Software Design Specification

for

AVAILSYS

Prepared by

- IIT2019204
MITTA LEKHANA REDDY
- IIT2019208
DHANUSH VASA
- IIT2019234
PRAVALLIKA KODI
- IIB2019030
KANDAGATLA MEGHANA SANTHOSHI

IIIT Allahabad

30-01-2021

1.Introduction:

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, state diagrams and activity diagrams.

1.1 **Purpose of this Document:**

This document will define the design of the one runway simulator. It contains specific information about the expected input, output, classes, and functions. The interaction between the classes to meet the desired requirements are outlined in detailed figures at the end of the document.

1.2 Scope of the development project:

Here we will be describing what features are in the scope of this project and what are not in the scope of the software to be developed.

In Scope:

a. Access for the faculty and admin to book lab and resources for lab
b. Faculty and admin have access to cancel a booked lab by themselves.
c. Faculty and Admin have the ability to edit sections and students details.
d. Send notification on the status of booking and cancellation of labs to students

Out of Scope:

1.3 **Definitions, acronyms and abbreviations:**

IEEE: Institute of Electrical and Electronics

1.4 **References:**

1. IEEE SDS Template

1.5 Overview of Document:

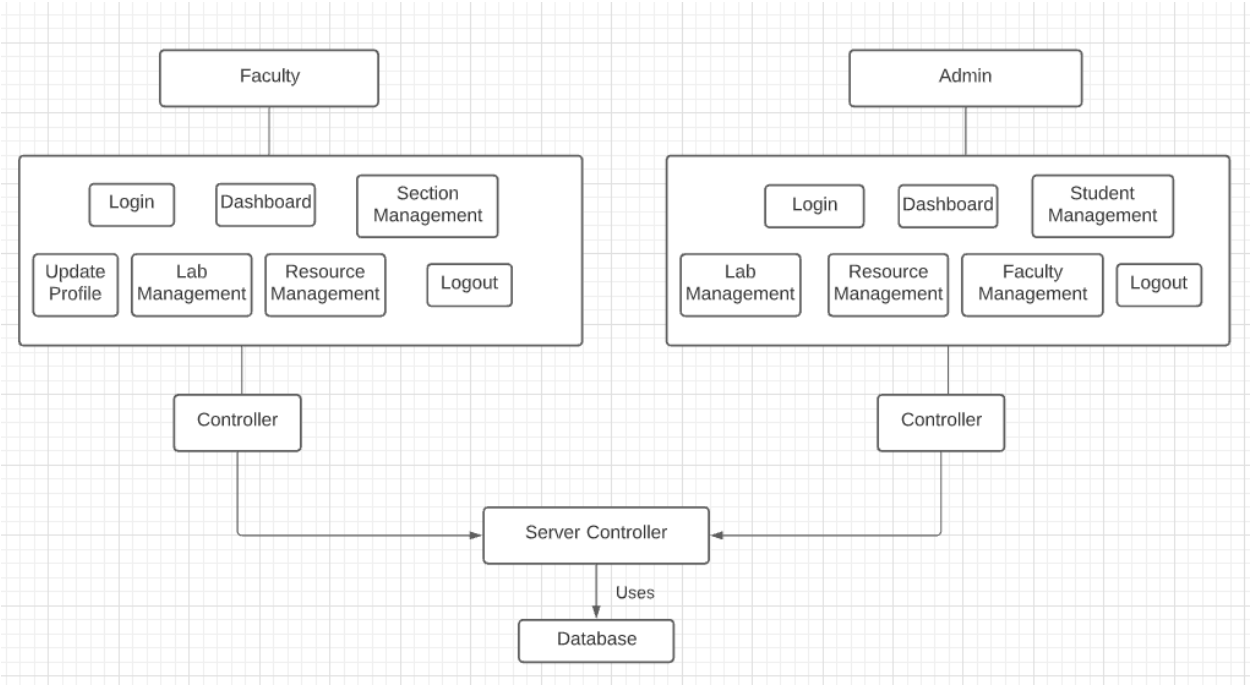
This SDS is divided into ------- sections with various sub-sections. The sections of the Software Design Document are :

1. **Introduction :** Describes about the document, purpose, scope of development project definitions and abbreviations used in the document.
2. **Conceptual Architecture/Architecture Diagram :** describes the overview of components, modules, structure and relationships and user interface issues.
3. **Logical Architecture :** describes Logical Architecture Description and Components.
4. **Execution Architecture :** defines the runtime environment, processes, deployment view.
5. **Design Decisions and Trade-offs :** describes the decisions taken along with the reason as to why they were chosen over other alternatives.
6. **Pseudocode for components :** describes pseudocode, as the name indicates.
7. **Appendices :** describes subsidiary matter if any.

