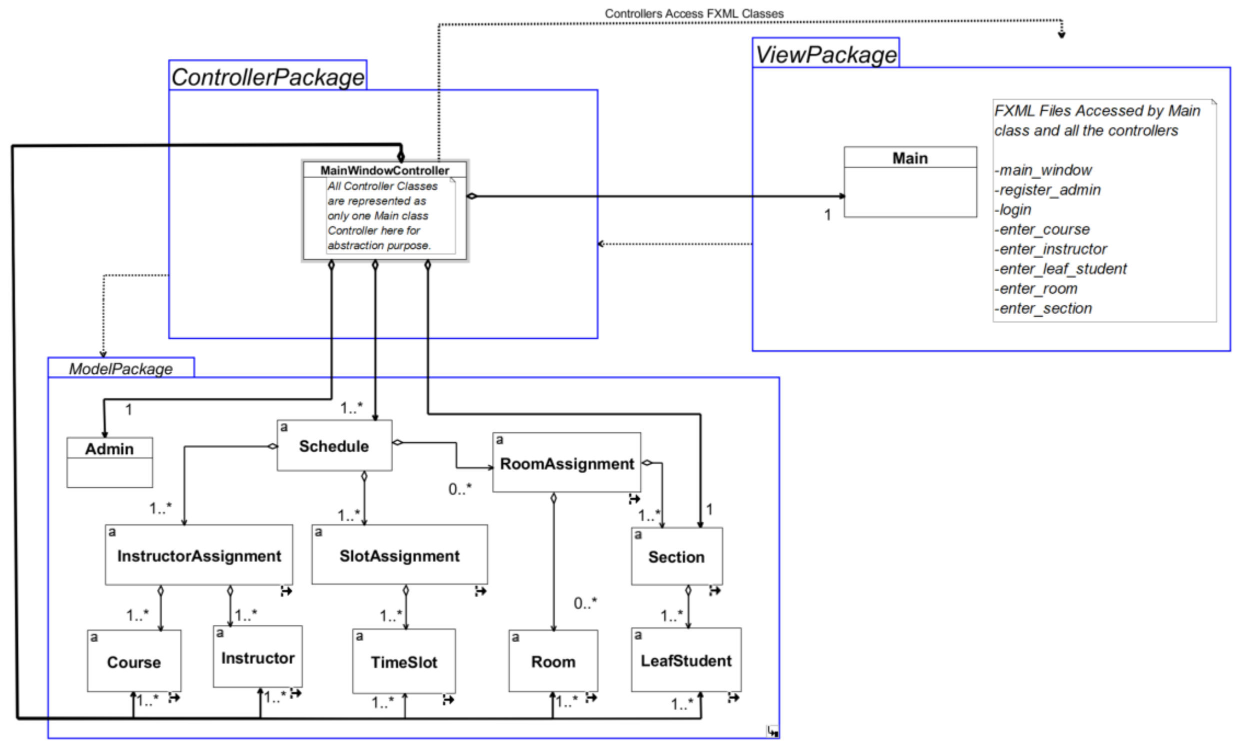
1: Admin can view unused time slots
   i. if time slot isn’t assigned to any course yet.
   ii. If time slot isn’t assigned to room yet.

Postcondition: “display unused time intervals”

