

**Software Requirements Specification**

**Final Document**

**App to Schedule Appointment**

**Team Members**

Gunnam, Aravinda Srinivasa Vara Prasad: 1795075

Katakam, Bala Kavya : 1794948

Kusukuntla, Vijayender Reddy : 1794960

Madugula, Mahender Reddy : 1794805

Matta, Sirisha : 1794858

Nayani, Jay Ram : 1795596

Sama, Saikumar Reddy : 1794716

Handed to

##### <Gilles Philiipe Gregoire>

**Version: 1.0** **Date: (24/08/2019)**

[Table of Contents](#_Toc17574185)

[**[SECTION-1](#_Toc17574185)** [3](#_Toc17574185)](#_Toc17574279)

[[1. INTRODUCTION 3](#_Toc17574185)](#_Toc17574280)

[[1.1. Purpose: 3](#_Toc17574185)](#_Toc17574281)

[[1.2. Scope: 3](#_Toc17574185)](#_Toc17574282)

[[1.3. Overview: 3](#_Toc17574185)](#_Toc17574283)

[**[SECTION-2](#_Toc17574185)** [4](#_Toc17574185)](#_Toc17574284)

[[2.1 Overall Description: 4](#_Toc17574185)](#_Toc17574285)

[[2.2 Product Functions: 4](#_Toc17574185)](#_Toc17574286)

[[2.2 Non functionalities: 6](#_Toc17574185)](#_Toc17574287)

[**[SECTION -3](#_Toc17574185)** [8](#_Toc17574185)](#_Toc17574288)

[[3.1 Use Case Description: 8](#_Toc17574185)](#_Toc17574289)

[[1. Login: 11](#_Toc17574185)](#_Toc17574290)

[[2. View staff: 12](#_Toc17574185)](#_Toc17574291)

[[3. Set Appointments: 13](#_Toc17574185)](#_Toc17574292)

[[4. Current Appointments: 14](#_Toc17574185)](#_Toc17574293)

[[5. Appointments history: 16](#_Toc17574185)](#_Toc17574294)

[[6. Appointments request: 16](#_Toc17574185)](#_Toc17574295)

[[7. Set Availability: 17](#_Toc17574185)](#_Toc17574296)

[[8. View Profile: 19](#_Toc17574185)](#_Toc17574297)

[[9. View Available staff: 20](#_Toc17574185)](#_Toc17574298)

[[10. Logout: 21](#_Toc17574185)](#_Toc17574299)

[[3.2 GUI Description: 22](#_Toc17574185)](#_Toc17574300)

[[3.3 Class Diagram: 31](#_Toc17574185)](#_Toc17574301)

[[3.3.1 Entity Relation Diagram: 31](#_Toc17574185)](#_Toc17574302)

[[3.4. Web Services: 32](#_Toc17574185)](#_Toc17574303)

[[3.4 SQL Queries: 54](#_Toc17574185)](#_Toc17574304)

[[SECTION-4 60](#_Toc17574185)](#_Toc17574305)

[[4.1. Development Team: 60](#_Toc17574185)](#_Toc17574306)

[[4.1.1. Front End Developer: 60](#_Toc17574185)](#_Toc17574307)

[[4.1.2. Back End Developer: 61](#_Toc17574185)](#_Toc17574308)

[[4.1.3. Android Developer: 62](#_Toc17574185)](#_Toc17574309)

[[4.1.6. Project Manager: 63](#_Toc17574185)](#_Toc17574310)

[[4.1 Project deliverables: 64](#_Toc17574185)](#_Toc17574311)

[3](#_Toc17574185)

[1. INTRODUCTION 3](#_Toc17574186)

[1.1. Purpose: 3](#_Toc17574187)

[1.2. Scope: 3](#_Toc17574188)

[1.3. Overview: 3](#_Toc17574189)

[**SECTION-2** 4](#_Toc17574190)

[2.1 Overall Description: 4](#_Toc17574191)

[2.2 Product Functions: 4](#_Toc17574192)

[2.2 Non functionalities: 6](#_Toc17574193)

[**SECTION -3** 8](#_Toc17574194)

[3.1 Use Case Description: 8](#_Toc17574195)

[1. Login: 11](#_Toc17574196)

[2. View staff: 12](#_Toc17574197)

[3. Set Appointments: 13](#_Toc17574198)

[4. Current Appointments: 14](#_Toc17574199)

[5. Appointments history: 16](#_Toc17574200)

[6. Appointments request: 16](#_Toc17574201)

[7. Set Availability: 17](#_Toc17574202)

[8. View Profile: 19](#_Toc17574203)

[9. View Available staff: 20](#_Toc17574204)

[10. Logout: 21](#_Toc17574205)

[3.2 GUI Description: 22](#_Toc17574206)

[3.3 Class Diagram: 31](#_Toc17574207)

[3.3.1 Entity Relation Diagram: 31](#_Toc17574208)

[3.4. Web Services: 32](#_Toc17574209)

[3.4 SQL Queries: 54](#_Toc17574210)

[SECTION-4 60](#_Toc17574211)

[4.1. Development Team: 60](#_Toc17574212)

[4.1.1. Front End Developer: 60](#_Toc17574213)

[4.1.2. Back End Developer: 61](#_Toc17574214)

[4.1.3. Android Developer: 62](#_Toc17574215)

[4.1.6. Project Manager: 63](#_Toc17574216)

[4.1 Project deliverables: 64](#_Toc17574217)

# 

# **SECTION-1**

## 1. INTRODUCTION

### 1.1. Purpose:

The main purpose of scheduling an appointment in education system for staff members is to know when the staff members are available on particular days and date. Everyone can reach them by looking at their availabilities provided by themselves. Schedule connects to your calendar and uses it as its basis for determining availability and for management of scheduled meetings and appointments.

### 1.2. Scope:

It maintains the time efficiency that adds, update and delete in various type of data from the system. The processing time for the staff decreases and they can be done their work within minutes or seconds rather than the hours.

The system can have accurate records regarding the staff names, so that if they lose the data in the system the data will be stored in the repositories.

It has the Web-portal that means the system can access 24/7 except when the server goes down. It’s very efficient way to reach staff by seeing their availabilities in online instead meeting them to know their availability.

### 1.3. Overview:

The next section, the Overall Description section, of this document gives an overview of the functionality of the application. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next section.

The third section, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the application.

# 

# **SECTION-2**

## 2.1 Overall Description:

The application to schedule an appointment describes that the admin adds the staff in the system with particular identification to each and every staff member. After then staff can login with their credentials and can make an appointment. If he/she wants to make any changes to the appointment they can update the appointment details and if they wants to remove the appointment they can simply delete the unwanted history.

But this application shows when the staff members are available in particular day, date, time. All the staff members can view all appointments. But they cannot make any changes to other staff member appointment, but they can make changes when they login to their account.

## 2.2 Product Functions:

Functional requirements of a system can be related between hardware and software or both in terms of calculations. Technical details, data manipulation and processing or both specific functionality that defines a system is supposed to accomplish.

In our application, we are having functional requirements for both admin and staff.

**ADMIN:**

1. **Login:**

Admin shall be able to login to the system to do further operations which are to be done by the admin. But to perform any function admin at first has to login to the system.

1. **View staff:**

Next is admin can view the entire staff details by view staff function.

**STAFF:**

1. **Login:**

Staff shall be able to login to the system to do further operations which are to be done by the staff. But to perform any function staff at first has to login to the system.

1. **Set Appointments:**

In this functionality the staff has to login at first, the staff has to set an appointment by choosing time, date and to which category appointment belongs.

1. **Current Appointments:**

In this functionality the staff can view the present appointments and by using the selection option the staff can update an appointment or delete an appointment.

1. **Appointments history:**

The next functionality is appointments history. Staff can view their appointments previous appointments.

1. **Create profile:**

For every functionality staff has to be able to login to the system. After having their identification number staff can create their own profile by their personal details.

1. **Set Availability:**

In this functionality at first staff can set his/her availability. So that remaining staff can know their availabilities.

1. **Appointment requests:**

This functionality describes about the list of appointment requests send to the other staff member and the particular staff can accept or decline the appointment according to their availability.

1. **Accepted requests:**

This functionality depicts the staff members list who accepts the appointments which is created by the particular staff and can the particular staff who created appointment can view which staff member has accepted which particular appointment.

1. **View available staff:**

In this view available staff one staff can view other staff availabilities. At first they can see by their Id and show availabilities option, when the staff wants to see any of the staff availability they can view them.

1. **Logout:**

Logout is to be done when every functions are done or any one of the functions is done that depends on user.

## 2.2 Non functionalities:

Non -functional requirement is a requirement that specifies criteria that can be used to judge the operation of system rather than specific behaviors. Non-functional requirement is also known as “quality attributes” of a system. The attributes of the system explains below.

1. **Reliability :**

Requirements about how often the software fails. The measurement is often expressed in MTBF (mean time between failures). The definition of a failure must be clear. Be sure to specify the consequences of software failure, how to protect from failure, a strategy for error detection, and a strategy for correction.

1. **Performance:**

The performance of the whole system aims to ensure ease of comfort to have the site load within seconds. The priority of the performance is semi-low because as you develop the system, the focus will be come from design of the system and the coding of the system. Performance of the system will come once everything else of the system has been finalized.

1. **Usability:**

The usability is moderately-high priority because the client will not accept a working system where the users are unable to use, read or operate the system. Client cannot accept until he reaches what functionality the client wants. Therefore developers will ensure the website or an application must be smooth design in which the client and users are able to read and use the features of an application or website. If the client and user cannot understand how to use, then they will not use the website or application.

1. **Privacy and Security:**

The system’s privacy and security needs to protect the user’s login details and as well as the database having encryption, so that the users and the organization’s data is neither comprised. If the implementation of the system is done correctly to the model view controller, then the system will be able to protect user’s details and the database.

1. **Compatibility:**

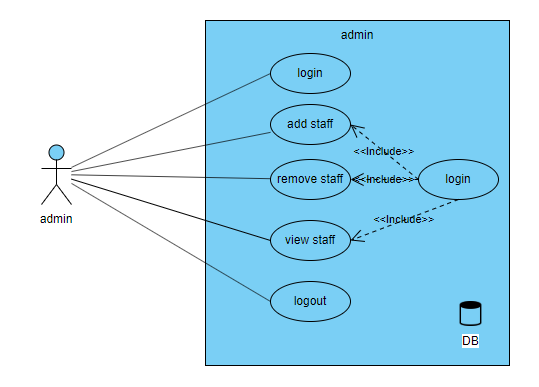
Compatibility requirements ensures that requirements ensures that application or system able to perform well on different set of platforms and software. The key compatibility are platforms, APIs and drivers, web browser and mobile devices.