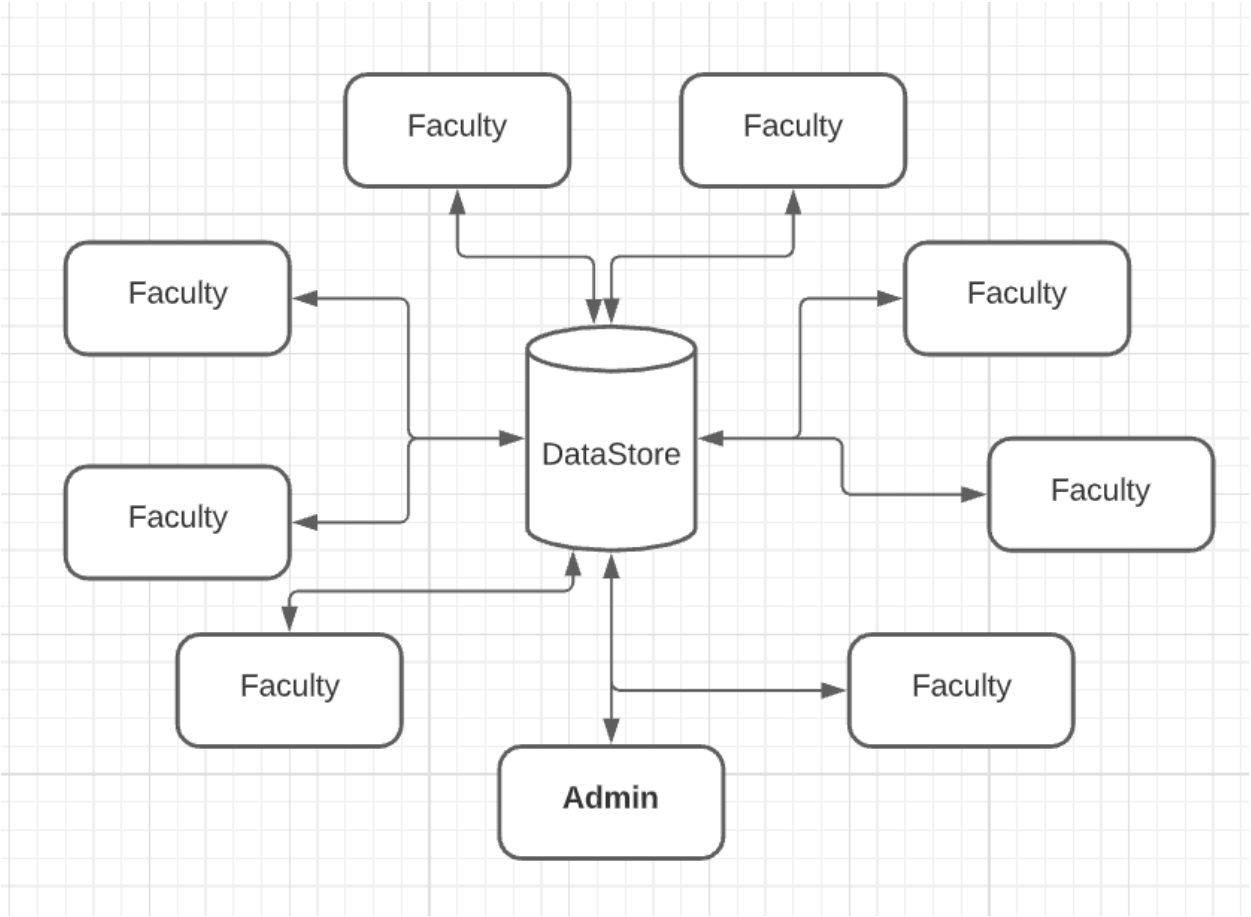
2. Conceptual Architecture

A. Architecture Diagram

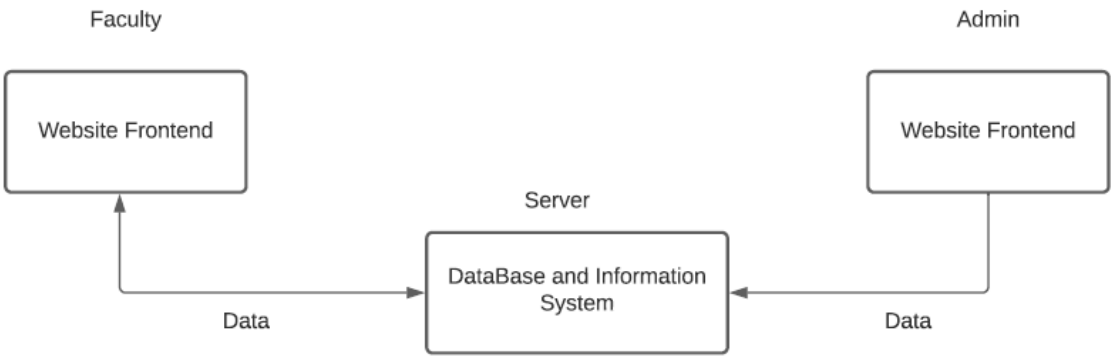
1: **High Level Design**



B. Architecture Diagram 2:

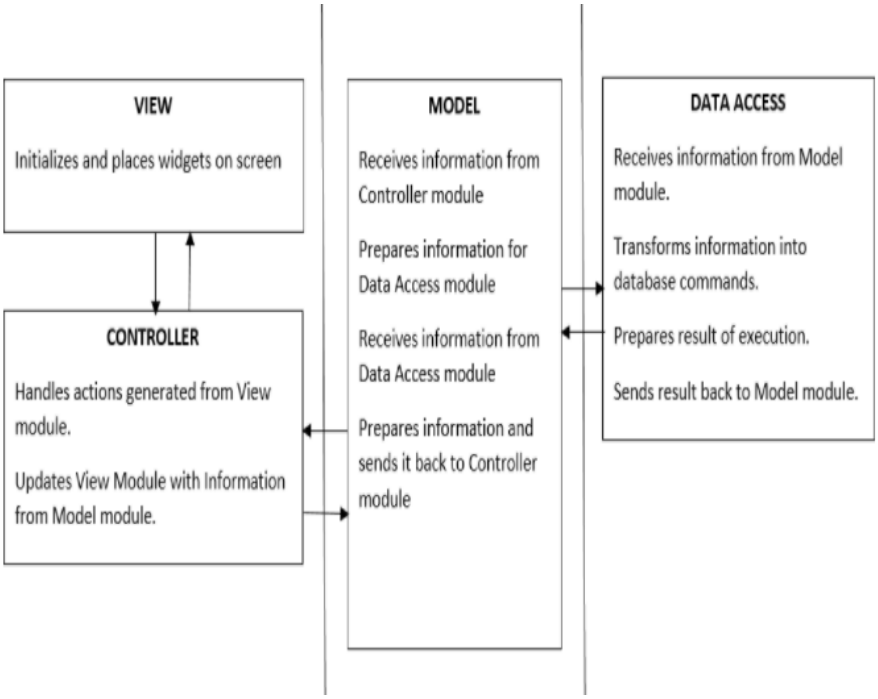


C. Architecture Diagram 3:



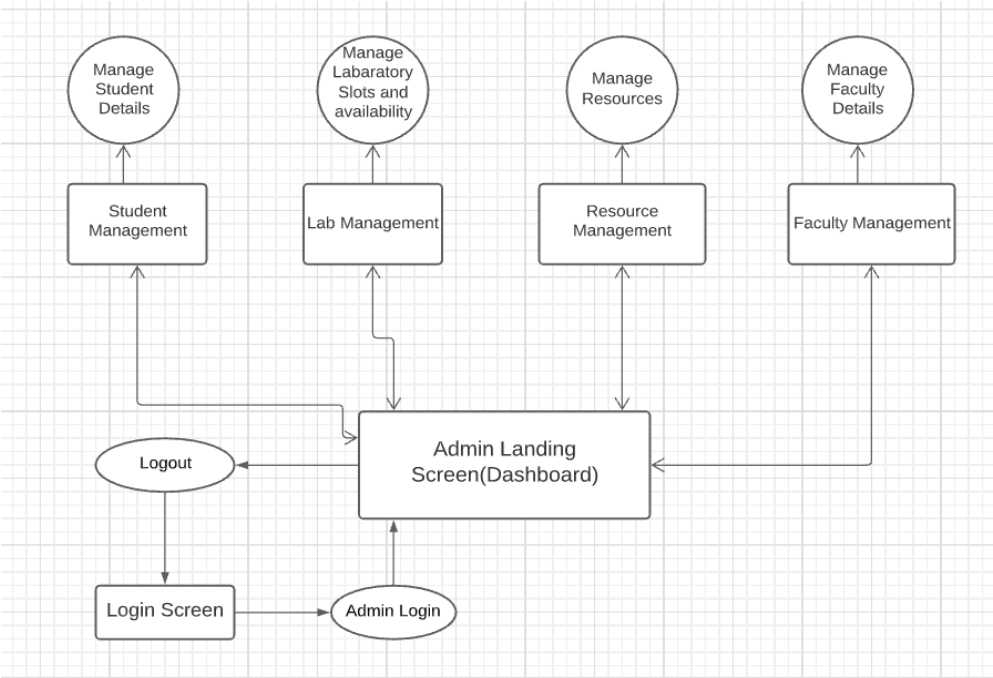
The above diagrams describe the basic architecture of data flow in the website between one user and another and conceptual architecture (understable to a Non-Technical person as well).

2.1 : Overview of Modules:

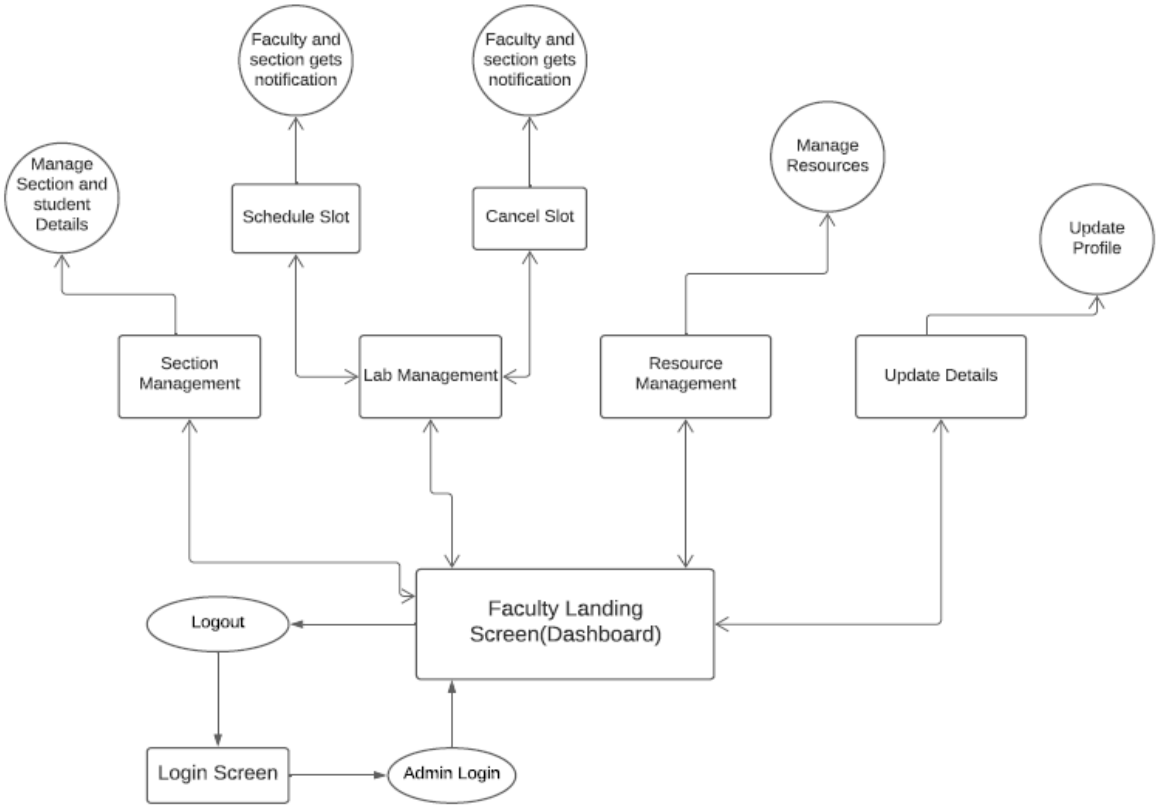


2.2 : Structure and Relationships :

2.2.1: Admin's Side:



2.2.2: Faculty Side:



NOTE:

- **BOXES:**They Represent separate and individual screens .
- **CIRCLES:**They represent activities which do not require separate screens.
- **ARROWS:** They Represent Navigation between the screens.

2.3: User Interface Issues:

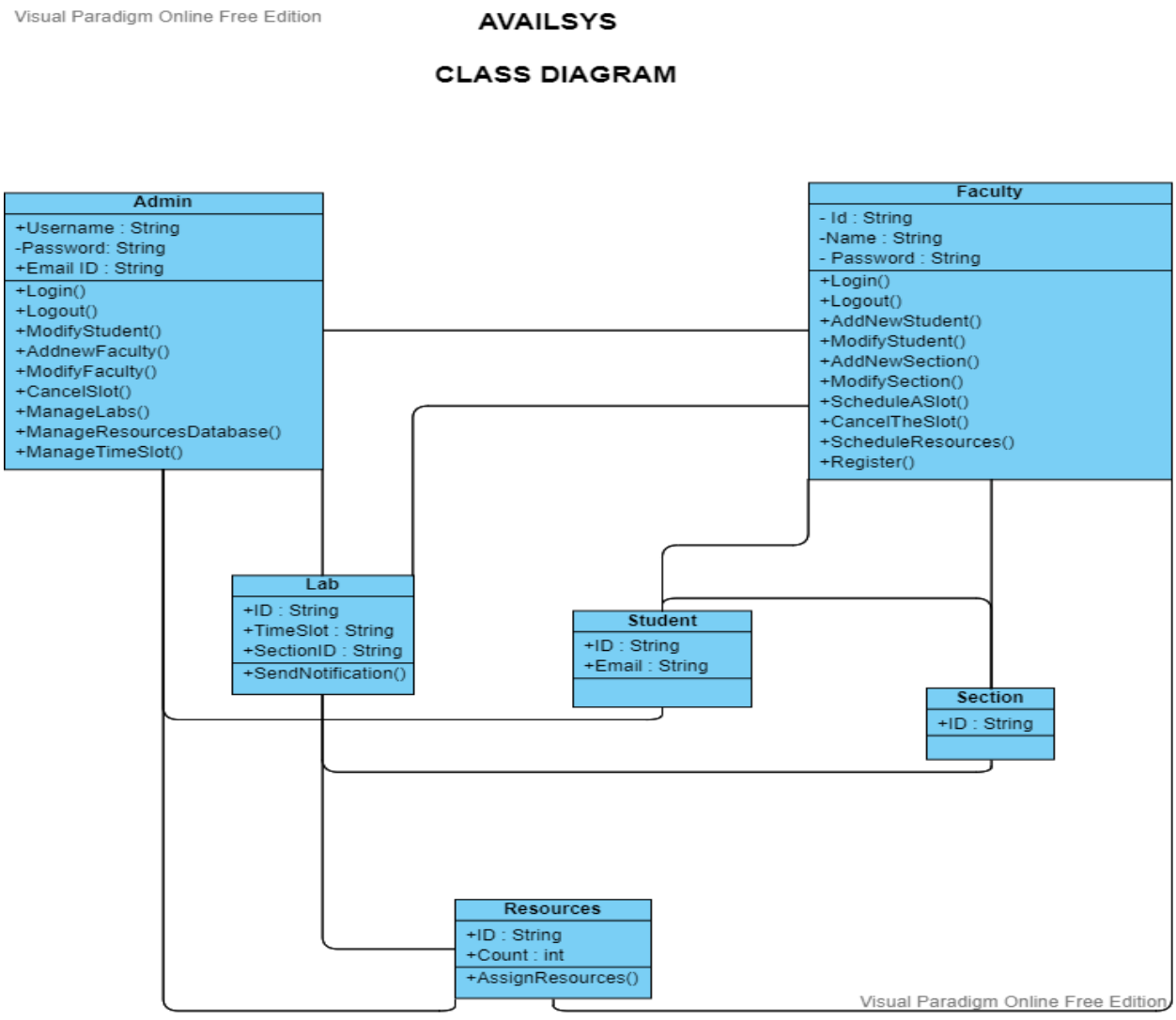
This section will address the issues faced by hypothetical users i.e;Faculty and Admin while using AvailSys .

- User-A is a 30 year-old male faculty who is trying to access Availsys and schedule labs.User-A is fairly comfortable with technologies and proficient with using common computer technologies.
    - Since he is comfortable with technology . AvailSys will use a common UI to comfort users.