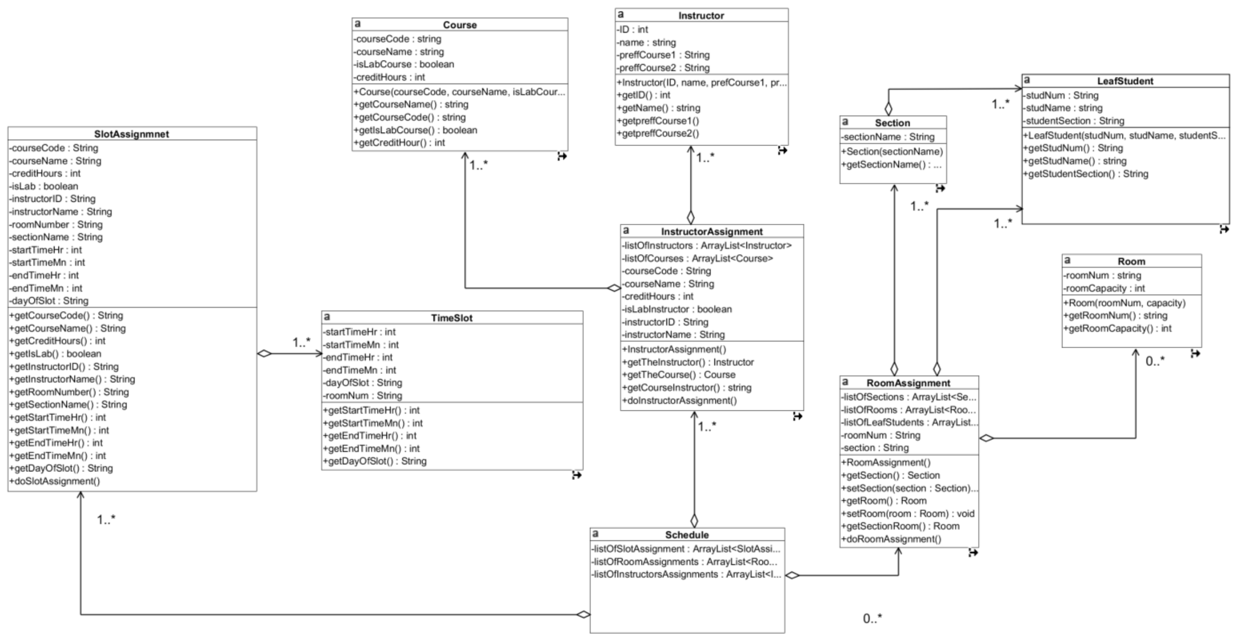
Simplified Domain Diagram



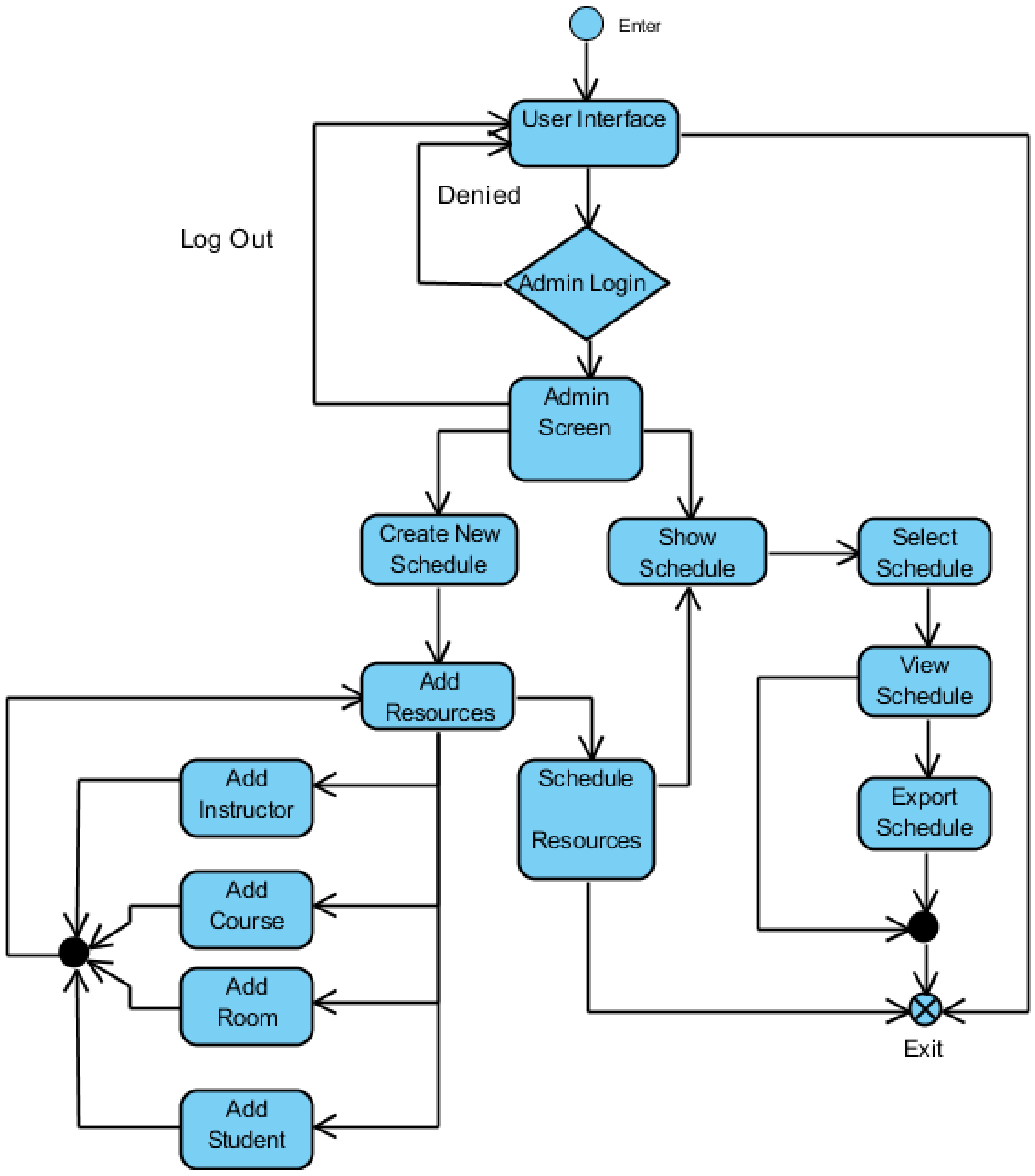
Package Diagram



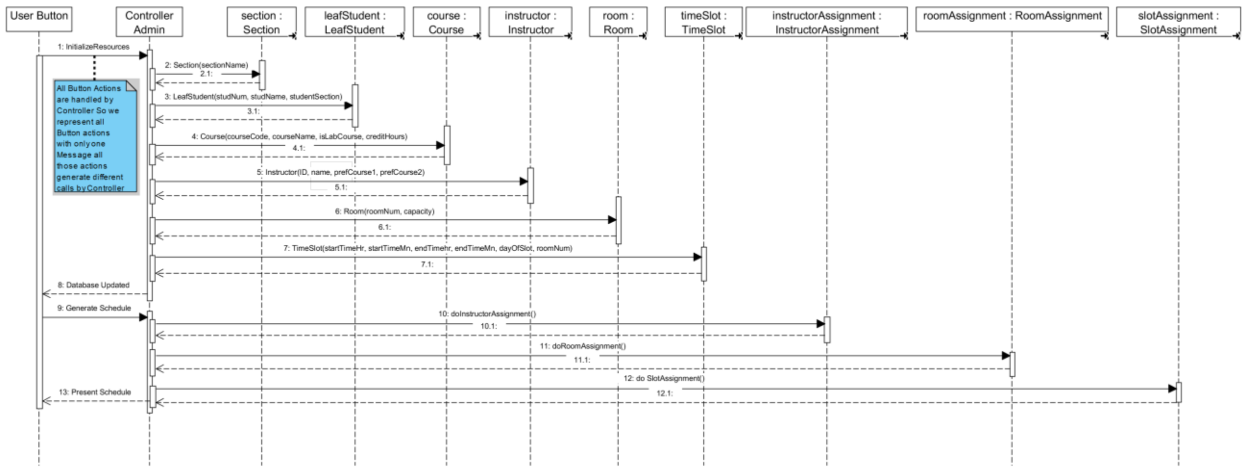
Class Diagram



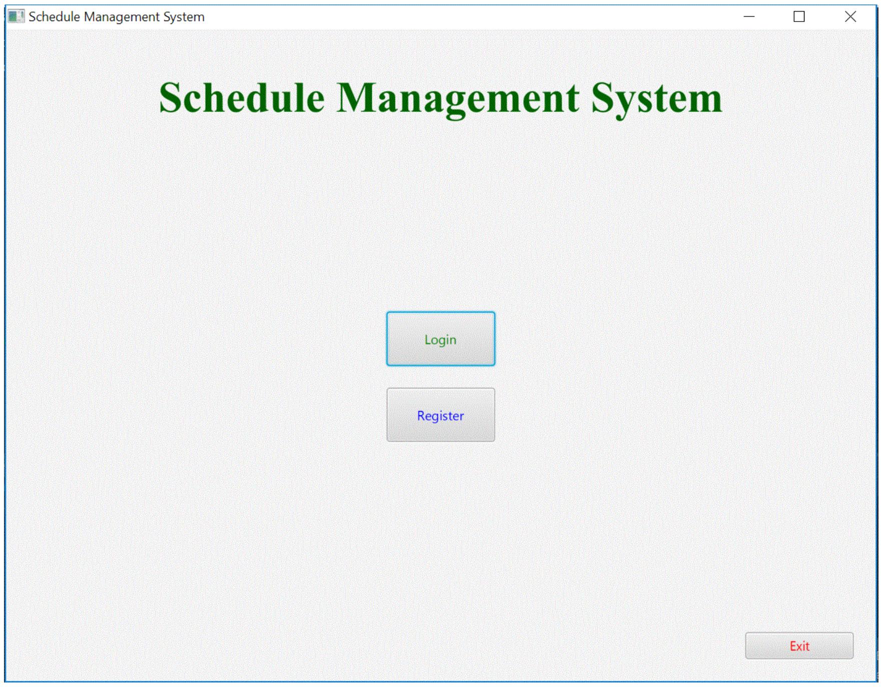
Activity Diagram



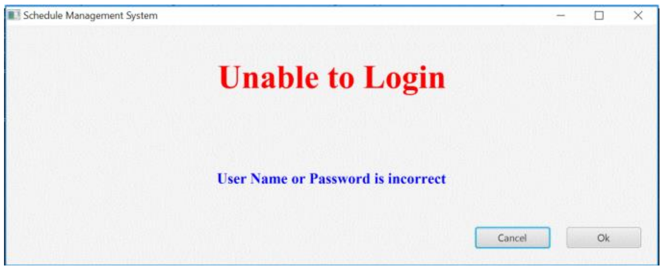
Sequence Diagram


User Button
Controller Admin
section : Section
leafStudent : LeafStudent
course : Course
instructor : Instructor
room : Room
timeSlot : TimeSlot
instructorAssignment : InstructorAssignment
roomAssignment : RoomAssignment
slotAssignment : SlotAssignment
1: InitializeResources
All Button Actions are handled by Controller So we represent all Button actions with only one Message all those actions generate different calls by Controller
2: Section(sectionName)
2.1:
3: LeafStudent(studNum, studName, studentSection)
3.1:
4: Course(courseCode, courseName, isLabCourse, creditHours)
4.1:
5: Instructor(ID, name, prefCourse1, prefCourse2)
5.1:
6: Room(roomNum, capacity)
6.1:
7: TimeSlot(startTimeHr, startTimeMn, endTimehr, endTimeMn, dayOfSlot, roomNum)
7.1:
8: Database Updated
9: Generate Schedule
10: doInstructorAssignment()
10.1:
11: doRoomAssignment()
11.1:
12: do SlotAssignment()
12.1:
13: Present Schedule