1. **Maintainability:**

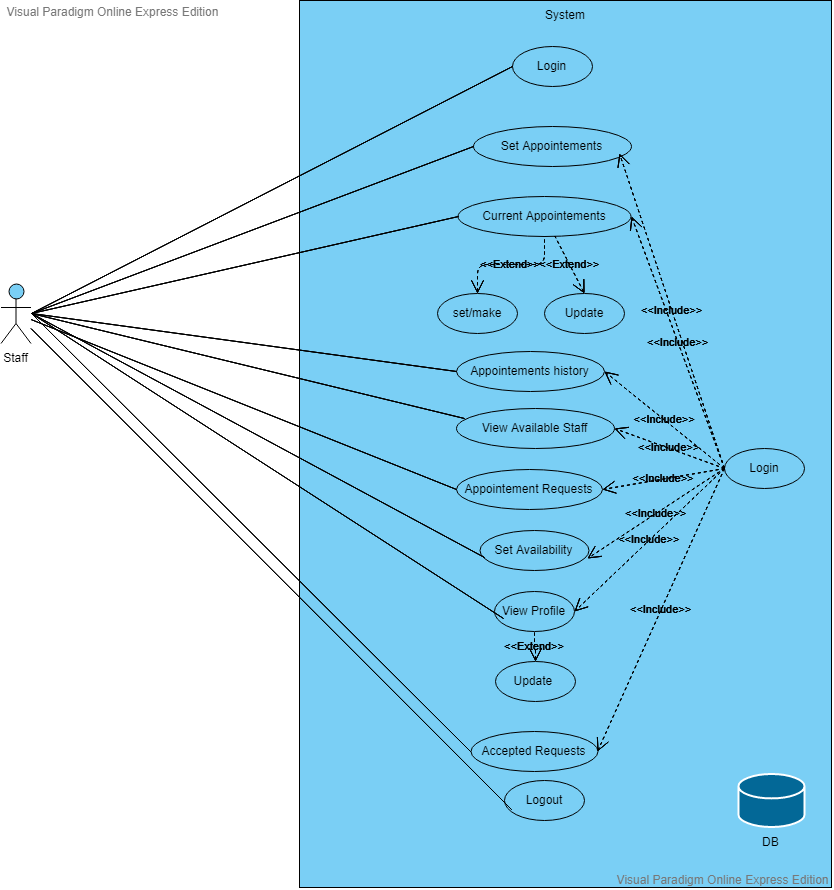
Maintainability is the ability of the application to go through changes with a fair degree of effortlessness. This attribute is the ability to be easily modified with which the application can be modified, for fixing issues, or to add new functionality with an easy way. Maintainability attributes will enhance availability and reduce runtime defects.

### **SECTION -3**

### 3.1 Use Case Description:

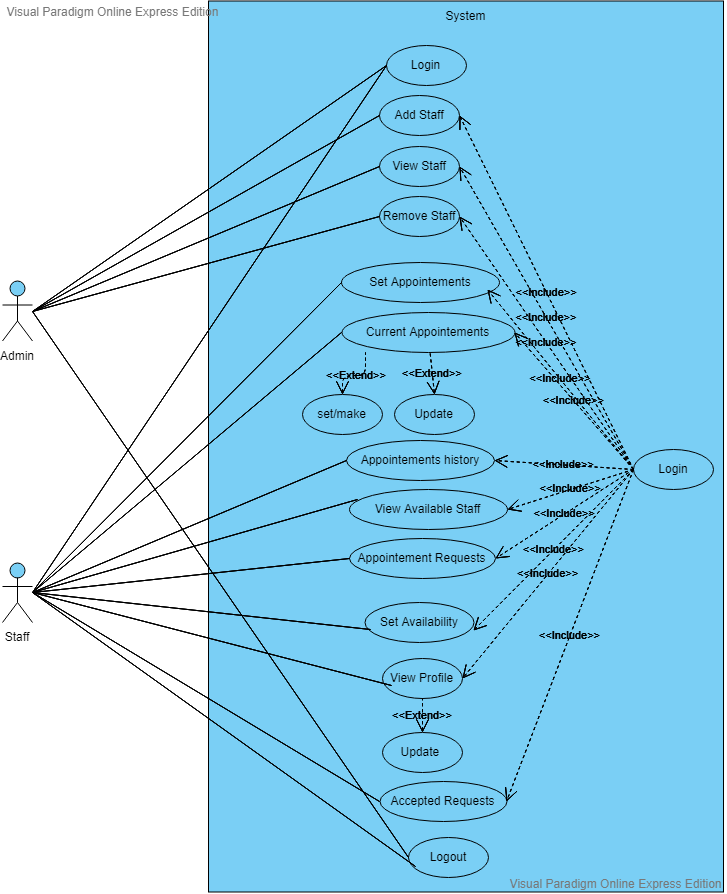


Here the actor is Admin, login use case is to login by the admin and add the staff with their details, remove staff use case is used to remove staff by admin can access only through the web. In View staff admin can see all staff members.



**Staff use case**

The above figure depicts the use case for staff. At first staff has to login to the system, then he/she can have the possibility to access appointments use case. Appointments extends with three other make, update and delete functionalities. Next to view the appointments staff has to login to view the appointments made. Staff can create profile when he/she logins. To view staff availabilities every staff has to login to view. At last after every functionality is done by the staff they can logout.



### Login:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | 1 | | |
| Use Case Name | Login | | |
| Created by | Kavya | Last updated by : Sama Sai Kumar Reddy | |
| Date created | 10/01/2019 | Date last updated : 11/01/2019 | |
| Actor | Admin and staff | | |
| Description | User can login with his details. | | |
| Triggering event | The user can access the login page to schedule an appointment | | |
| Preconditions | The login page opens. | | |
| Flow of events | Actor | | System |
| 1. The user clicks login page. 2. The user enters the user id. 3. The password is entered by the user. 4. User is signed in and returned to the homepage as logged in user. | | 1.The system displays the login page with user id and password  2. The system checks for the username entered by the user and it asks the user to enter password.  3. The system checks the database for the password entered by the user is valid or invalid and displays there by display menu |
| Alternative low of events | Actor | | System |
| 1. The user enters the user id. 2. The user creates new password and confirm password 3. The user can login. | | 1. The system shows reasons why the user failed.   2. The system presents the user with suggestions for changes, it allows the user to pass authentication and ask user to re-enter the valid information.  3. The system checks for the user id, password entered by the user according to the conditions given by the system. |
| Post conditions | Now the user is able to access the application. | | |
| Exception – Conditions | If the user enters wrong password system gives the user chance to create a new password. | | |

### 

### 2. View staff:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | 2 | | |
| Use Case Name | View Staff | | |
| Created by | Aravind | Last updated by :Jay Ram | |
| Date created | 18/07/2019 | Date last updated : 18/07/2019 | |
| Actor | Admin | | |
| Description | The admin is having the access to view the staff information, so that admin can view the information. | | |
| Triggering event | The admin is able to access the application and can view the information about staff. | | |
| Preconditions | The user has login successfully | | |
| Flow of events | Actor | | System |
| 1. The user selects the option view staff. | | 1. The system displays the id, name of enroll staff. |
| Post conditions | The enrolled staff details can be seen. | | |
| Exception - Conditions | If there is no staff the admin cannot view details of the staff. | | |

### 3. Set Appointments:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | 3 | | |
| Use Case Name | Set Appointments | | |
| Created by | Kavya | Last updated by :Kavya | |
| Date created | 18/07/2019 | Date last updated : 18/07/2019 | |
| Actor | Staff | | |
| Description | The staff is having the access to make the appointments. | | |
| Triggering event | The staff is able to access the application and can do the operations regarding the appointments. | | |
| Preconditions | The user has login successfully | | |
| Flow of events | Actor | | System |
| 1. The user selects the option set appointments.        1. The user selects date by using calendar. 2. The user selects start time and end time. 3. The user enters the room number. 4. The user selects category list 5. The user then clicks set. 6. The user receives a successfully created message. | | 1. The system displays set appointment page with some specific fields date, start time, end time, room number and category. 2. The system shows the calendar with date. 3. The system shows the time to select. 4. The system checks the room number is exists or not. 5. The system shows the list of categories. 6. The system checks the information given by user is correct or not. |
| Alternative low of events | Actor | | System |
| 1.The user selects set appointments   1. The user enters the room number. 2. The user again enters valid number. 3. The user gets successfully message. | | 1. The system displays a page with date ,start time, end time, room number 2. The system checks whether the room number entered is saved in database or not. If the number is not in database automatically it is invalid room number. |
| Post conditions | By set appointments user can able to create an appointment and can save the appointment. | | |
| Exception – Conditions | If we don’t have the room number the staff don’t know where to attend the appointment. | | |

### 4. Current Appointments:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | 4 | | |
| Use Case Name | Current Appointments | | |
| Created by | Kavya | Last updated by :Aravind | |
| Date created | 18/07/2019 | Date last updated : 18/07/2019 | |
| Actor | Staff | | |
| Description | The staff can see only the present and future appointments by that the staff can update the appointments. | | |
| Triggering event | The staff is able to access the application and can do the operations regarding the appointments. | | |
| Preconditions | The user has login successfully | | |
| Flow of events | Actor | | System |
| 1. The user selects the option current appointments.        1. If the user wants to update particular appointment, then the user selects update option. 2. The user can only update any one of the field at a time provided by the system. 3. The user has to invite and gets successfully updated message. | | 1. The system displays only current and future appointments by including:  * Update  1. The system displays the page to create appointment including:  * Date, * Start time * End time * Room number |
| Alternative low of events | Actor | | System |
|  | |  |
| Post conditions | By using selection of particular appointment the update and delete operations are done. | | |
| Exception – Conditions | If there is no selection of an appointment the user cannot update or delete.  At once only one selection can be used for update and multiple selections are can be used for delete. | | |

### 5. Appointments history:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | 5 | | |
| Use Case Name | Appointments history | | |
| Created by | Mahender Reddy | Last updated by : Sirisha | |
| Date created | 20/07/2019 | Date last updated : 20/07/2019 | |
| Actor | Staff | | |
| Description | When staff wants to see the appointments he/she can view the appointments of the past. | | |
| Triggering event | The user is able to access the application once when the user is login. | | |
| Preconditions | Then is now able to view appointments when the user login the application. | | |
| Flow of events | Actor | | System |
| 1. The user selects appointments history. 2. The user can view the appointments. | | 1. The system will display the appointments of past. |
| Post conditions | The staff can view their own appointments. | | |
| Exception - Conditions | If there is no appointment done by the staff member, he/she cannot view the appointments. | | |

### 6. Appointments request:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | 6 | | |
| Use Case Name | Appointment requests | | |
| Created by | Kavya | Last updated by :Kavya | |
| Date created | 18/07/2019 | Date last updated : 18/07/2019 | |
| Actor | Staff | | |
| Description | The staff can receive requests from any other staff regarding appointments. | | |
| Triggering event | The staff is able to access the application and can do the operations regarding the appointments. | | |
| Preconditions | The user has login successfully | | |
| Flow of events | Actor | | System |
| 1. The user selects the option appointments requests. 2. The user selects any of the appointment      1. The user clicks accept. 2. If the user clicks decline | | 1. The system displays the invitations page with present and future appointments.  2. The system displays another page with request accept or decline for an appointment.  3. The system gives a message that invite has been accepted.  4. The system gives a message that invite has been declined. |
| Post conditions | By creating an appointment there will be option to send request to other staff member and can accept or decline the appointments. | | |
| Exception – Conditions | If we don’t have the appointment the staff cannot invite other staff. | | |

### Set Availability:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | 7 | | |
| Use Case Name | Set Availability | | |
| Created by | Vijayender Reddy | Last updated by :Kavya | |
| Date created | 18/07/2019 | Date last updated : 18/07/2019 | |
| Actor | Staff | | |
| Description | The staff is having the access to set the availability. | | |
| Triggering event | The staff can set availability when they are login. | | |
| Preconditions | The user has login successfully | | |
| Flow of events | Actor | | System |
| 1. The user selects the option set availability.        1. The user selects date by using calendar. 2. The user selects start time and end time. 3. The user chooses the day of the week 4. The user selects the day from the checklist. 5. The user then clicks set. 6. The user receives a successfully created message. | | 1. The system displays set appointment page with some specific fields date, start time, end time, day.   4. The system provides weekdays in the checklist.  6.The system verifies and send message. |
| Alternative low of events | Actor | | System |
| 1. The user selects set availability. 2. The user selects the blank in the checklist. 3. The user selects the day in the checklist. | | 1. The system displays a page with date, start time, end time, day. 2. The system checks for the week days in the database and it notifies the user to select the day. |
| Post conditions | By set availability user can able to create an availability and can save the availability. | | |
| Exception – Conditions | If we don’t have the weekday the staff cannot set availability. | | |