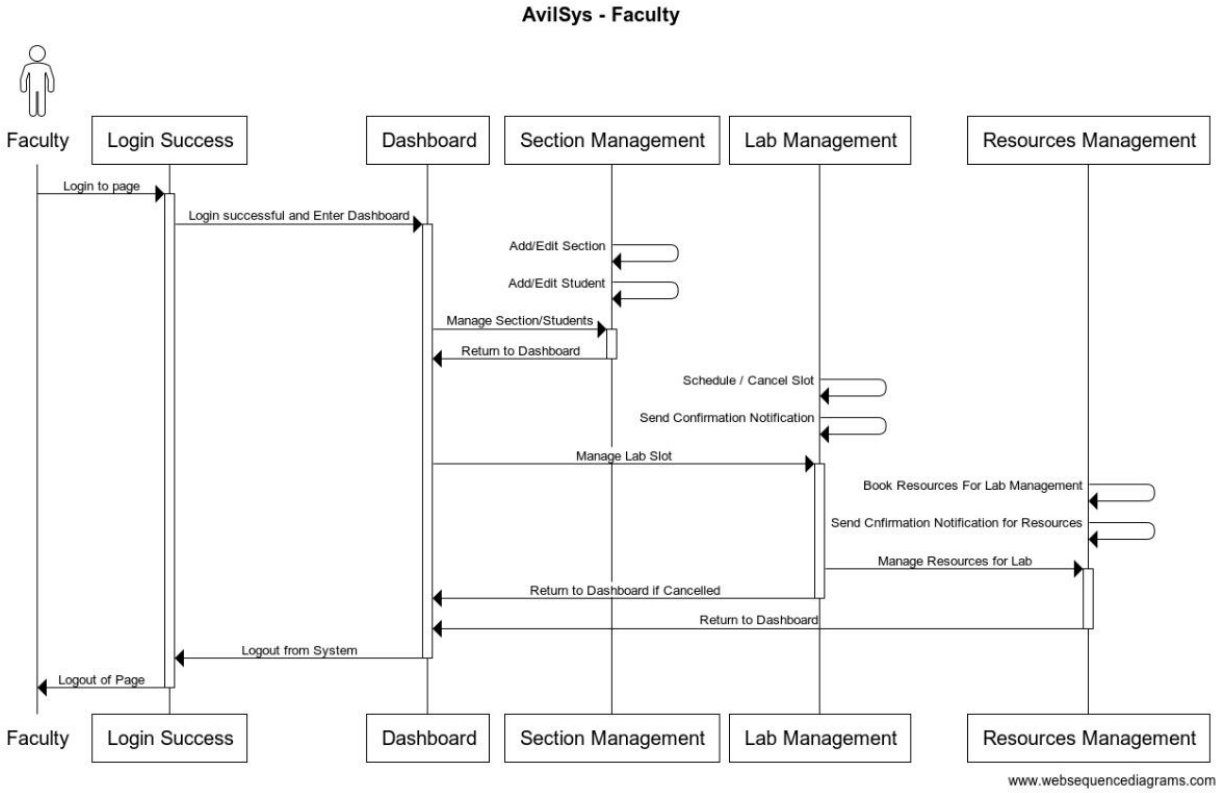
- User-B is a 40 year-old female staff from the administration department.She is familiar with the database management system from core.
    - Since she is comfortable with DBMS , she can operate and manage every detail of Availsys and we can use basic frontend to overcome this problem.

3. Logical Architecture (Class Diagram, Sequence Diagram, State Diagram, Activity Diagram)

Class Diagram:



Sequence Diagram: Faculty



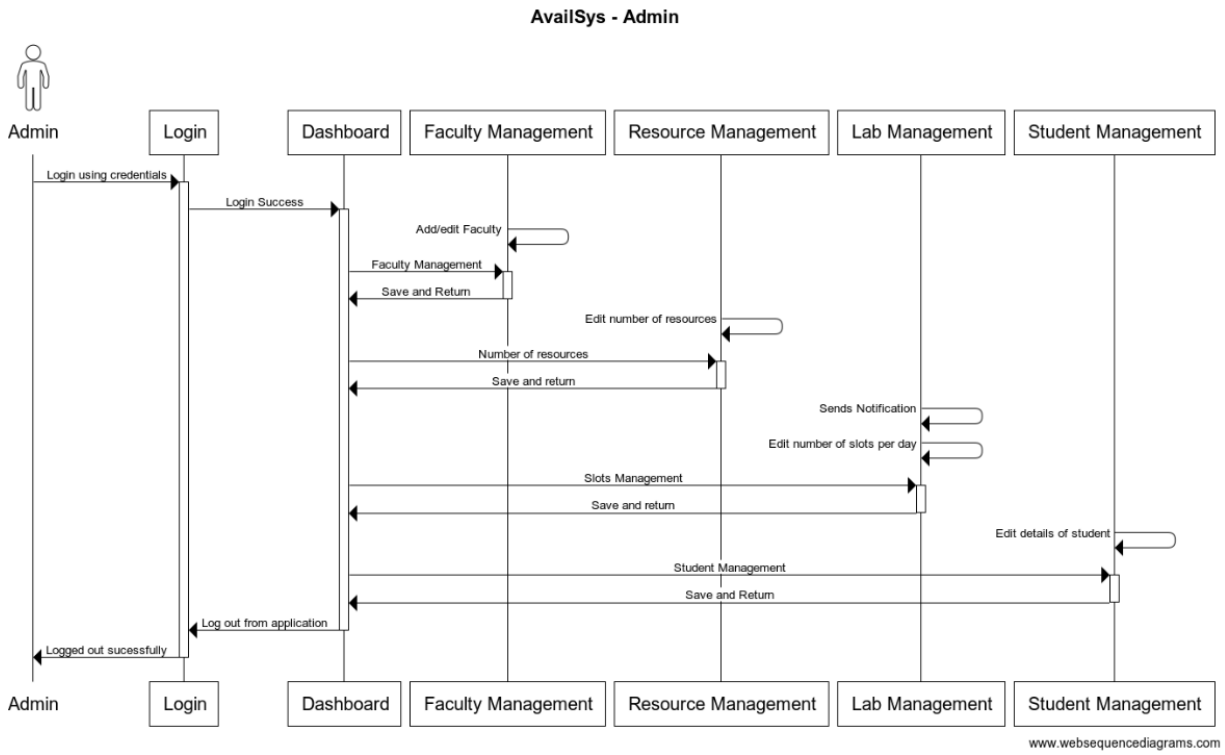
This is UML sequence diagram of laboratory reservation system with respect to Faculty.

- The instance of class objects involved in this diagram are :
    1. Login success
    2. Dashboard
    3. Section Management
    4. Lab Management
    5. Resources Management

- Here Faculty will be able to login into the gateway and enter the dashboard.
- From the Dashboard, Faculty can manage all operations on Section, Resource and Lab.
- All pages such as lab,student and Resources are safe and secure and Faculty can't access them without giving proper verification .

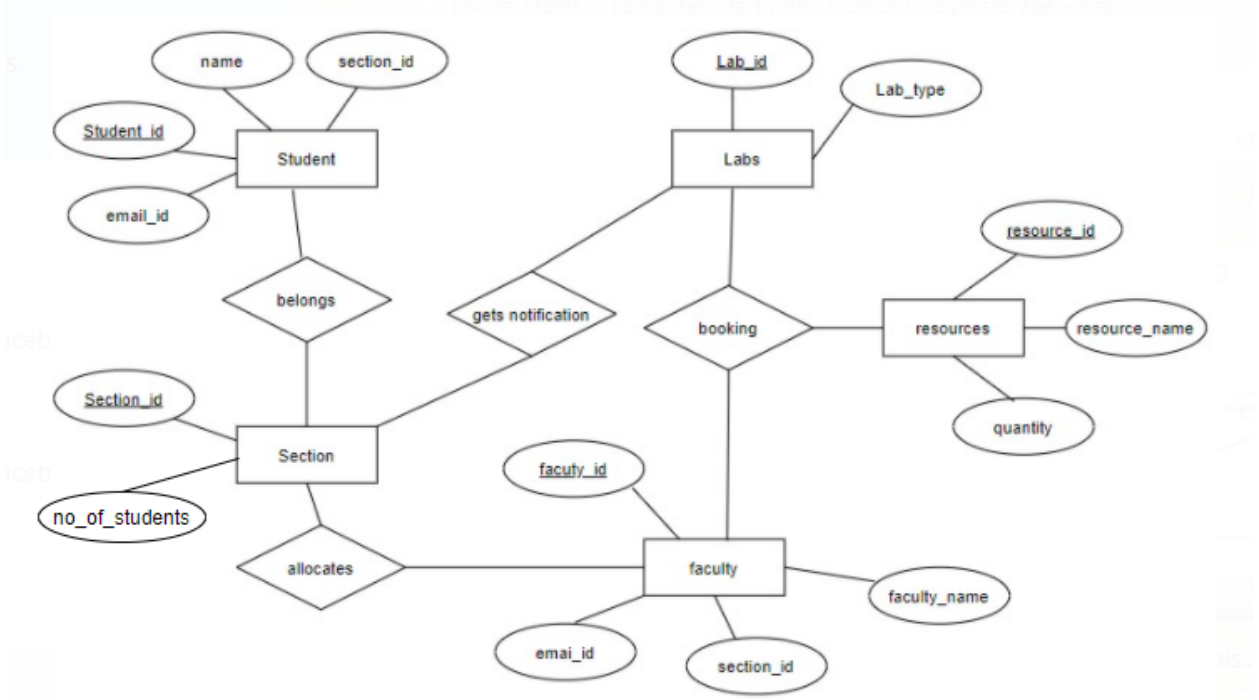
Sequence Diagram: Admin

This is UML sequence diagram of laboratory reservation system with respect to Admin.