Startup Mockup

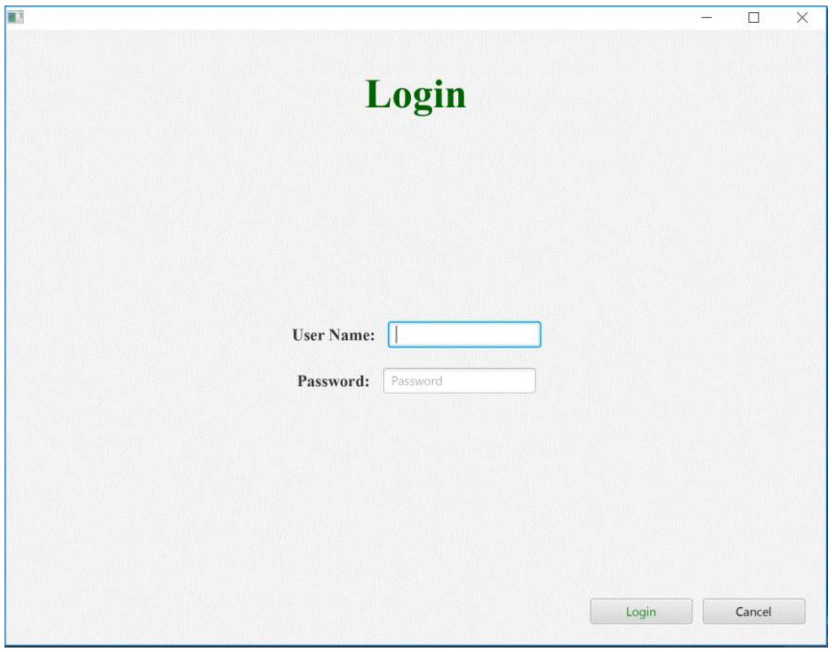

Schedule Management System

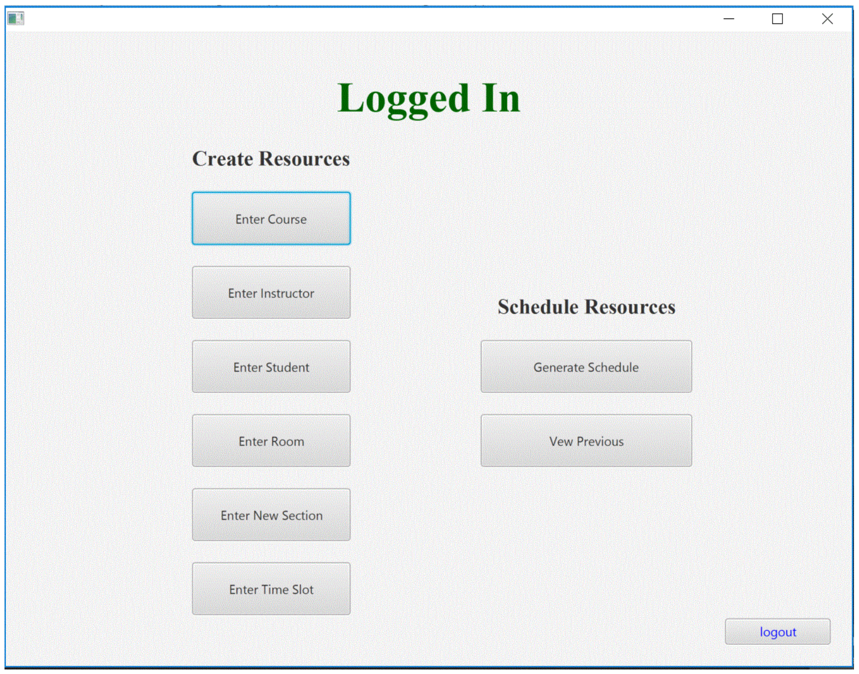
Schedule Management System

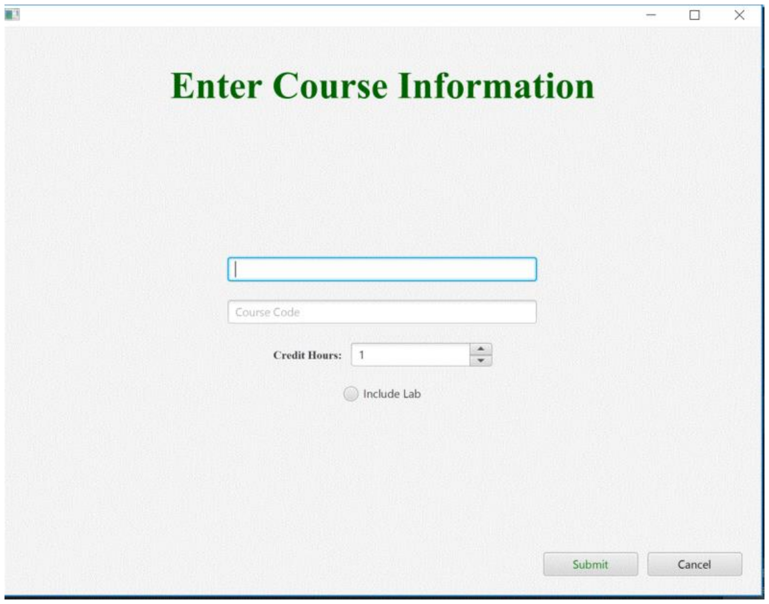

Schedule Management System

Unable to Login

User Name or Password is incorrect


Login
User Name:
Password:
Password
Login
Cancel


Logged In
Create Resources
Enter Course
Enter Instructor
Enter Student
Enter Room
Enter New Section
Enter Time Slot
Schedule Resources
Generate Schedule
Vew Previous
logout