### 8. View Profile:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case ID | 8 | | | |
| Use Case Name | View profile | | | |
| Created by | Jay ram | Last updated by :Vijayender Reddy | |
| Date created | 18/07/2019 | Date last updated : 18/07/2019 | |
| Actor | Staff | | | |
| Description | The staff is having the access to view and even they can update the profile their own profile. | | | |
| Triggering event | The staff is able to access the application and they can update profile by their details. | | | |
| Preconditions | The user has login successfully | | | |
| Flow of events | Actor | | System | |
| 1. The user selects view profile.        1. If the user wants to update their profile then clicks update. 2. The user enters email. 3. The user enters phone number   .   1. The user gets profile has been successfully created. | | 1. The system displays information of the staff including:  * First Name, * Last name, * Email, * Designation, * Phone  1. The system displays name and designation, but they can update only email and phone. 2. The system verifies. 3. The system verifies entered number is correct or not. 4. The system create profile for particular staff. | |
| Alternative flow of events | Actor | | System | |
| 1. The user selects the option view profile. 2. The user enters the email. 3. The user confirms the message and enters the correct email. | | 1. The system displays the user information of the staff including:  * First Name, * Last name, * Email, * Designation, * Phone number  1. The system displays a message that tells the user about missed information or the entered information is not accepted because the entered mail is already exists. 2. The system create profile for the staff details. | |
| Post conditions | By the details entered the staff profile is created and saved. | | | |
| Exception – Conditions | If there is no staff in the management then there is no chance to create profile. | | | |

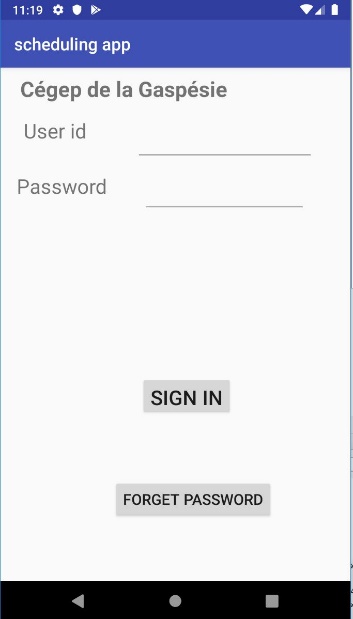
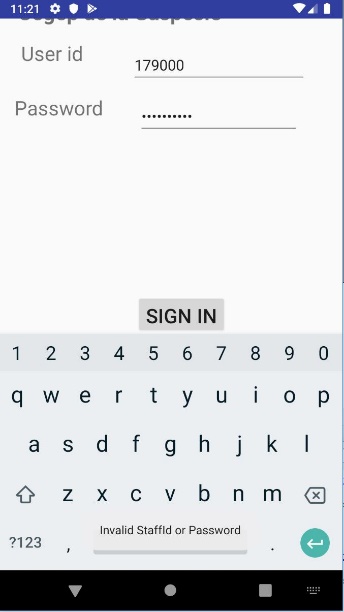
### 9. View Available staff:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | 9 | | |
| Use Case Name | View available staff | | |
| Created by | Sai Kumar Reddy | Last updated by :Aravind | |
| Date created | 18/07/2019 | Date last updated : 18/07/2019 | |
| Actor | Staff | | |
| Description | The staff members can view other staff availabilities by their own login. | | |
| Triggering event | The staff is able to access the application to view availabilities of remaining staff members. | | |
| Preconditions | The user has login successfully | | |
| Flow of events | Actor | | System |
| 1. The user selects view available staff.        1. The user can selects any one of the name from the list provided by the system and clicks availability. | | 1. The system displays the staff availability page with names and availability.  2. The system displays the availabilities of the selected staff member. |
| Post conditions | By the selecting any one of the name the remaining staff availabilities are known. | | |
| Exception – Conditions | If there is no staff availabilities in the system, they are unable to search for availabilities. | | |

### 10. Logout:

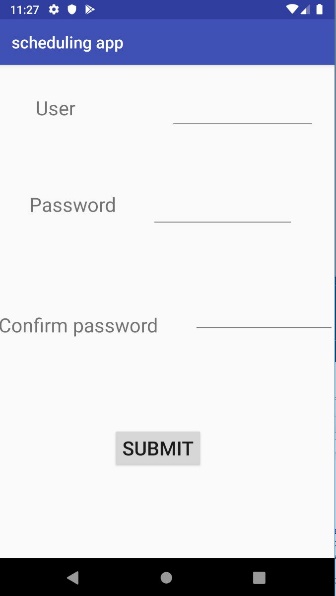
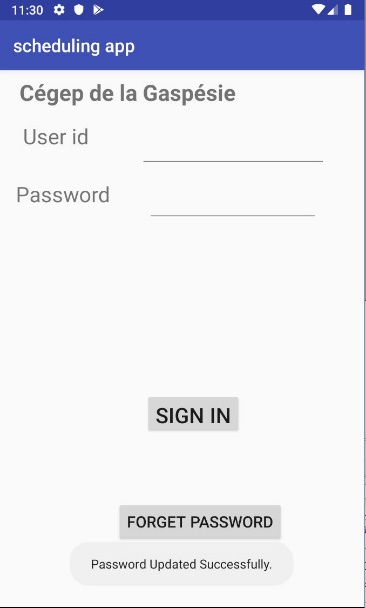
|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID | 10 | | |
| Use Case Name | Logout | | |
| Created by | Sirisha | Last updated by : Kavya | |
| Date created | 20/07/2019 | Date last updated : 20/07/2019 | |
| Actor | Admin and Staff | | |
| Description | When the user done the required action with the app then afterwards the user should logout. | | |
| Triggering event | The user is able to access the application once when the user is login. | | |
| Preconditions | Then is now able to logout when the user is already login the application. | | |
| Flow of events | Actor | | System |
| 1. The User will select “Logout” 2. The User will select an option (Y/N) 3. The User will continue as necessary | | 1. The System will bring up confirmation screen displaying, “Are you sure you would like to log out?” 2. The System will logout the User when (Y) .The System will bring the User back to main menu (N). |
| Post conditions | The user is successfully logout from the application. | | |
| Exception - Conditions | If the user didn’t select the option yes/no then the user couldn’t logout from the current application. | | |

## 3.2 GUI Description:

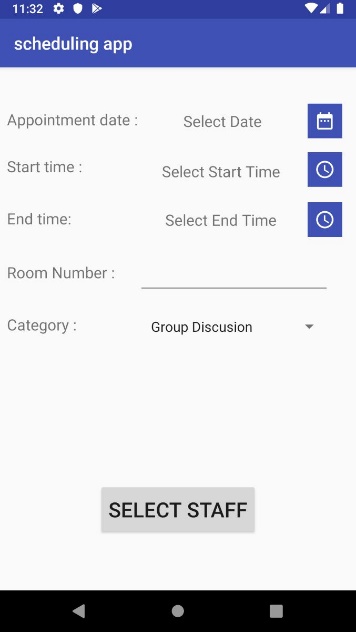
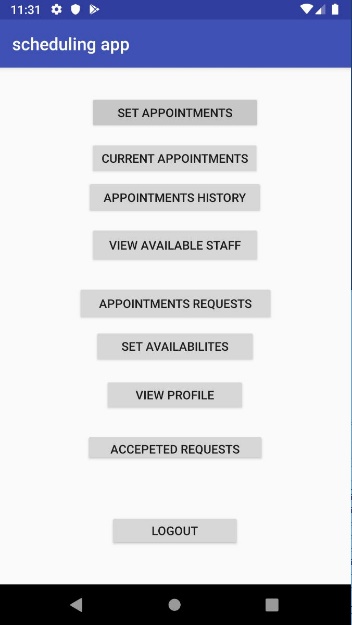
**Staff login login page-1**

The first screen depicts the login page of the user (staff), when the user enters wrong password it shows the message invalid staffed or password. So that the user can choose forgot password option to change his/her password.

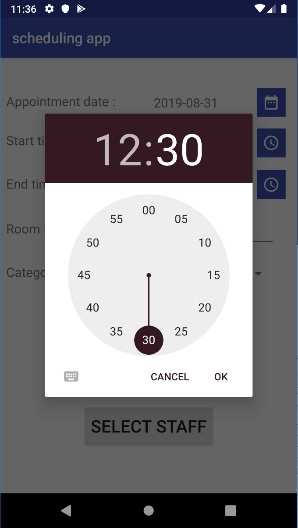
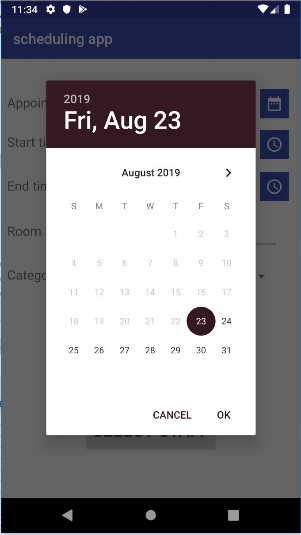
**Forgot password Forgot password-1**

When the user unable to login then they may forgot their password, the user can reset their password by using the forgot password page. In this page they can reset password by giving their Id and can create new password. After creating they can get password successfully created and can login again by their credentials.

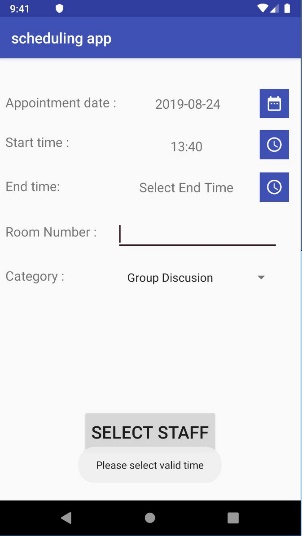
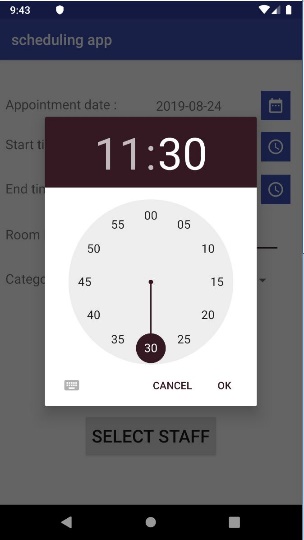


**Staff homepage**

The above screen tells about the staff home page. By which staff can do all the operations. The second screen explains about to set/make a appointment with another staff including some required fields.