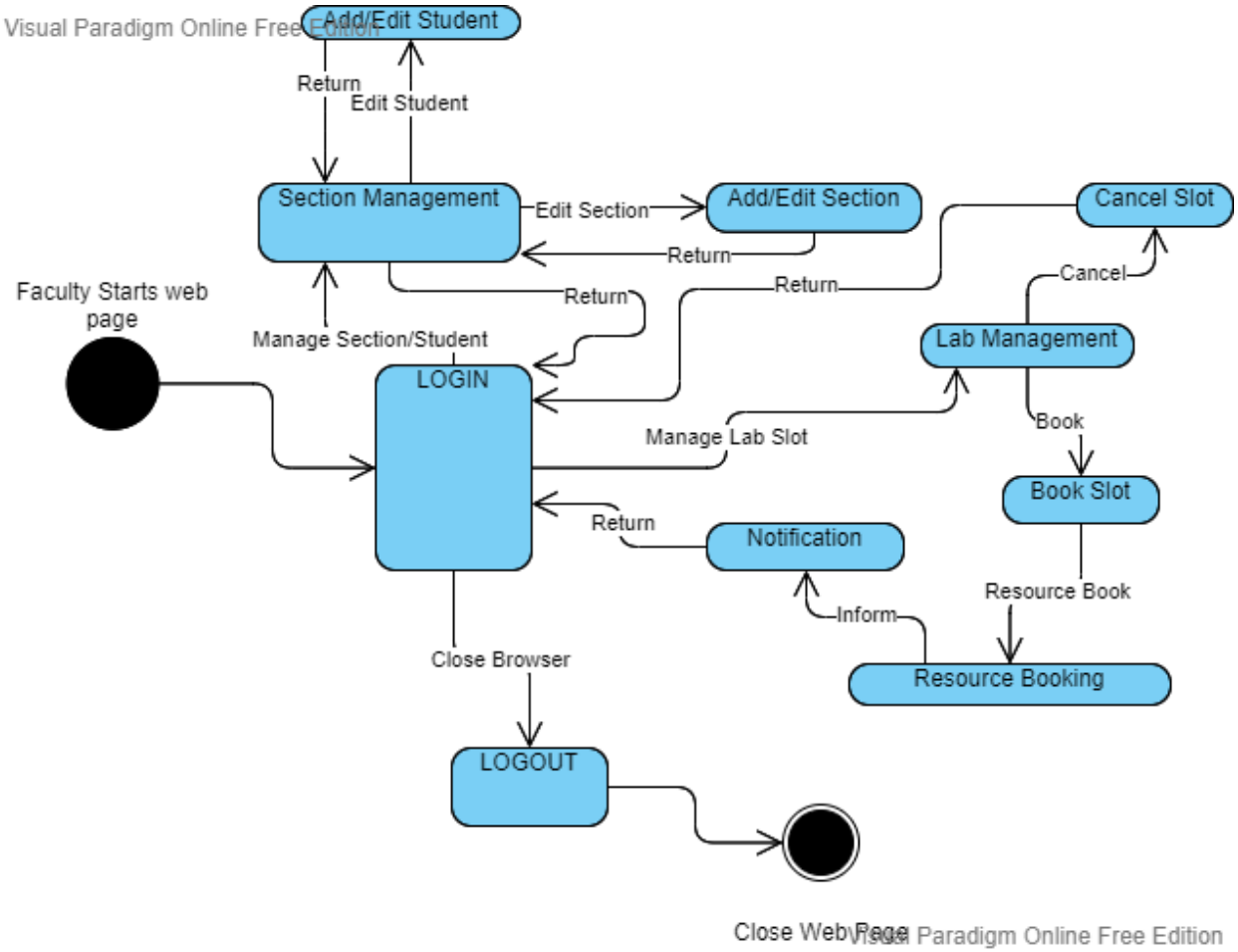
- The instance of class objects involved in this diagram are :
    1. Login
    2. Dashboard
    3. Faculty Management
    4. Resource Management
    5. Lab Management
    6. Student Management
- Here admin will be able to login into the gateway and enter the dashboard.
- From Dashboard Admin can manage all operations on Faculty , Resource , Lab and student.
- All pages such as lab,student and faculty are safe and secure and Admin can't access them without giving proper verification .

ER Diagram:



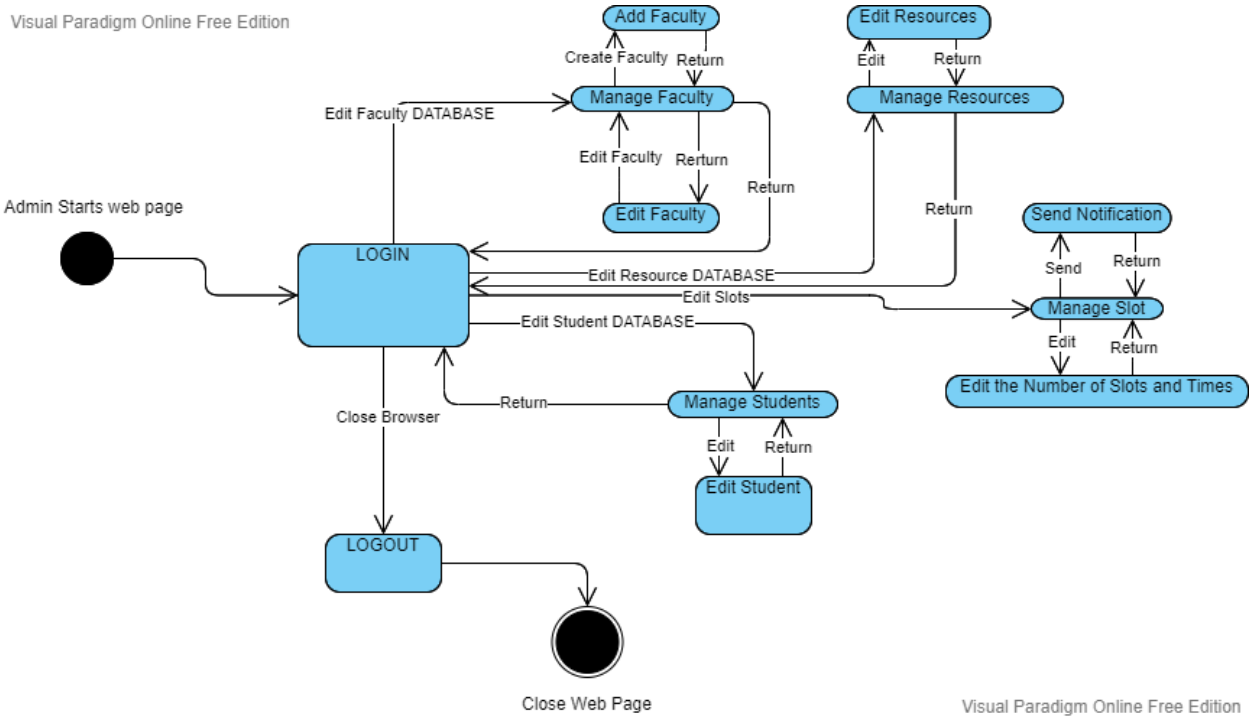
- Basically there are totally 5 entities - Student, Section, Lab, Resource and faculty
- Entity Student has attributes as section_id, email_id and name and has student_id as primary key.
- Entity Section has attributes no_of_students and has Section_id as primary key.
- Entity Lab has attributes Lab_type and has Lab_id as primary key.
- Entity Resource has attributes resource_name, Quantity and has resource_id as primary key.
- Entity Faculty has attributes faculty_name, section_id, email_id and has Faculty_id as primary key.
- Student and Section has **“belongs”** relationships.
- Section and Lab has a **“gets notified**” relationship.
- Section and faculty have **“allocates”** relationships.
- There is a ternary relationship **“booking”** between lab, faculty and resource.