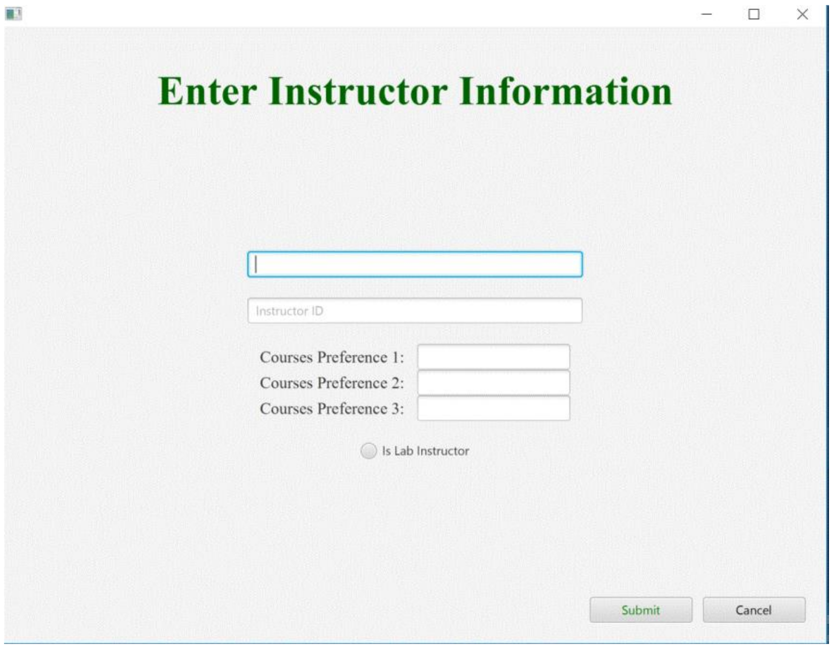
Enter Course Information

Course Code

Credit Hours: 1

Include Lab

Submit Cancel



Enter Instructor Information

Instructor ID

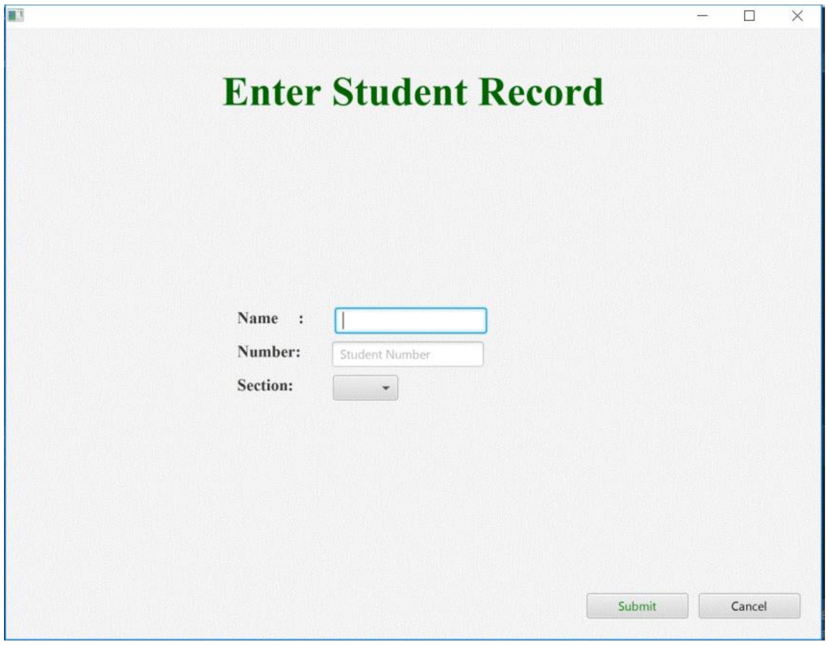
Courses Preference 1:

Courses Preference 2:

Courses Preference 3:

Is Lab Instructor

Submit Cancel



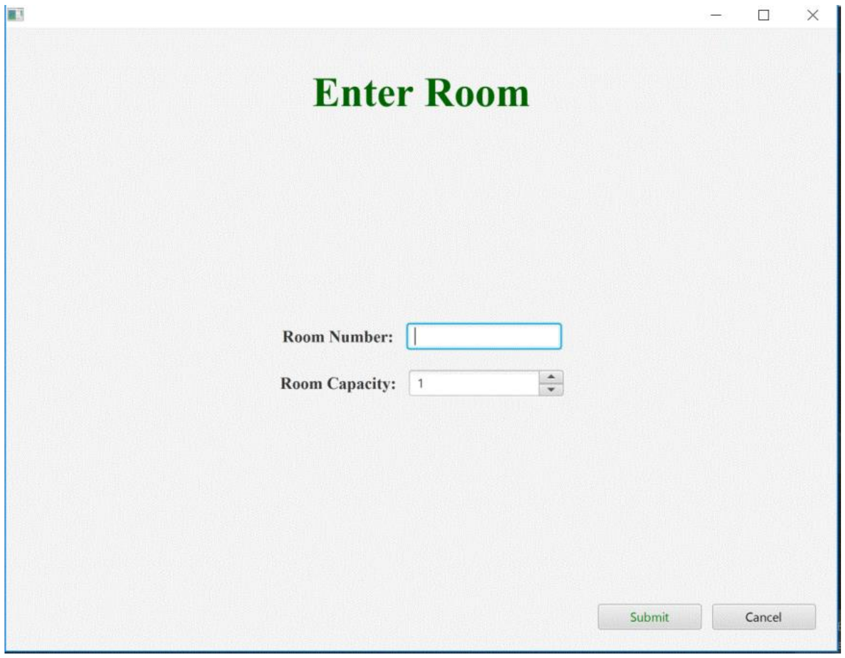
Enter Student Record

Name :

Number: Student Number

Section:

Submit Cancel

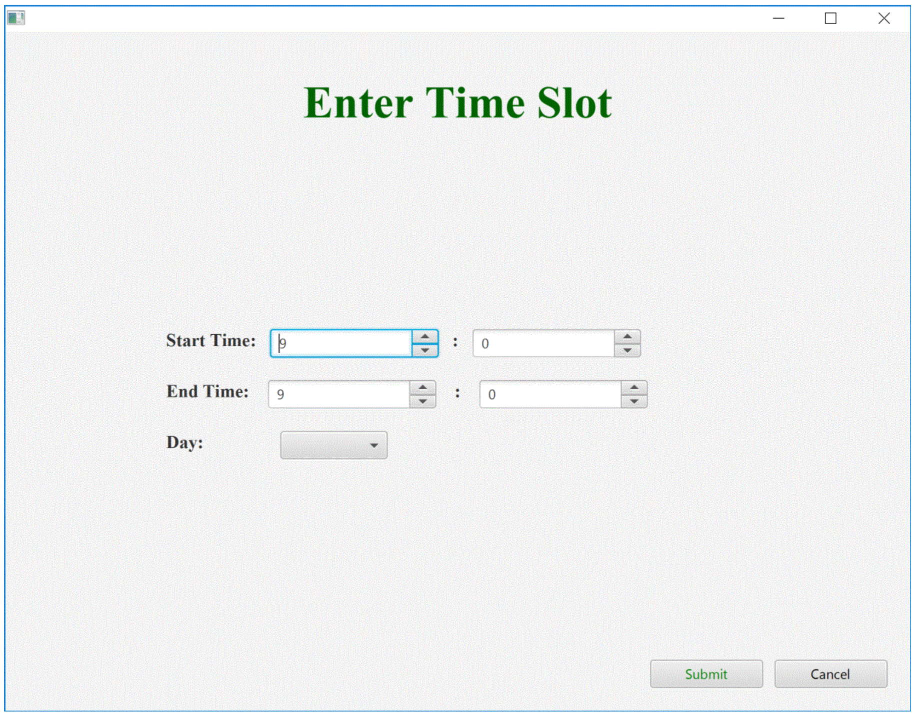


Enter Room

Room Number:

Room Capacity: 1

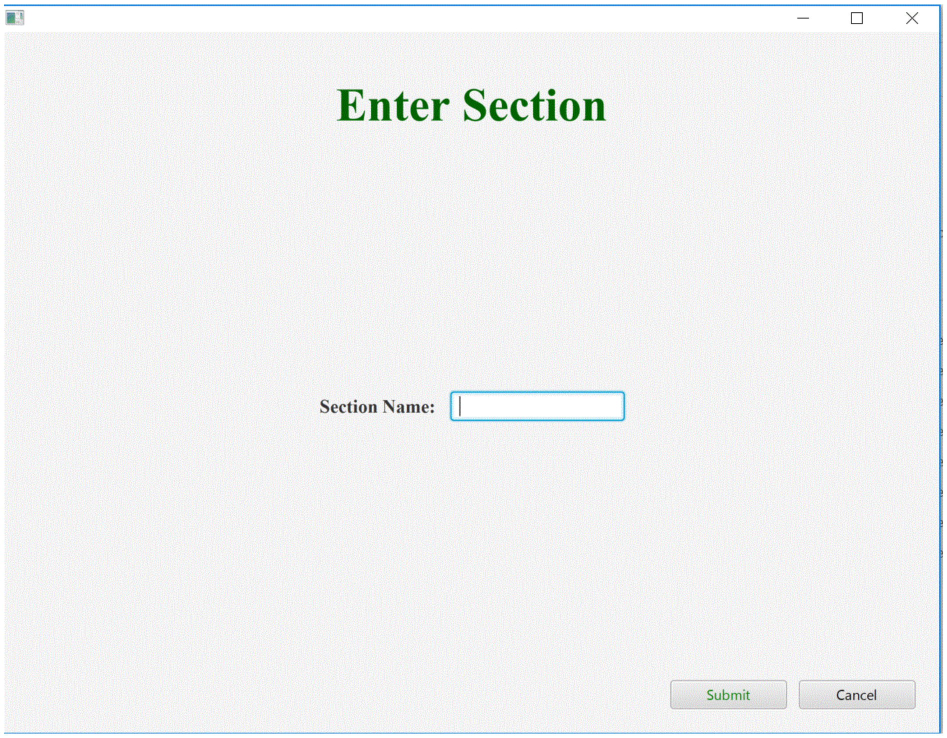
Submit Cancel


Enter Time Slot

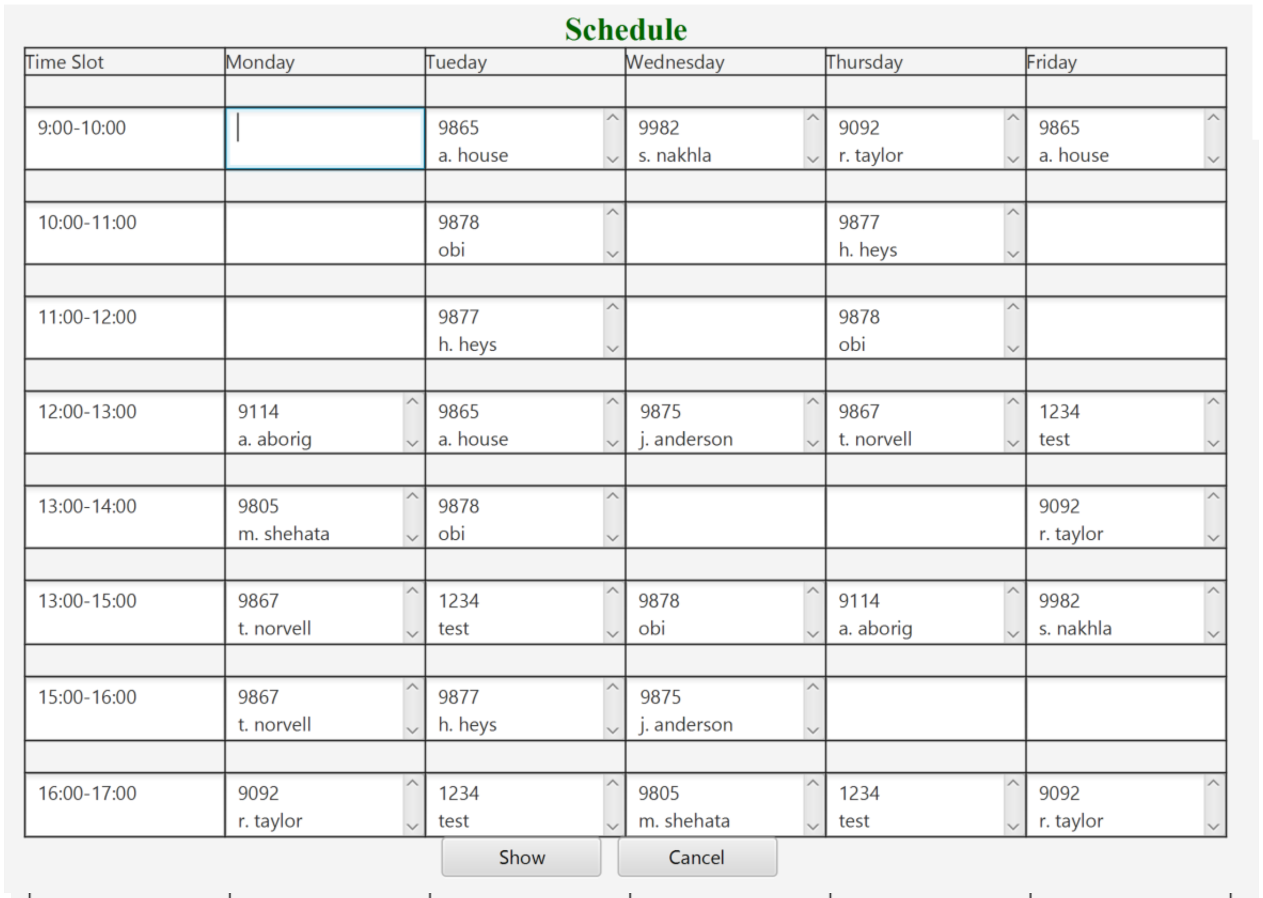
Start Time: 9 : 0

End Time: 9 : 0

Day:

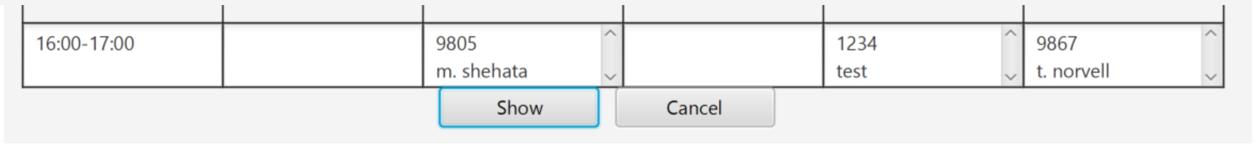

Enter Section

Section Name:

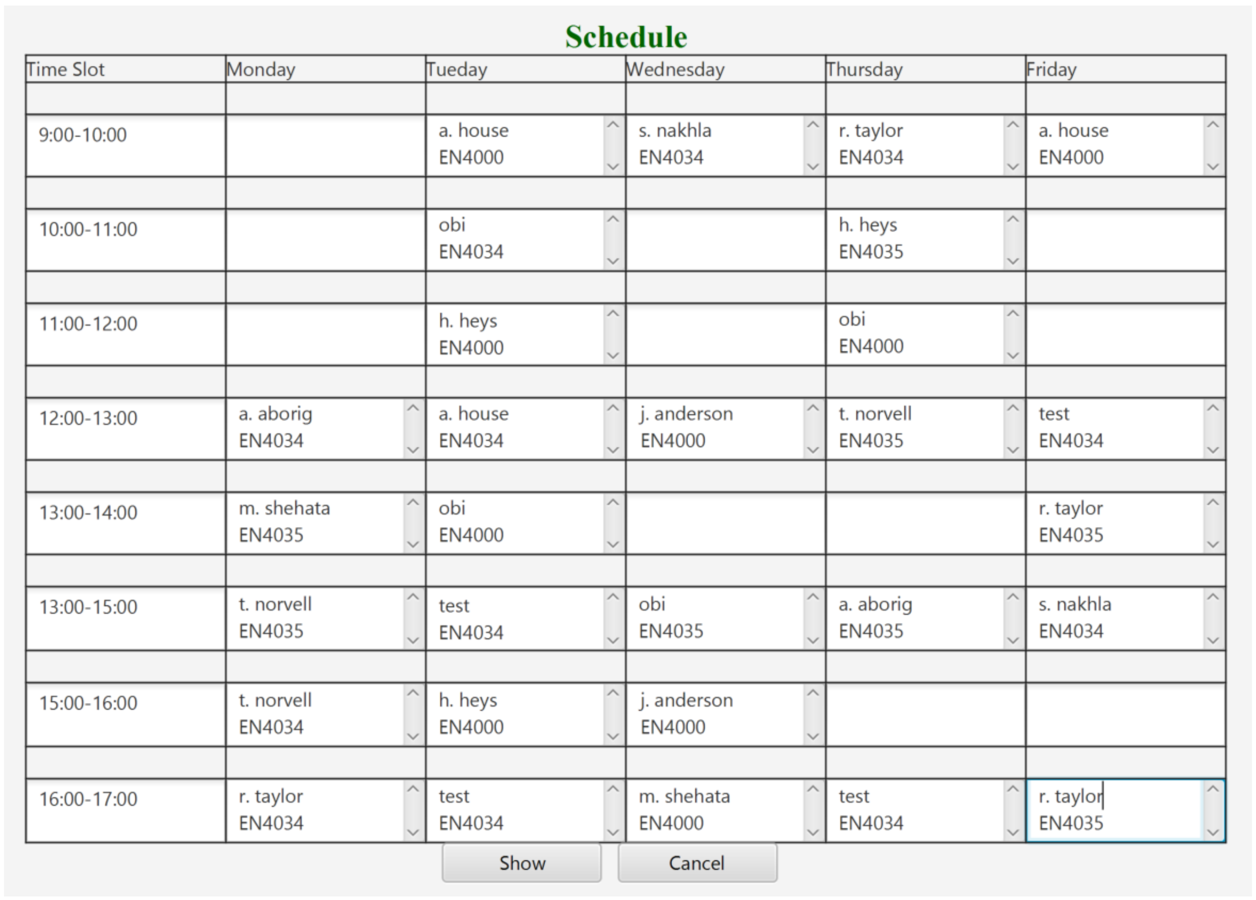


Schedule

Time Slot	Monday	Tueday	Wednesday	Thursday	Friday
9:00-10:00		9865 a. house	9982 s. nakhla	9092 r. taylor	9865 a. house
10:00-11:00		9878 obi		9877 h. heys	
11:00-12:00		9877 h. heys		9878 obi	
12:00-13:00	9114 a. aborig	9865 a. house	9875 j. anderson	9867 t. norvell	1234 test
13:00-14:00	9805 m. shehata	9878 obi			9092 r. taylor
13:00-15:00	9867 t. norvell	1234 test	9878 obi	9114 a. aborig	9982 s. nakhla
15:00-16:00	9867 t. norvell	9877 h. heys	9875 j. anderson		
16:00-17:00	9092 r. taylor	1234 test	9805 m. shehata	1234 test	9092 r. taylor



16:00-17:00		9805 m. shehata		1234 test	9867 t. norvell



Schedule

Time Slot	Monday	Tueday	Wednesday	Thursday	Friday
9:00-10:00		a. house EN4000	s. nakhla EN4034	r. taylor EN4034	a. house EN4000
10:00-11:00		obi EN4034		h. heys EN4035	
11:00-12:00		h. heys EN4000		obi EN4000	
12:00-13:00	a. aborig EN4034	a. house EN4034	j. anderson EN4000	t. norvell EN4035	test EN4034
13:00-14:00	m. shehata EN4035	obi EN4000			r. taylor EN4035
13:00-15:00	t. norvell EN4035	test EN4034	obi EN4035	a. aborig EN4035	s. nakhla EN4034
15:00-16:00	t. norvell EN4034	h. heys EN4000	j. anderson EN4000		
16:00-17:00	r. taylor EN4034	test EN4034	m. shehata EN4000	test EN4034	r. taylor EN4035

Conclusion:

We have meet with one of the main requirement of the project which was scheduling the resources. We were concerned to let the user input to schedule as less. Scheduling of resources is done the business logic of our application. Here is a review on how much we were on track and how much we a successful in our product. Some key points on our hard constraints are mentioned below,

We used 'InstructorAssignment' class to implement assignment of courses to instructor willing to teach those courses based on the preference of the instructor.

We used 'RoomAssignment' class to implement the assignment of rooms to sections but here we take help from another class which record the strength of section. Enter information of LeafStudent need to provide the Section name as well which will be recorded by SectionStrength class to compare the capacity of the rooms with the section strength in RoomAssignment class.

We used 'SlotAssignment' class which is doing the allotment of the time slots provided by the administrator during entering resources. Slots are assigned randomly and once a slot is assigned that slot will not be assigned again to any session.

R0: As an Administrator, I can create new schedule, open existing schedule from schedules in Database (text files).

Our Outcome: We are successful in generating new schedule which is generated randomly. But implementing the scheduling log is very close to implementation which needs only couple of hours effort. Database is successfully linked.

R1: As an Administrator, I can create room with name and seating capacity

Our Outcome: Successful

R2: As an Administrator, I can remove the room

Our Outcome: Method is implemented to delete admin from database, but GUI form is not implemented yet. Little effort is required to do complete this functionality.

R3: As an Administrator, I can edit/change the room name and capacity

Our Outcome: Update queries are not implemented yet

R4: As an Administrator, I can create course with name and course enrollment limit

Our Outcome: Successfully created course but enrollment limit is not considered yet due to section strength logic usage on room assignment

R5: As an Administrator, I can remove course

Method is implemented to delete course from database, but GUI form is not implemented yet. Little effort is required to do complete this functionality.

R6: As an Administrator, I can edit the course name and enrollment limit

Update queries are not implemented on database yet

R7: As an Administrator, I can create Sections

Successfully created

R8: As an Administrator, I can create Subsections

Sections and subsections are considered only at one level (Sections)

R9: As an Administrator, I can remove individual subsections

Can remove complete table from database but GUI functionality is not given yet

R10: As an Administrator, I can remove complete Subsection

Can remove complete table from database but GUI functionality is not given yet

R11: As an Administrator, I can create set of students leaf
Successfully completed
R12: As an Administrator, I can create set of students composite

Successfully completed (Populating section list into student record entry)

R13: As an Administrator, I can create variable length time intervals

Successfully achieved

R14: As an Administrator, I can create instructor

Successfully achieved

R15: As an Administrator, I can associate subject to instructor

Successfully achieved

R16: As an Administrator, I can view the schedule from (section, room#, Instructor and time interval) point of view

Successfully achieved

R17: As an Administrator, I can export finished schedule to a printable format (e.g. pdf)

Not implemented yet. Need little effort to complete

R18: As an Administrator, I can edit the schedule

Update queries are not implemented yet

R19: As an Administrator, I can view the unused time intervals

Successfully achieved

General Remarks: Project is meeting some of the basic requirements more functionality can be added to application easily with little effort. We achieved the conflict resolution by assigning resources and then at rum time removing those assigned resources from the available resource list. We can implement conflict resolution with help of dialogs to give more assistance to admin. If we will be given extra time to work on the project, then we will add extra functionality by creating schedule log into database so that different schedules can be accessed later by admin. Database update implementation with GUI interaction will be our second target.