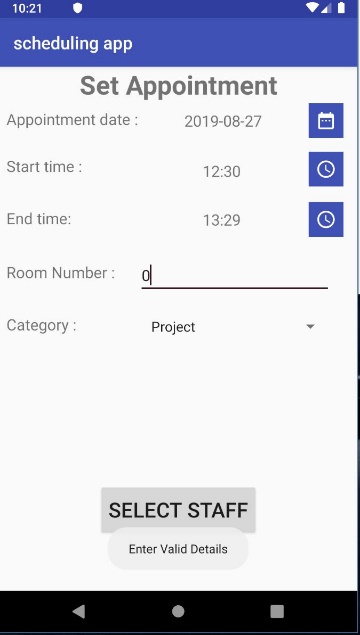
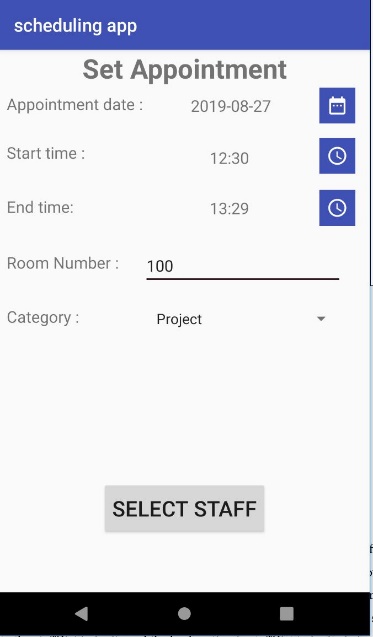
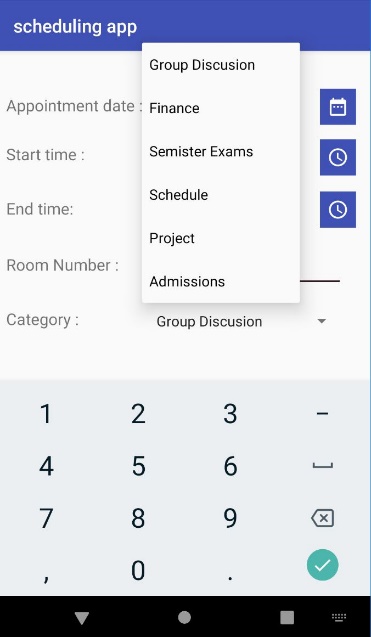


**Set Appointments Set starttime-1**



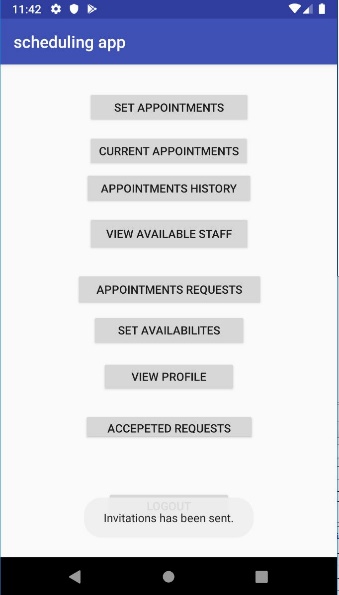
**Set End Time Invalid page**

The above screens describes the set appointments pages. The first screen shows how the staff can make an appointment by having date, day, start and end time, room number. By this he/she can make/set an appointment. The second screen shows that how to select date through calendar, the following is to select start time and end time by the user. The last page shows that please select valid time because if the user enters end time less than the start time then it appears a toast message.

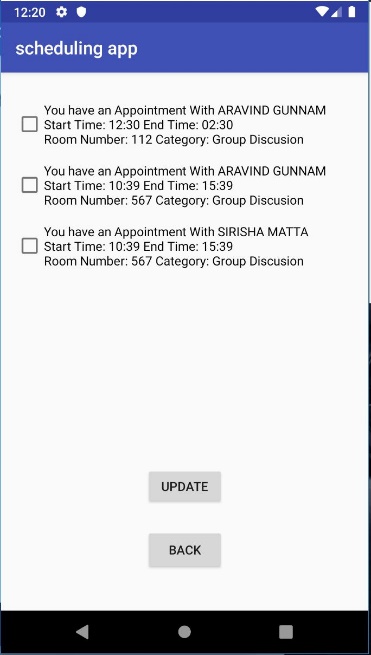
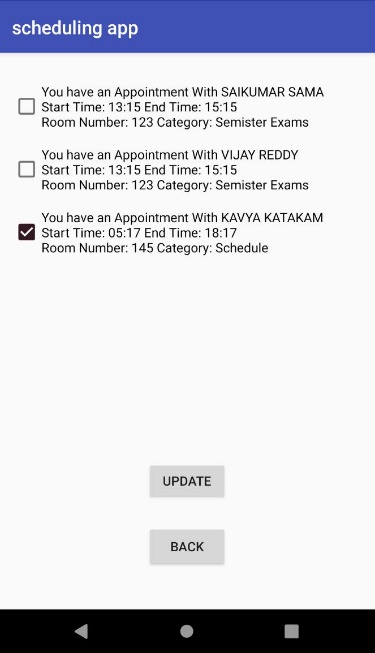
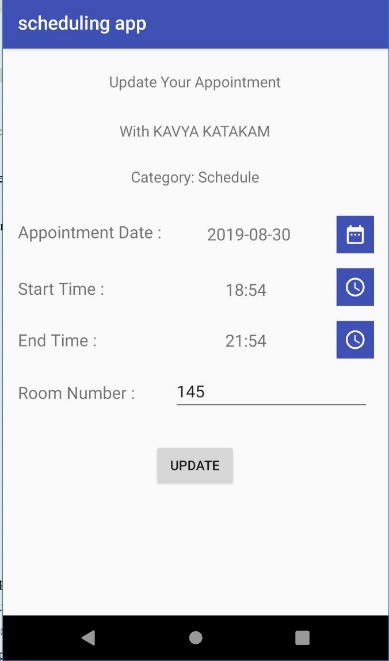
**Invalid page-2 set appointment -2 set appointment page**

The above screens explains, when the user gives room no as 0 then it doesn’t accept that number and it appears a toast message as enter valid details. Then immediately he/she should able to enter more than 0 but not 0. The second screen is to select the category type from the given list. The third screen shows the appointment details and when he/she is done with making an appointment then he/she can select staff to invite for their created appointment.



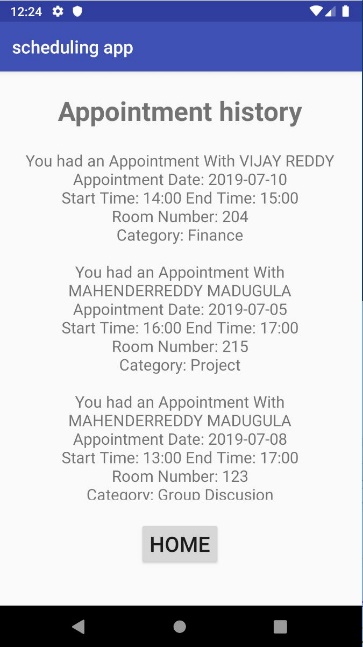
**Staff list invitation successful**

After making an appointment by one particular staff then he/she invites another staff which he/she wants to be communicate is shown in staff list to invite by staff names and check box to select the staff member and send invitation. The second screen shows the pop-up message to one who invites another staff is just to confirm that the invitation has been sent. The back option for second screen is to go back to the staff list to invite and the back option in staff list to invite is to go back to the set appointments page.

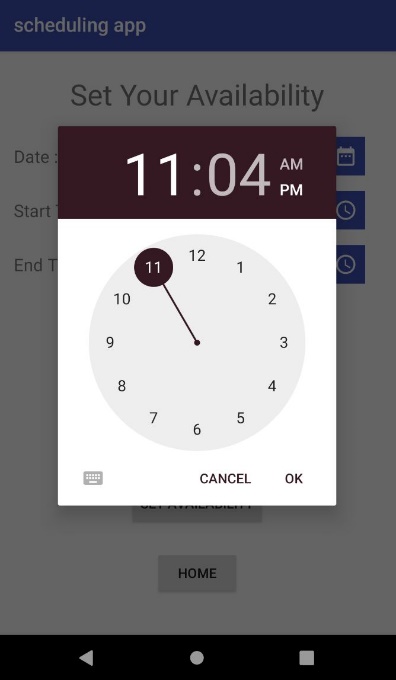
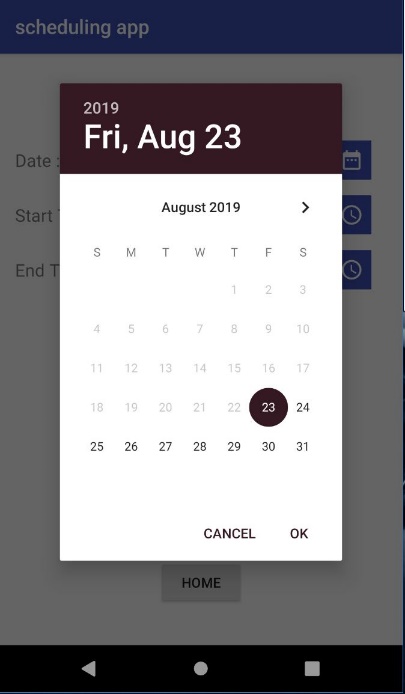
**Current appointments to update**

The above screen shows the current appointments and also the future appointments which he/she is going to attend. The appointments are followed by checkbox because there are number of appointments so, that the staff can select one or more than. From current appointments page there is update options. If the particular staff wants to update any of the appointments he/she should select one of the appointment and can make changes as if they wants. After the changes are done he/she can view the staff list to whom they have to make changes and send invitation then they can receive a pop-up message. Here the back goes to the current appointment page.

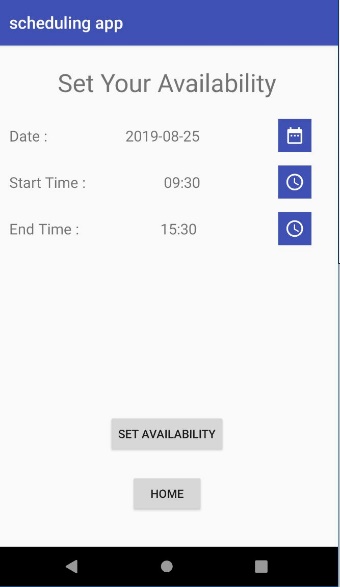
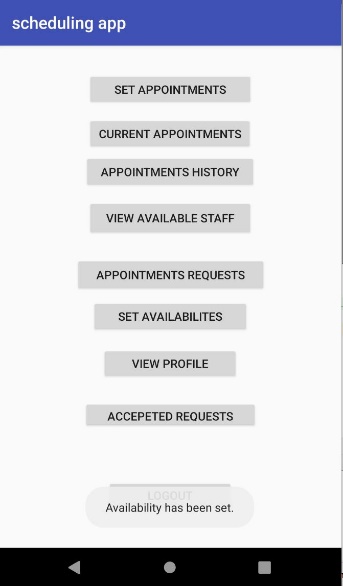


**Appointment History**

When the staff wants to see their appointments they can view their appointments by history appointments. In this they can view with whom they have done the appointment and with whom they are having the appointment. Home goes to the staff home page.