State Diagram: Faculty



The state diagram for Faculty firstly he/she will login with the required credentials, now he/she can manage the slots(book the slots or cancel the slots).If the faculty books the slot, then the respective section gets notified and along with it faculty books the required resources.Faculty can also manage the sections and students(add or edit section and students). Whenever the faculty logouts, the webpage will be closed.

State Diagram: Admin



The state diagram for the faculty firstly he/she will login with the required credentials, now he/she can manage the slots(book the slots or cancel the slots).If the faculty books the slot, then the respective section gets notified and along with it faculty books the required resources. Faculty can also manage the sections and students(add or edit section and students). Whenever the faculty logouts, the webpage will be closed

3.2 Class Diagram Description:

3.2.1 Class name: Admin

Description: This class enables the admin to enter into the subsystem (Landing page) after authenticating the entered credentials.

Method 1: Login()
Input: view,email ,password
Output: Admin landing page of login successful
Method Description:
This method basically takes input as email and password from the user and checks whether it is authorized login or not and if the result is successful it leads to another activity page.

Method 2: Logout()
Input: logout option.
Output: Admin goes back to the login page.
Method Description:
This method basically takes the user back to the login page.

Method 3: ModifyStudent()
Input: Student_Id, Student_name
Output: Updation of student details in the student's database.
Method Description:
This method takes ID,name of the student as input and updates the student details in the student's database.

Method 4: CancelThe Slot()
Input: Id,Timeslot
Output: Cancellation of the slot
Method Description:
The method takes input of Id and timeslot of the lab and cancels that slot.

Method 5: AddnewFaculty()
Input: Id , Name , Password
Output: Added successfully page.
Method Description:
This method takes input of the id , Name of the faculty and password from the admin and adds the new faculty to the faculty details database to give access for the new faculty to book labs for his sections.

Method 6: ModifyFaculty()

Input: Id, Name
Output: Updated successfully page.
Method Description:

This method takes input of the Id and Name of the faculty from the admin and is able to edit the faculties details like the Id, Name and also the Password in the database.

Method 7: ManageLabs()
Input: ID
Output: Outcome landing page of success.
Method Description:

This method takes input of the Lab Id and allows the admin to either block the lab from being booked.

Method 8: Manage Resources Database()
Input: ID, Count
Output: Updated successfully page.
Method Description:

This method takes input of the resource id and the reserve count of the and updated it in the resource database.

Method 9: ManageTimeSlot()
Input: Id
Output: Updated successfully page
Method Description:

This method takes input of the lab id and assigns the different time slots that are available for that specific lab.

3.2.2 Class name: Faculty

Description: This class enables the Faculty to enter into the subsystem (Landing page) after authenticating the entered credentials.

Method 10: Login()
Input: email ,password
Output: Admin landing page of login successful
Method Description:

This method basically takes input as email and password from the user and checks whether it is authorized login or not and if the result is successful it leads to another activity page.

Method 11: Logout()
Input: logout option.
Output: Admin goes back to the login page.
Method Description:

This method basically takes the user back to the login page.

Method 12: ModifyStudent()
Input: Student_Id, Student_name
Output: Updation of student details in the student's database.
Method Description:
This method takes ID,name of the student as input and updates the student details in the student's database.

Method 13: CancelThe Slot()
Input: Id,Timeslot
Output: Cancellation of the slot
Method Description:
The method takes input of Id and timeslot of the lab and cancels that slot.

Method 14: AddNewStudent()
Input: Id,Email,sectionId
Output: Updated the student page
Method Description:
The faculty is able to add the new student to the database.

Method 15: AddNewSection()
Input: faculty id, section id
Output: New section created page
Method Description:
This method takes the input of the faculty id and the section id and adds this to the section database.

Method 16: ModifySection()
Input: faculty id, section id
Output: Updated the section page
Method Description:
This method takes the input of the faculty id and section id and it gives the ability to remove the section completely.

Method 17: ScheduleASlot()
Input: lab id, timeslot, section id
Output: Booking of lab is successful or lab is already booked.
Method Description:
This method takes the input of the lab id, timeslot and section id to check if a lab is booked for that specific time slot. If it is empty then it books that lab for that specific section id.

Method 18: ScheduleResources()
Input: resource id,lab id,timeslot,count

Output: Booking of resources is successful or resources are already booked.
Method Description:

This method takes the input of the resource id,lab id, timeslot and the count of the quantity of the specific resources needed. Then the method looks through the database for resources and checks if there are enough quantities of the resources and then assigns them to the lab id for that particular time slot.

Method 19: Register()
Input: Id,Name,Email,Password
Output: registered successfully page
Method Description:

This method takes the input of the faculty id, name, email and the password. Then this data gets stored in the faculty database and then gives access to the new registered faculty the access to login and book a lab for his sections.

3.2.3 Class name: Lab

Description: This class enables the Faculty to enter into a web page, where he can manage the labs(book the slots or cancel the slots) after successful login.

Method 20: SendNotification()
Input: Id, TimeSlot, SectionID
Output: Notification is sent
Method Description:

This method takes the input of the lab id, timeslot and sectionID and sends a notification to every student and the teacher showing they are related to the imputed sectionID.

3.2.4 Class name: Student

Description: This class enables the Student to enter into the subsystem (Landing page) after authenticating the entered credentials.

3.2.5 Class name: Section

Description: This class enables the admin and faculty to enter into the web page, where they can manage the various sections for each faculty.

3.2.6 Class name: Resources

Description: This class enables the admin and faculty to enter into the web page, where they can request or update the resource inventory.

Method 21: AssignmentResources()
Input: ID, Count
Output: Updated successfully
Method Description:

This method takes the input of the resource ID and counts for that id and updates it in the Resources Database.

4.1 Execution Architecture:

Runtime requirement is any device compatible with running a web application locally.If hosted we stable require internet connection and can be run on any browser.To run locally we need an offline application on PC like netbeans , atom or visual studio.

4.2 Reuse and Relationships to other products:

NIL
AvailSys is an individual non-profitable web application for the convenience of faculty and synchronisation of scheduling laboratory resources in university.

5. Design Decisions and Tradeoffs:

The design decision to use two separate screens separately for admin and faculty is to provide encapsulation to avoid glitches in the application as it may be possible to get all the information on single and one screens.
However , two separate screens ensure that two screens will keep the data of Faculty and Admin separate and not accessible to the other set of users.

A possible tradeoff when considering links for navigation is to use buttons on the dashboard representing each function for the selected option.This design decision to use buttons betweens screens is to enhance and improve visibility for the user.

From, section 2.a : High level Design

We can check design as described , i.e; as the user enters his credentials and directed to the dashboard according to the type of user i.e; Admin or Faculty.