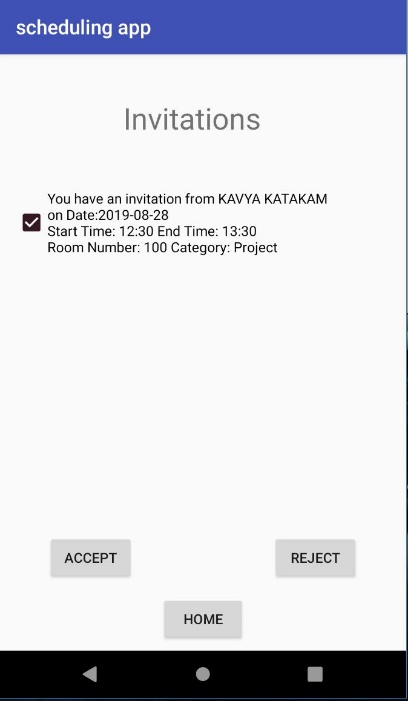
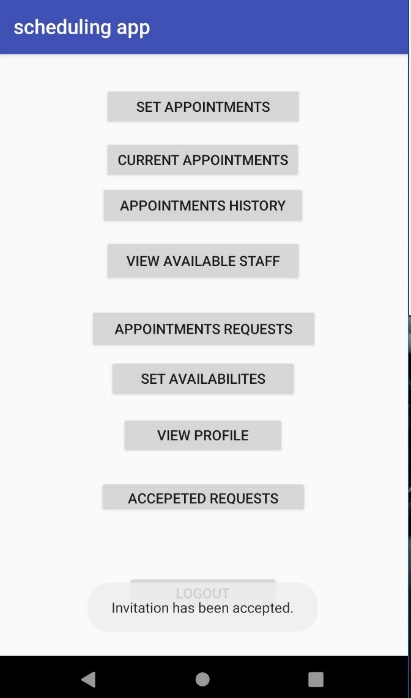


**Set Availability Set Availability -1**

**Set Availability-3 Successful page**

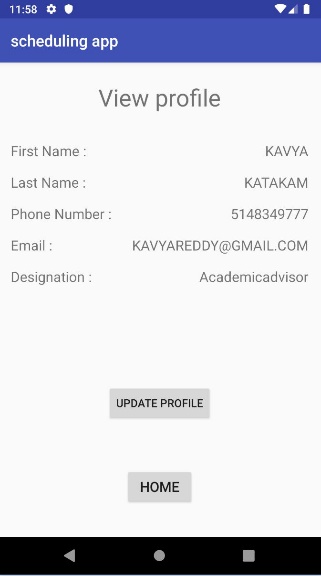
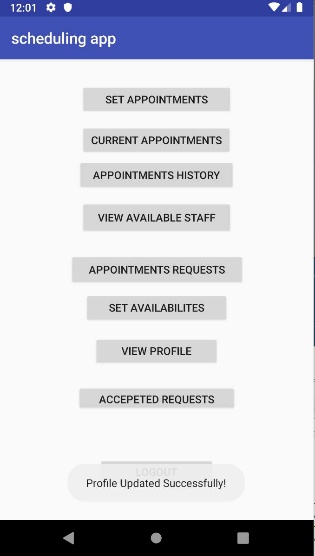
The staff has to set their availabilities, so that they has to set their availabilities. The first screen shows how to set the availability by date, time and day. When they are successfully created availability they receive a pop-up message.

**View staff availability Appointment requests Successful page**

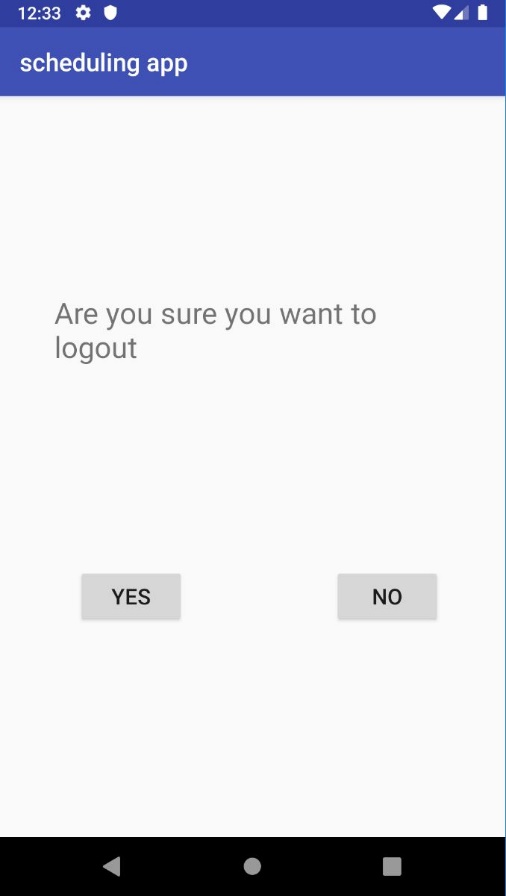
When on staff wants to know the availability of another staff they can check availabilities by view staff which is in home page. So that they can know all staff availabilities including with their name, day, date, start time and end time.

When on staff sends an appointment request/invitation to another staff the one who receives invitation can see their requests in appointment requests option. One staff can receive many invitations by another staff member then he/she can select any one of those then they can accept or decline the request depends on their interest. When they accepts the requests they receive pop-up message the same way when the user rejects they receive pop-up-message.

**Accepted requests View profile updated page**

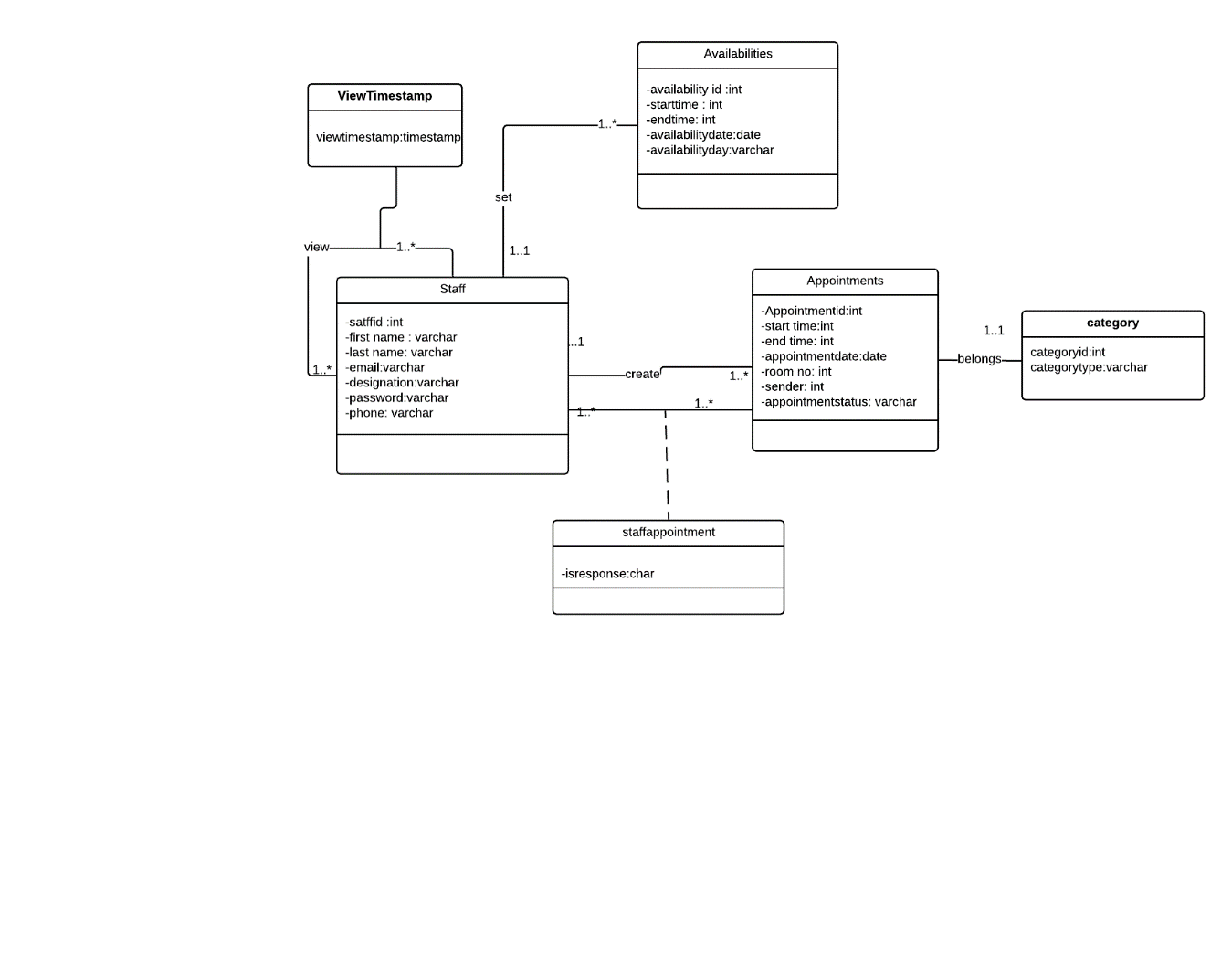
In the first screen the staff member who sent request can see the created appointment has been requested to many staff with names, time date. The second screen shows the profile of the staff with first name, last name, phone, email, designation. The third screen displays the successful message.



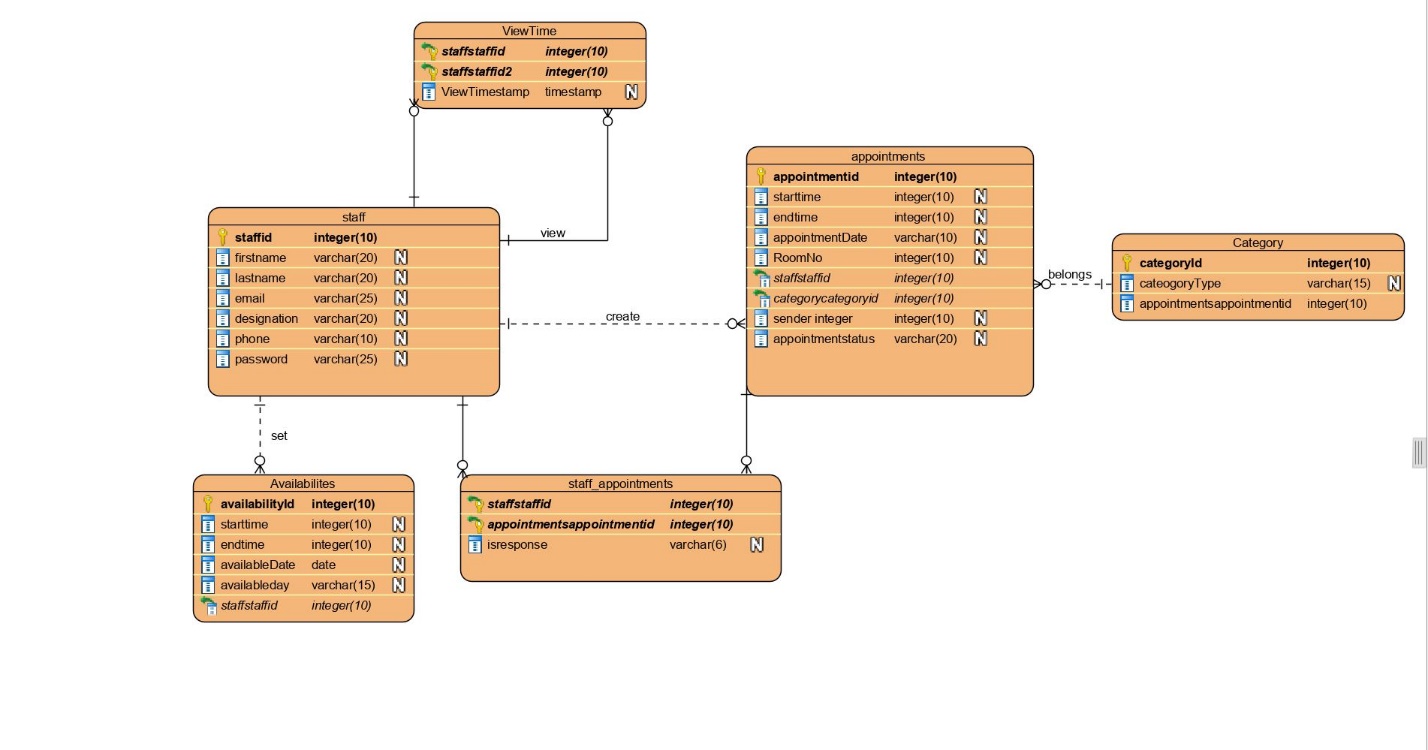
**Logout page**

When the user clicks login after his operations are done then the confirmation screen displays with are you sure want to quit? Follows by Y/N options. If the user clicks yes then the login page opens where as if the user clicks no the user goes back to the last operation done by the user.

### Class Diagram:



## Entity Relation Diagram:



### Web Services:

SELECT \* FROM STAFF where STAFFID=" + "'" + STAFFID + "'" + "AND PASSWORD=" + "'" + PASSWORD + "';

|  |  |  |  |
| --- | --- | --- | --- |
| URL | <http://localhost:8080/FinalProject/mad306/team3/login&179000&kavya5595> | | |
| Call | Login | | |
| Parameters | StaffId:179000&kavya5595 | | |
| Response | {  "Status":"OK",  "Timestamp": 1562813394213,  “message”:”exists”,  “Staffid”:”179000”,  }  Example 2  {  "Status":"Error",  "Timestamp": 1566622201644,  "Message " : " Invalid details "  } | | |
| Keys | “Status” | Status of the response  Possible values :“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| “Staffid" | The id of the staff | Mandatory |
| "Password" | The password of the Staff | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”Invalid details ” | Mandatory |

UPDATE STAFF SET PASSWORD='"+password+"' WHERE STAFFID ="+staffid;

|  |  |  |  |
| --- | --- | --- | --- |
| URL | http://localhost:8080/FinalProject/mad306/team3/forgetpassword&1794805&Mahi007 | | |
| Call | Forgot password | | |
| Parameters | StaffId:1794805&Mahi007 | | |
| Response | {  "Status":"OK",  "Timestamp": 1562813394213,  “message”:”exists”,  }  Example 2  {  "Status":"Error",  "Timestamp": 1566622201644,  "Message " : " Id doesn’t exists"  } | | |
| Keys | “Status” | Status of the response  Possible values :“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| “Staffid" | The id of the staff | Mandatory |
| "Password" | The password of the Staff | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”Id doesn’t exists” | Mandatory |

INSERT INTO APPOINTMENTS (APPOINTMENTID,STARTTIME,ENDTIME,APPOINTMENTDATE,ROOMNO,STAFFSTAFFID,CATEGORYCATEGORYID) VALUES(" +num\_of\_records+","+STARTTIME+","+ENDTIME+", DATE '"+APPOINTMENTDATE+"',"+ROOMNO+","+staffID+","+CATEGORYCATEGORYID+")";

|  |  |  |  |
| --- | --- | --- | --- |
| URL | <http://localhost:8080/FinalProject/mad306/team3/setappointment&17&1200&1400&2019-08-10&1&1794805&2> | | |
| Call | SetAppointment | | |
| Parameters | AppointmentId&starttime&endtime&AppointmentDate&RoomNo&StaffId&CategoryCategoryId | | |
| Response | {  "Status":"OK",  "Timestamp": 1562813394213,  “message”:”successfull”,  }Example 2  {  "Status":"Error",  "Timestamp":1562813435274,  "Message " : " Invalid details"  } | | |
| Keys | “Status” | Status of the response “OK”,“WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| "StaffId" | The id of the Staff | Mandatory |
|  | “starttime” | The starttime of the appointment | Mandatory |
|  | “endtime” | The endtime of the appointment | Mandatory |
|  | “AppointmentDate” | The date of the appointment | Mandatory |
|  | “AppointmentDay” | The day of the appointment | Mandatory |
|  | “RoomNo” | The roomnumber for the appointment | Mandatory |
|  | “CategoryCategoryId” | The categoryid for the appointment | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”StaffId Does not exist” | Mandatory |

SELECT \* FROM (SELECT \* FROM APPOINTMENTS WHERE SENDER="+senderId+" AND APPOINTMENTSTATUS='PENDING' AND APPOINTMENTDATE>= DATE '"+appointmentdate+"') WHERE ROWNUM<=3"

|  |  |  |  |
| --- | --- | --- | --- |
| URL | http://localhost:8080/FinalProject/mad306/team3/currentappointmet&1794805&2019-08-24 | | |
| Call | CurrentAppointment | | |
| Parameters | StaffId&appointemntdate | | |
| Response | {  "Status":"OK",  "Timestamp": 1566669698667,  “StaffId”:172602,  “aptId”:15,  “startTime”:1315,  “endTime”:1515,  “AppointmentDate”:”2019-08-24”,  “roomNumber”:123  “Category”:”Semister Exams”,  “StaffName”:”SAIKUMAR SAMA”  “message”:”Successfull”  }  {  "Status":"OK",  "Timestamp": 1566669698667,  “StaffId”:178000,  “aptId”:16,  “startTime”:1315,  “endTime”:1515,  “AppointmentDate”:”2019-08-24”,  “roomNumber”:123,  “Category”:”Semister Exams”,  “StaffName”:”VIJAY REDDY”,  “message”:”Successfull  }  {  "Status":"OK",  "Timestamp": 1566669698667,  “StaffId”:179000,  “aptId”:22,  “startTime”:1315,  “endTime”:1515,  “AppointmentDate”:”2019-08-24”,  “roomNumber”:123,  “Category”:”Semister Exams”,  “StaffName”:”KAVYAREDDY ”,  “message”:”Successful”  }  Example 2  {  "Status":"Error",  "Timestamp": 1566622201644,  "Message " : " Id doesn’t exists"  } | | |
| Keys | “Status” | Status of the response  Possible values :“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| “Staffid" | The id of the staff | Mandatory |
| "Password" | The password of the Staff | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”Id doesn’t exists” | Mandatory |
|  | “ AppointmentDate” | The appointment date | Mandatory |
|  | “starttime” | The starttime of apoointment | Mandatory |
|  | “endtime” | The endtime of appointment | Mandatory |
|  | “roomNumber” | The roomnumber for appointment | Mandatory |
|  | “Category” | The appointment type | Mandtory |
|  | “StaffName” | The name of the staff | Mandatory |

UPDATE APPOINTMENTS SET STARTTIME="+startTime+", ENDTIME="+endTime+", APPOINTMENTDATE= DATE '"+aptDate+"', ROOMNO="+roomNum+", APPOINTMENTSTATUS='PENDING' WHERE APPOINTMENTID="+aptId

|  |  |  |  |
| --- | --- | --- | --- |
| URL | http://localhost:8080/FinalProject/mad306/team3/updateappointmentwithStaff&23&2019-08-31&1230&1430&125 | | |
| Call | UpdateAppointmentwithstaff | | |
| Parameters | AppointmentId&AppointmentDate&starttime&endtime&Roomno | | |
| Response | {  "Status":"OK",  "Timestamp": 1562813394213,  “message”:”exists”,  }  Example 2  {  "Status":"Error",  "Timestamp": 1566622201644,  "Message " : " Id doesn’t exists"  } | | |
| Keys | “Status” | Status of the response  Possible values :“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| “Staffid" | The id of the staff | Mandatory |
| "AppointmentId" | The id of the appointment | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”Id doesn’t exists” | Mandatory |
|  | “appointmentdate” | The date for the appointment | Mandatory |
|  | “starttime” | The start time for the appointment | Mandatory |
|  | “endtime” | The endtime for the appointment | Mandatory |
|  | “roomno” | The roomnumber for the appointment | Mandatory |

SELECT \* FROM APPOINTMENTS WHERE SENDER="+senderId+" AND APPOINTMENTDATE <= DATE '"+appointmentdate+"'";

|  |  |  |  |
| --- | --- | --- | --- |
| URL | http://localhost:8080/FinalProject/mad306/team3/appointmenthistory&1794805&2019-08-31 | | |
| Call | Appointmenthistory | | |
| Parameters | StaffId&appointemntdate:1794805&2019-08-31 | | |
| Response | {  "Status":"OK",  "Timestamp": 1566669698667,  “StaffId”:1794278,  “aptId”:3,  “startTime”:1100,  “endTime”:1300  “AppointmentDate”:”2019-06-23”,  “roomNumber”:214,  “Category”:”Schedule”,  “StaffName”:”ARAVIND GUNNAM”  “aptStatus”:”ACCEPTED”,  “message”:”Successfull”  }  {  "Status":"OK",  "Timestamp": 1566669698667,  “StaffId”:172602,  “aptId”:15,  “startTime”:1315,  “endTime”:1515,  “AppointmentDate”:”2019-08-24”,  “roomNumber”:123,  “Category”:”Semister Exams”,  “StaffName”:”SAIKUMAR SAMA ”,  “aptStatus”:”PENDING”,  “message”:”Successfull”  }  {  "Status":"OK",  "Timestamp": 1566669698667,  “StaffId”:178000,  “aptId”:15,  “startTime”:1315,  “endTime”:1515,  “AppointmentDate”:”2019-08-24”,  “roomNumber”:123,  “Category”:”Semister Exams”,  “StaffName”:”VIJAY REDDY”,  “aptStatus”:”PENDING”,  “message”:”Successfull  }  {  "Status":"OK",  "Timestamp": 1566669698667,  “StaffId”:179000,  “aptId”:15,  “startTime”:1315,  “endTime”:1515,  “AppointmentDate”:”2019-08-24”,  “roomNumber”:123,  “Category”:”Semister Exams”,  “StaffName”:”KAVYAREDDY ”,  “aptStatus”:”ACCEPTED”,  “message”:”Successful”  }  Example 2  {  "Status":"Error",  "Timestamp": 1566622201644,  "Message " : " Id doesn’t exists"  } | | |
| Keys | “Status” | Status of the response  Possible values :“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| “Staffid" | The id of the staff | Mandatory |
| "Password" | The password of the Staff | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”Id doesn’t exists” | Mandatory |
|  | “ AppointmentDate” | The appointment date | Mandatory |
|  | “starttime” | The starttime of apoointment | Mandatory |
|  | “endtime” | The endtime of appointment | Mandatory |
|  | “roomNumber” | The roomnumber for appointment | Mandatory |
|  | “Category” | The appointment type | Mandtory |
|  | “StaffName” | The name of the staff | Mandatory |
|  | “aptStatus” | The status of appointment | Mantaory |

INSERT INTO AVAILABILITES (AVAILABILITYID,STARTTIME,ENDTIME,AVAILABILEDATE,AVAILABILEDAY,STAFFSTAFFID) VALUES("+ num\_of\_records+","+STARTTIME+","+ENDTIME+",DATE '"+AVAILABILEDATE+"','"+AVAILABILEDAY+"',"+SENDERID+")";

|  |  |  |  |
| --- | --- | --- | --- |
| URL | <http://localhost:8080/FinalProject/mad306/team3/setAvilability&1794805&1100&100&2019-10-12&tuesday> | | |
| Call | SetAvailability | | |
| Parameters | StaffId&starttime&endtime&AvailableDate&AvailableDay | | |
| Response | {  "Status":"OK",  "Timestamp": 1562813394213,  “message”:”successfull”,  }  Example 2  {  "Status":"Error",  "Timestamp":1562813435274,  "Message " : " Invalid details"  } | | |
| Keys | “Status” | Status of the response  Possible values:“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| "StaffId" | The id of the Staff | Mandatory |
|  | “starttime” | The starttime of the availability | Mandatory |
|  | “endtime” | The endtime of the availability | Mandatory |
|  | “AvailableDate” | The date of the available | Mandatory |
|  | “AvailableDay” | The day of the available | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”StaffId Does not exist” | Mandatory |