- If it is admin , He enters the dashboard of AdminType and he can see navigation buttons to execute functional features of Admin provided by AvailSys.He can see Faculty Management, Lab Management , Resource Management and student Management.
- If it is Faculty , He enters the dashboard of FacultyType and he can see navigation buttons to execute functional features of Faculty provided by AvailSys.He can see Section Management, Lab Management , Resource Management and Update profile.
- For faculty , in the Lab Management he can schedule or cancel the scheduled lab .

6. Pseudocode For Components

6.1 Class name: Admin

Method 1: Login()

Pseudo-Code:

Input: email ,password

Output: Admin landing page of login successful

1. The input email is stored into a variable string email
2. The input password is stored into a variable string password
3. If email is null then
4. Print the email is empty
5. Return
6. End if
7. If password is null then
8. Print the password is empty
9. Return
10. End if
11. Show the progress by printing “Logging in please wait”
12. verify(email,password) by sending to the database to check
13. The result from verify is stored here into a variable result
14. If the result is successful then
15. finish();
16. Start another activity leading to the admin landing page
17. End if

Method 2: Logout()

Pseudo-Code:

Input: logout option.

Output: Admin goes back to the login page.

1. Exit the page by returning to the login page

Method 3: ModifyStudent()

Pseudo-Code:

Input: Student_Id, Student_name

Output: Updation of student details in the student's database.

1. The input student_id is stored in the variable id
2. The input student_name is stored in the variable name
3. verify(id,name) by sending it to the database
4. If the verification is successful then
5. Take input of 1 or 0 (1 yes to update the id else no)
6. If 1 is taken then take input of new_student_id
7. Take input of 1 or 0 (1 yes to update the name else no)
8. If 1 is taken then take input of new_student_name
9. Update(id,name)
10. Else
11. Print "no such student found"
12. End if
13. Return to landing page

Method 4: CancelThe Slot()

Pseudo-Code:

Input: Id,Timeslot

Output: Cancellation of the slot

1. The input of the id is stored to lab_id variable
2. The input of the timeslot is stored into the variable timeslot
3. Verify(id,timeslot) by sending it to the database
4. If the verification is successful then
5. If (id is equal to the lab_id and timeslot is equal to the timeslot) then
6. If (timeslot_booked (bool) is true) then
7. Set the timeslot_booked to false;
8. Return;
9. Else print "no booking has been made"
10. Return;
11. End if
12. End if
13. Else
14. Print "invalid lab"
15. Return
16. End if
17. Return;

Method 5: AddnewFaculty()

Pseudo-Code:

Input: Id , Name , Password

Output: Added successfully page.

1. The input of the id is stored to the variable faculty_id
2. The input of the name is stored to the variable faculty_name
3. The input of the password is stored to the variable faculty_password
4. verify(id) by sending it to the database
5. If verification is successful then
6. Print “can create same faculty with same id”
7. Return;
8. Else
9. Insert (id,name,password) into the database
10. Return;
11. End if
12. Return;

Method 6: ModifyFaculty()

Pseudo-Code:

Input: Id, Name

Output: Updated successfully page.

1. The input id is stored in the variable id
2. The input name is stored in the variable name
3. verify(id,name) by sending it to the database
4. If the verification is successful then
5. Take input of 1 or 0 (1 yes to update the id else no)
6. If 1 is taken then take input of new_student_id
7. Take input of 1 or 0 (1 yes to update the name else no)
8. If 1 is taken then take input of new_student_name
9. Update(id,name)
10. Return;
11. Else
12. Print “no such student found”
13. Return;
14. End if
15. Return to landing page

Method 7: ManageLabs()

Pseudo-Code:

Input: ID

Output: Outcome landing page of success.

1. The input id is stored in the variable id
2. verify(id) by sending it to the database
3. If the verification is unsuccessful then
4. Take input of 1 or 0 (1 yes to add new lab else no)
5. Return;
6. Else
7. Take input of 1 or 0 (1 yes to delete lab else no)
8. Return;
9. End if
10. Return;

Method 8: Manage Resources Database()

Input: ID, Count

Output: Updated successfully page.

1. The input id is stored in the variable id
2. The input count is stored in the variable quantity
3. verify(id) by sending it to the database
4. If the verification is unsuccessful then
5. Take input of 1 or 0 (1 yes to add new resource and update quantity else no)
6. Return;
7. Else
8. Take input of 1 or 0 (1 yes to update the quantity of the resource else no)
9. Return;
10. End if
11. Return;

Method 9: ManageTimeSlot()

Input: lab_Id

Output: Updated successfully page

1. The input lab_id is stored in the variable id
2. The input count is stored in the variable quantity
3. verify(id) by sending it to the database
4. If the verification is successful then
5. Takes input of timeslot to be added
6. verify(id,timeslot) in the database
7. If verification is successful then

8. Print “time slot already exists”
9. Return
10. Else
11. Updates a new timeslot to the database for that lab_id
12. End if
13. Else
14. Print “lab does not exist”
15. Return;
16. End if
17. Return;

Method Description:

This method takes input of the lab id and assigns the different time slots that are available for that specific lab.

6.2 Class - Faculty

Method 10: Login()

Input: email ,password

Output: Admin landing page of login successful

Pseudo code:

1. read email
2. read password
3. if verify(email,password) is true then
4. then login successfully
5. else
6. Login failed
7. end if

Method 11: Logout()

Input: logout option.

Output: Admin goes back to the login page.

Pseudo code:

1. if logout then
2. return back to the login page.

Method 12: ModifyStudent()

Input: Student_Id, Student_name

Output: Updation of student details in the student’s database.

Pseudo code:

1. read Student_Id
2. read Student_Name
3. if verify(Student_Id, Student_Name) is true then
4. User enters 1 for updating Student_Id

5. Take input and store in change_ id
6. if change_id == 1 then
7. User will be asked to enter new_Student_Id
8. update(new_Student_Id,Student_Name)
9. Print message Updation successful
10. end if
11. User enters 1 for updating Student_Name
12. Take input and store in change_ Name
13. if change_Name == 1 then
14. User will be asked to enter new_Student_Name
15. update(new_Student_Id,new_Student name)
16. Print message Updation successful
17. end if
18. else print invalid details
19. end if

Method 13: CancelThe Slot()

Input: Id,Timeslot

Output: Cancellation of the slot

Pseudo code:

1. Read Id
2. Read timeslot
3. if Verify(Id,timeslot) is true then
4. if (timeslot_booked (bool) is true) then
5. Set the timeslot_booked to false;
6. return;
7. Else print “no booking has been made”
8. return;
9. End if
10. Else
11. Print “invalid lab”
12. return;
13. End if
14. Return;

Method 14: AddNewStudent()

Input: Id,Email,sectionId

Output: Updated the student page

Pseudo code:

1. Read id
2. Read email
3. Read sectionId
4. If verify(id) is true then
5. print student already present

6. Else
7. insert(id,email,sectionId) into database
8. Print successfully added student
9. End if
10. Return;

Method 15: AddNewSection()

Input: faculty id, section id

Output: New section created page

Pseudo code:

1. Read faculty_id
2. Read section_id
3. If verify(faculty_id,section_id) is true then
4. Printf section already present
5. Else
6. insert(faculty_id,section_id) into database
7. Print successfully added new section
8. End if

Method 16: ModifySection()

Input: faculty id, section id

Output: Updated the section page

Pseudo code:

1. Read faculty_id
2. Read section_id
3. If verify(section_id,faculty_id) is true then
4. Take input of 0 or 1
5. If 1 then
6. Delete section
7. Return;
8. End if
9. Take input 0 or 1 or 2
10. If 1 then
11. Read studentId,StudentName
12. modifyStudent(StudentId,StudentName)
13. return ;
14. Else if 2 then
15. Read studentId,StudentName
16. AddNewStudent(StudentId,StudentName)
17. return ;
18. End if
19. Else
20. Print invalid section

21. Return;
22. End if
23. Return;

Method 17: ScheduleASlot()

Input: lab id, timeslot, section id

Output: Booking of lab is successful or lab is already booked.