SELECT \* FROM (SELECT \* FROM AVAILABILITES WHERE STAFFSTAFFID!="+staffid+" AND AVAILABILEDATE >= DATE '"+currentDate+"') WHERE ROWNUM<=7"

|  |  |  |  |
| --- | --- | --- | --- |
| URL | http://localhost:8080/FinalProject/mad306/team3/getavailability&1791234&2019-08-24 | | |
| Call | getavailability | | |
| Parameters | StaffId&AppointmentDate | | |
| Response | {  "Status":"OK",  "Timestamp": 1562813394213,  “Staffid”:”1794805”,  “startTime”:1200,  “endTime”:2300,  “ADate”:”2019-08-24”,  “ADay”:”SATURDAY”,  “StaffName”:”MAHENDERREDDY MADUGULA”  “message”:”successfull”,  }  {  "Status":"OK",  "Timestamp": 1562813394213,  “Staffid”:”179000”,  “startTime”:1415,  “endTime”:1515,  “ADate”:”2019-08-31”,  “ADay”:”SATURDAY”,  “StaffName”:”KAVYA KATAKAM  “message”:”successfull”,  }  {  "Status":"OK",  "Timestamp": 1562813394213,  “Staffid”:”179000”,  “startTime”:1430,  “endTime”:1800,  “ADate”:”2019-08-29”,  “ADay”:”THURSDAY”,  “StaffName”:”KAVYA KATAKAM”  “message”:”successfull”,  }  {  "Status":"OK",  "Timestamp": 1562813394213,  “Staffid”:”1794278”,  “startTime”:1130,  “endTime”:1530,  “ADate”:”2019-09-19”,  “ADay”:”THURSDAY”,  “StaffName”:”ARAVIND GUNNAM”  “message”:”successfull”,  }  {  "Status":"OK",  "Timestamp": 1562813394213,  “Staffid”:”1794278”,  “startTime”:1000,  “endTime”:1800,  “ADate”:”2019-08-30”,  “ADay”:”FRIDAY”,  “StaffName”:”ARAVIND GUNNAM”  “message”:”successfull”,  }  {  "Status":"OK",  "Timestamp": 1562813394213,  “Staffid”:”172602”,  “startTime”:1400,  “endTime”:1630,  “ADate”:”2019-08-30”,  “ADay”:”FRIDAY”,  “StaffName”:”SAIKUMAR SAMA”  “message”:”successfull”,  }  {  "Status":"OK",  "Timestamp": 1562813394213,  “Staffid”:”1795598”,  “startTime”:930,  “endTime”:1130,  “ADate”:”2019-08-31”,  “ADay”:”SATURDAY”,  “StaffName”:”SIRISHA MATTA”  “message”:”successfull”,  }  Example 2  {  "Status":"Error",  "Timestamp":1562813435274,  “staffId”:”1791234”,  "Message " : " Does not exist "  } | | |
| Keys | “Status” | Status of the response  Possible values :“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| “staffid" | The id of the Staff | Mandatory |
| "Appointmentdate" | The date for the appointment | mandatory |
| "starttime" | The starttime of the appointment | Mandatory |
| “endtime” | The endtime of the appointment | Mandatory |
|  | “ADate” | The available date of the staff | Mandatory |
|  | “ADay” | The available day of the staff | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”SaffId Does not exist” | Optional |
|  | “staffname” | The name of the staff | Mandatory |

SELECT \* FROM (SELECT \* FROM APPOINTMENTS WHERE STAFFSTAFFID="+staffid+" AND APPOINTMENTSTATUS='PENDING') WHERE ROWNUM<=3;

|  |  |
| --- | --- |
| URL | http://localhost:8080/FinalProject/mad306/team3/getappointments&178000 |
| Call | getappointments |
| Parameters | StaffId:178000 |
| Response | {  "Status":"OK",  "Timestamp": 1566669698667,  “StaffId”:178000,  “startTime”:1315,  “endTime”:1515,  “AppointmentDate”:”2019-08-24”,  “roomNumber”:123,  “Category”:”Semister Exams”,  “SenderId”:1794805,  “SenderName”:”MAHENDER REDDY”,  “message”:”Successfull  }  }  Example 2  {  "Status":"Error",  "Timestamp":1562813435274,  "Message " : " Invalid details"  } |

|  |  |  |  |
| --- | --- | --- | --- |
| Keys | “Status” | Status of the response  Possible values :“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| “Staffid" | The id of the staff | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”Id doesn’t exists” | Mandatory |
|  | “ AppointmentDate” | The appointment date | Mandatory |
|  | “starttime” | The starttime of apoointment | Mandatory |
|  | “endtime” | The endtime of appointment | Mandatory |
|  | “roomNumber” | The roomnumber for appointment | Mandatory |
|  | “Category” | The appointment type | Mandatory |
|  | “SenderName” | The name of the staff who sends appointments | Mandatory |
|  | “SenderId” | The id of sender | Mandatory |

SELECT \* FROM (SELECT \* FROM APPOINTMENTS WHERE SENDER="+senderID+" AND APPOINTMENTSTATUS='ACCEPTED') WHERE ROWNUM<=3";

|  |  |  |  |
| --- | --- | --- | --- |
| URL | http://localhost:8080/FinalProject/mad306/team3/acceptedAppointments&1794805 | | |
| Call | acceptedAppointments | | |
| Parameters | StaffId | | |
| Response | {  "Status":"OK",  "Timestamp": 1562813394213,  “SatffId”:1794278,  “startTime”:1100,  “endTime”:1300,  “AppointmentDate”:”2019-06-23”,  “roomnumber”:214,  “Category”:”Schedule”,  “Sender”:1794805,  “StaffName”:”ARAVIND GUNNAM”  “message”:”successfull”,  }  {  "Status":"OK",  "Timestamp": 1562813394213,  “SatffId”:179000,  “startTime”:1315,  “endTime”:1515,  “AppointmentDate”:”2019-08-24”,  “roomnumber”:123  “Category”:”Semsiter Exams”,  “Sender”:1794805,  “StaffName”:”KAVYA KATAKAM”  “message”:”successfull”,  }  Example 2{  "Status":"Error",  "Timestamp":1562813435274,  "Message " : " Invalid details"  } | | |
| Keys | “Status” | Status of the response  Possible values:“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| "StaffId" | The id of the Staff | Mandatory |
|  | “starttime” | The starttime of the appointment | Mandatory |
|  | “endtime” | The endtime of the appointment | Mandatory |
|  | “AppointmentDate” | The date of the appointment | Mandatory |
|  | “AppointmentDay” | The day of the appointment | Mandatory |
|  | “RoomNo” | The roomnumber for the appointment | Mandatory |
|  | “Category” | The categoryid for the appointment | Mandatory |
|  | “sender” | The id of the appointment sender | Mandatory |
|  | “StaffName” | The name of the staff | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”StaffId Does not exist” | Mandatory |

SELECT \* FROM STAFF WHERE STAFFID= 178000;

|  |  |  |  |
| --- | --- | --- | --- |
| URL | <http://www.localhost:8080/FinalProject/mad306/team3/SingleStaff&178000> | | |
| Call | StaffProfile | | |
| Parameters | StaffId:178000 | | |
| Response | {  "Status":"OK",  "Timestamp": 1562813394213,  “message”:”exists”,  “Staffid”:”178000”,  "Firstname":"VIJAY",  "Lastname":"REDDY",  “email”: “REDDYVIJAY007@GMAIL.COM”  "designation": “DIRECTOR”,  "phone":5148349145  }  Example 2  {  "Status":"Error",  "Timestamp":1562813435274,  “staffed”:”179000”,  "Message " : " Does not exist "  } | | |
| Keys | “Status” | Status of the response  Possible values :“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| “Firstname" | The first name of the Staff | Optional |
| "Lastname" | The last name of the Staff | Optional |
| "email" | The email of the Staff | Optional |
| “designation” | The designation of the Staff | Optional |
|  | “phone” | The phone of the Staff | Optional |
|  | “message” | Message of the response possible values: “successful”,”SaffId Does not exist” | Optional |
|  | “staffid” | The id of the staff | Mandatory |

UPDATE STAFF SET EMAIL='"+EMAIL+"', PHONE="+PHONE+" WHERE STAFFID="+staffId;

|  |  |  |  |
| --- | --- | --- | --- |
| URL | http://localhost:8080/FinalProject/mad306/team3/updateprofile&1794805&mahender@yahoo.com&514678901 | | |
| Call | UpdateProfile | | |
| Parameters | StaffId:1794805&mahender@yahoo.com&514678901 | | |
| Response | {  "Status":"OK",  "Timestamp": 1562813394213,  “message”:”successfull”,  }  Example 2  {  "Status":"Error",  "Timestamp":1562813435274,  "Message " : " Invalid details"  } | | |
| Keys | “Status” | Status of the response  Possible values :“OK”, “WRONG”,“ERROR” | Mandatory |
| “Timestamp” | Epoch UTC format response timestamp | Mandatory |
| "email" | The email of the Staff | Mandatory |
|  | “phone” | The phonenumber of the Staff | Mandatory |
|  | “message” | Message of the response possible values: “successful”,”SaffId Does not exist” | Mandatory |
|  | “staffid” | The id of the staff | Mandatory |

### SQL Queries:

Creation:

**STAFF:**

Create table STAFF (staffId integer Primary Key not null,

FirstName varchar (20) not null,

LastName varchar (20) not null,

Email varchar (25),

Designation varchar (20),

Phone varchar (10));

**AVAILABILITES:**

Create table AVAILABILITES (availabilityId integer Primary Key ,

StartTime INTEGER not null,

EndTime INTEGER not null,

availableDate date,

availableDay varchar(15),

staffstaffid integer,

foreignkey (staffstaffid) references staff(staffid));

**APPOINTMENTS:**

Create table APPOINTMENTS (APPOINTMENTID int PRIMARY KEY not null,

Starttime int,

Endtime int,

Appointmentdate date,

Appointmentday varchar (15),

RoomNo int,

STAFFSTAFFID int,

CATEGORYCATEGORYID int, sender int, appointment status varchar(20),

CONSTRAINT FK\_STAFF FOREIGN KEY (STAFFSTAFFID) REFERENCES STAFF (STAFFID),

CONSTRAINT FK\_CATEGORY FOREIGN KEY (CATEGORYCATEGORYID) REFERENCES CATEGORY (CATEGORYID));

**CATEGORY:**

CREATE TABLE CATEGORY (CATEGORYID INT PRIMARY KEY not null,

CATEGORYTYPE VARCHAR (15) not null);

**Staff Appointments:**

Create table STAFFAPPOINTMENTS (STAFFSTAFFID integer, APPOINTMENTSAPPOINTMENTID integer, ISRESPONSE CHAR (1),

CHECK (ISRESPONSE IN ('Y','N')), constraint PK\_STAFFAPPOINTMENTS primary key (STAFFSTAFFID, APPOINTMENTSAPPOINTMENTID),

Foreign key (STAFFSTAFFID) references STAFF (STAFFID),

Foreign key (APPOINTMENTSAPPOINTMENTID) references APPOINTMENTS (APPOINTMENTID));

**VIEWTIME**

Create table VIEWTIME (STAFFSTAFFID integer, STAFFSTAFFID2 integer, VIEWTIMESTAMP TIMESTAMP,