Pseudo code:

1. read lab_id
2. Read timeslot
3. Read sectionId
4. if timeslot_booked (bool) is false and lab_booked(bool) is false then
5. print successfully booked the lab
6. Else
7. print booking failed
8. End if

Method 18: ScheduleResources()

Input: resource id,lab id,timeslot,count

Output: Booking of resources is successful or resources are already booked.

Pseudo code:

1. read lab_id
2. Read timeslot
3. Read sectionId
4. Read no_resources;
5. if Resources_available(no_resources) is false then
6. print successfully scheduled resources
7. Else
8. print booking failed
9. End if

Method 19: Register()

Input: Id,Name,Email,Password

Output: registered successfully page

Pseudo code:

1. Read faculty_id
2. Read name
3. Read email
4. Read password
5. if verify(email,password) is true then

6. Print account already exist
7. Else
8. print Registration successful
9. End if

3.2.3 Class name: Lab

Method 20: SendNotification()

Input: Id, TimeSlot, SectionID

Output: Notification is sent

Pseudo Code:

1. Read the id as input
2. Read the time slot as input
3. Read the sectionID as input
4. If verify(id, TimeSlot, SectionID) is true then
5. Print “Notification is sent”
6. Python code is run to send notification to all the students with the same sectionID and Faculty
7. Return;
8. End if
9. Return;

3.2.6 Class name: Resources

Method 21: AssignmentResources()

Input: ID, Count

Output: Updated successfully

Pseudo Code:

1. The input id is stored in the variable id
2. The input count is stored in the variable quantity
3. verify(id) by sending it to the database
4. If the verification is unsuccessful then
5. Take input of 1 or 0 (1 yes to add new resource and update quantity else no)
6. Return;
7. Else
8. Take input of 1 or 0 (1 yes to update the quantity of the resource else no)
9. Return;
10. End if
11. Return;

7.Traceability Matrix :

REQUIREMENT ID : from software requirement specifications document(4.1*.1*)
USE CASE - ID: from software requirement specifications (represented as UC - 1*)
HIGH LEVEL DESIGN : from Sequence Diagram (represented by SD - 1*.1*)
LOW LEVEL DESIGN - 1: from class diagram classes (represented by CD - 3.2.1*)
LOW LEVEL DESIGN - 2:from class diagram methods (represented by M1*)
* - represents the continuation of series

REQUIREMENT ID	USE CASE - ID	HIGH LEVEL DESIGN	LOW LEVEL DESIGN 1	LOW LEVEL DESIGN 2
4.1.1	UC - 1,UC - 8	SD - 1.1	CD - 3.2.1	M1,M2
4.1.2	UC - 2	SD - 1.2 SD - 1.6	CD - 3.2.1 CD - 3.2.2 CD - 3.2.4 CD - 3.2.5	M3,M12,M14, M15,M16
4.1.3	UC - 3	SD - 1.2 SD - 1.3	CD - 3.2.1 CD - 3.2.2	M5,M6,M19
4.1.4	UC - 4	SD - 1.2 SD - 1.5	CD - 3.2.1 CD - 3.2.2	M7,M9,M17
4.1.5	UC - 5	SD - 1.2 SD - 1.4	CD - 3.2.1	M8
4.1.6	UC - 6 , UC - 13	SD - 1.2 SD - 1.4 SD - 1.5	CD - 3.2.1	M7
4.2.1	UC - 7	SD - 2.1	CD - 3.2.2	M19
4.2.2	UC - 1,UC - 8	SD - 2.1	CD - 3.2.2	M10,M11
4.2.3	UC - 9	SD - 2.2	CD - 3.2.1	M5,M6
4.2.4	UC - 10	SD - 2.2 SD - 2.3	CD - 3.2.2 CD - 3.2.5	M15,M16
4.2.5	UC - 11	SD - 2.2 SD - 2.3 SD - 2.4	CD - 3.2.1 CD - 3.2.2	M9,M17
4.2.6	UC - 12	SD - 2.2 SD - 2.3 SD - 2.5	CD - 3.2.2 CD - 3.2.5	M18,M21
4.2.7	UC - 6, UC - 13	SD - 2.2	CD - 3.2.2	M13

4.2.8	UC - 14,UC - 15	SD - 2.2 SD - 2.3 SD - 2.4 SD - 2.5	CD - 3.2.1 CD - 3.2.2 CD - 3.2.3 CD - 3.2.4	M7,M9,M13, M17,M20
4.3.1	UC - 14,UC - 15	SD - 2.2 SD - 2.3 SD - 2.4 SD - 2.5	CD - 3.2.1 CD - 3.2.2 CD - 3.2.3 CD - 3.2.4	M7,M9,M13, M17,M20

8.Appendix:

Requirements :

4.1.1 - Login
4.1.2 - Manage Student Database
4.1.3 - Manage Faculty Database
4.1.4 - Manage Lab
4.1.5 - Manage Resource Database
4.1.6 - Cancel the Slot
4.2.1 - Register
4.2.2 - Login
4.2.3 - Update Details
4.2.4 - Manage Section
4.2.5 - Schedule a Slot
4.2.6 - Schedule Resource
4.2.7 - Cancel Slot
4.2.8 - Receive Emails
4.1.1 - Receive Emails

Use Case Diagram:

Use Case - 1: Login
Use Case - 2: Manage Student Database
Use Case - 3: Manage Faculty Database
Use Case - 4: Manage Labs
Use Case - 5: Manage Resource Database

Use Case - 6:Cancel the Slot
Use Case - 7:Register
Use Case - 8:Login
Use Case - 9:Update Details
Use Case - 10:Manage Sections
Use Case - 11:Schedule Lab
Use Case - 12:Schedule Resource
Use Case - 13:Cancel Slot
Use Case - 14:Receive Emails
Use Case - 15:Receive Emails

Class Diagram Classes :

Class Diagram - 3.2.1:Admin
Class Diagram - 3.2.2:Faculty
Class Diagram - 3.2.3:Lab
Class Diagram - 3.2.4:Student
Class Diagram - 3.2.5:Section
Class Diagram - 3.2.6:Resource

CLASS DIAGRAM	**METHODS**
CD - 3.2.1	M1,M2,M3,M4,M5,M6,M7,M8,M9
CD - 3.2.2	M10,M11,M12,M13,M14,M15,M16,M17,M18,M19
CD - 3.2.3	M20
CD - 3.2.4	NIL
CD - 3.2.5	NIL
CD - 3.2.6	M21

Class Diagram Methods:

Method-1: Login()
Method-2: Logout()

Method-3: ModifyStudent()
Method-4: CancelTheSlot()
Method-5: AddNewFaculty()
Method-6: ModifyFaculty()
Method-7: ManageLabs()
Method-8: ManageResourceDatabase()
Method-9: ManageTimeSlot()
Method-10: Login()
Method-11: Logout()
Method-12: ModifyStudent()
Method-13: CancelTheSlot()
Method-14: AddNewStudent()
Method-15: AddNewSection()
Method-16: ModifySection()
Method-17: ScheduleASlot()
Method-18: ScheduleResource()
Method-19: Register()
Method-20: SendNotification()
Method-21: AssignmentResources()

Sequence Diagram :

Sequence Diagram - 1:Admin
1.1 - Login
1.2 - Dashboard
1.3 - Faculty Management
1.4 - Resource Management
1.5 - Lab Management
1.6 - Student Management

Sequence Diagram - 2:Faculty
2.1 - Login Success
2.2 - Dashboard
2.3 - Section management
2.4 - Lab Management
2.5 - Resource Management