Constraint PK\_VIEWTIME primary key (STAFFSTAFFID, STAFFSTAFFID2));

**Insertion:**

**STAFF:**

INSERT INTO STAFF(STAFFID,FIRSTNAME,LASTNAME,EMAIL,DESIGNATION,PHONE)

VALUES(1794805,'MAHENDERREDDY','MADUGULA','MAHENDERMADUGULA@GMAIL.COM','ACCOUNTANT','5145231545');

(172602,'SAIKUMAR,'SAMA',’SAMASAI@GMAIL.COM','MANAGER','5146495552');

(1794278,'ARAVIND','GUNNMA','GUNNAMARVIND@GMAIL.COM','MANAGER','4383417410');

(1795598,'SIRISHA','MATTA’,'SIRISHAMATTA@GMAIL.COM','INSTRUCTOR','5148349969');

(1791234,'JAIRAM','NAYANI','BALRAJNAYANI194@GMAIL.COM','PROFESSOR','5148349897');

(179000,'KATAKAM','KAVYA',KAVYAREDDY@GMAIL.COM,'PROFESSOR','5148349970');

(178000,'VIJAY','REDDY','REDDYVIJAY007@GMAIL.COM'.'DIRECTOR','5148349145');

**AVAILABILITIES:**

Insert into AVAILABILITES(AVAILABILITYID,STARTTIME,ENDTIME,AVAILABILEDATE,AVAILABILEDAY,STAFFSTAFFID)

values ('1','1200','1300',TO\_DATE('2019/07/24','yyyy/mm/dd'),'MONDAY','179602');

Insert into AVAILABILITES(AVAILABILITYID,STARTTIME,ENDTIME,AVAILABILEDATE,AVAILABILEDAY,STAFFSTAFFID)

values ('2','1000','1200',TO\_DATE('2019/08/02','yyyy/mm/dd'),'TUESDAY','1794278');

Insert into AVAILABILITES(AVAILABILITYID,STARTTIME,ENDTIME,AVAILABILEDATE,AVAILABILEDAY,STAFFSTAFFID)

values ('3','1500','1700',TO\_DATE('2019/07/06','yyyy/mm/dd'),'THURSDAY','1791234');

Insert into AVAILABILITES(AVAILABILITYID,STARTTIME,ENDTIME,AVAILABILEDATE,AVAILABILEDAY,STAFFSTAFFID)

values ('4','1100','1400',TO\_DATE('2019/08/12','yyyy/mm/dd'),'FRIDAY','178000');

Insert into AVAILABILITES(AVAILABILITYID,STARTTIME,ENDTIME,AVAILABILEDATE,AVAILABILEDAY,STAFFSTAFFID) values ('5','1400','1600',TO\_DATE('2019/07/27','yyyy/mm/dd'),'WEDNESDAY','1794278');

Insert into AVAILABILITES(AVAILABILITYID,STARTTIME,ENDTIME,AVAILABILEDATE,AVAILABILEDAY,STAFFSTAFFID) values ('6','8300','1100',TO\_DATE('2019/08/05','yyyy/mm/dd'),'FRIDAY','179000');

Insert into AVAILABILITES(AVAILABILITYID,STARTTIME,ENDTIME,AVAILABILEDATE,AVAILABILEDAY,STAFFSTAFFID) values ('7','1200','1500',TO\_DATE('2019/08/06','yyyy/mm/dd'),'SATURDAY','1795598');

Insert into AVAILABILITES(AVAILABILITYID,STARTTIME,ENDTIME,AVAILABILEDATE,AVAILABILEDAY,STAFFSTAFFID) values ('8','1400','1530',TO\_DATE('2019/07/10','yyyy/mm/dd'),'WEDNESDAY','1794278');

Insert into AVAILABILITES(AVAILABILITYID,STARTTIME,ENDTIME,AVAILABILEDATE,AVAILABILEDAY,STAFFSTAFFID) values ('9','1030','1230',TO\_DATE('2019/07/14','yyyy/mm/dd'),'SUNDAY','178000');

**CATEGORY:**

INSERT INTO CATEGORY (CATEGORYID, CATEGORYTYPE)

VALUES (1,'GROUPDISCUSSION');

INSERT INTO CATEGORY (CATEGORYID, CATEGORYTYPE)

VALUES (2,'FINANCE');

INSERT INTO CATEGORY (CATEGORYID, CATEGORYTYPE)

VALUES (3,'SEMSTEREXAMS');

INSERT INTO CATEGORY (CATEGORYID, CATEGORYTYPE)

VALUES (4,'SCHEDULE');

INSERT INTO CATEGORY (CATEGORYID, CATEGORYTYPE)

VALUES (5,'PROJECT');

INSERT INTO CATEGORY (CATEGORYID, CATEGORYTYPE)

VALUES (6,'ADMISSIONS');

**STAFFAPPOINTMENTS:**

INSERT INTO STAFFAPPOINTMENTS (STAFFSTAFFID, APPOINTMENTSAPPOINTMENTID, ISRESPONSE) VALUES ('1794805','1','Y');

INSERT INTO STAFFAPPOINTMENTS (STAFFSTAFFID, APPOINTMENTSAPPOINTMENTID, ISRESPONSE) VALUES ('172602','3','N');

INSERT INTO STAFFAPPOINTMENTS (STAFFSTAFFID, APPOINTMENTSAPPOINTMENTID, ISRESPONSE) VALUES ('179000','2','Y');

INSERT INTO STAFFAPPOINTMENTS (STAFFSTAFFID, APPOINTMENTSAPPOINTMENTID, ISRESPONSE) VALUES ('178000','3','N');

INSERT INTO STAFFAPPOINTMENTS (STAFFSTAFFID, APPOINTMENTSAPPOINTMENTID, ISRESPONSE) VALUES ('1791234','4','N');

INSERT INTO STAFFAPPOINTMENTS (STAFFSTAFFID, APPOINTMENTSAPPOINTMENTID, ISRESPONSE) VALUES ('1794278','6','Y');

INSERT INTO STAFFAPPOINTMENTS (STAFFSTAFFID, APPOINTMENTSAPPOINTMENTID, ISRESPONSE) VALUES ('1795598','4','Y');

APPOINTMENTS

INSERT INTO APPOINTMENTS(APPOINTMENTID,STARTTIME,ENDTIME,APPOINTMENTDATE,APPOINTMENTDAY,ROOMNO,STAFFSTAFFID,CATEGORYCATEGORYID)

VALUES(1,1200,1300,2019-05-03,MONDAY,202,1794805,1),

(2,1400,1500,2019-07-10,TUESDAY,204,178000,2),

(3,1100,1300,2019-06-23,FRIDAY,214,1794278,4),

(4,1500,1600,2019-06-28,WEDNESDAY,214,179000,5),

(5,1600,1700,2019-07-05,THURSDAY,215,172602,5),

(6,1600,1700,2019-07-05,THURSDAY,215,1794805,5),

(7,1200,1400,2019-07-08,TUESDAY,205,1791234,4);

VIEWTIME:

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1794805,179000,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1794805,178000,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1794805,1791234,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1791234,1794805,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1791234,172602,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1791234,179000,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1794805,179000,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1794805,179000,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1794805,179000,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1794805,179000,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1794805,179000,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1794805,179000,CURRENT\_TIMESTAMP);

INSERT INTO VIEWTIME(STAFFSTAFFID,STAFFSTAFFID2,VIEWTIMESTAMP)

VALUES(1794805,179000,CURRENT\_TIMESTAMP);

# SECTION-4

**DEVELOPMENT TEAM**

## 4.1. Development Team:

*A self-organizing, cross-functional team of people who collectively are responsible for all of the work necessary to produce working, and producing validating functions.*

### 4.1.1. Front End Developer:

*A front-end web developer is probably what most people think of as a “web developer”.**A front-end web developer is responsible for implementing visual elements that users see and interact with in a web application. They are usually supported by back-end web developers, who are responsible for server-side application logic and integration of the work front-end developers do.*

**Responsibilities:**

* *Develop new user-facing features*
* *Build reusable code and libraries for future use*
* *Ensure the technical feasibility of UI/UX designs*
* *Optimize application for maximum speed and scalability*
* *Assure that all user input is validated before submitting to back-end*
* *Collaborate with other team members and stakeholders*

**Background:**

* *HTML/CSS.*
* *JavaScript /jQuery*
* *Version control/Git*
* *Responsive Design*
* *Testing/debugging*
* *JavaScript ,MVC frameworks*
* *Browse developer tools.*
* *Strong written and verbal communication skills.*
* *Understanding of working in development and production environments.*

**Number of persons required: 4**

### 4.1.2. Back End Developer:

*A back-end Web developers are generally responsible for server-side web application logic and integration of the work front-end developers do. In addition, Back-end Web Developers usually write the web services as well as APIs used by front-end developers and mobile application developers*

**Responsibilities:**

* *Responsible for maintaining and develop all parts of a complex service-oriented system*
* *Work with a large arsenal of open source and cutting-edge technologies such as Node.js, Angular.js, MySQL and Docker.*
* *Plan and implement model data structures for scalability*
* *Write secure code*
* *Participate in all phases of the development life cycle*
* *Work on the maintenance and improvement of existing data structures*
* *Research, evaluate and analyze technical and design requirements*
* *Develop additional server level functionality and tools using a .NET language (C# preferred).*
* *Troubleshoot and fix bugs/flaws to ensure servers are running optimally.*
* *Assist with website strategy and planning.*
* *Formulate concepts and ideas for additional products, tools and services.*

**Background:**

* *Basic understanding of front-end technologies and platforms, such as JavaScript, HTML5, and CSS3*
* *Understanding accessibility and security compliance, “session management” in a distributed server environment****.***
* *User authentication and authorization between multiple systems, servers, and environments*
* *Integration of multiple data sources and databases into one system*
* *Data migration, transformation, and scripting, Setup and administration of backups*
* *Creating database schemas that represent and support business processes*
* *Implementing automated testing platforms and unit tests*
* *Proficient knowledge of a back-end programming language Depending on the specific case, a developer should have the knowledge of one or more of PHP, Python, Ruby, Java, .NET, JavaScript etc.*
* *Make sure to mention any other framework, libraries, or any other technology related to your development stack.*

**Numbers of persons required:** 3

### 4.1.3. Android Developer:

*An Android developer is in charge of developing applications for devices powered by the Android operating system. They have a strong understanding of the patterns and practices that revolve around such platform. Moreover, they ensure the best possible performance, quality, and responsiveness of the application*

**Roles and Responsibilities:**

* *Design and build Android mobile applications*
* *Collaborate with cross-functional teams to analyze, design, and ship new features*
* *Fully responsible for mobile app development in the team*
* *Build and publish applications in app stores*
* *Implement new technologies to maximize application performance*
* *Unit-test code for robustness, including edge cases, usability, and general reliability*
* *Work on bug fixing and improving application performance*

**Background:**

* *BS/MS degree in Computer Science, Engineering or a related subject*
* *Proven software development experience and Android skills development*
* *Proven working experience in Android app development and*
* *Have published at least one original Android app*
* *Experience with Android SDK*
* *Experience working with remote data via REST and JSO and Experience with third-party libraries and APIs*
* *Working knowledge of the general mobile landscape, architectures, trends, and emerging technologies*

**Numbers of persons required: 2**

* *The job of a software test manager is not an easy one.*
* *He has to recruit testing team and take responsibility for getting them trained.*
* *A software manager has to perform ongoing analysis of various testing processes and ensure that the testing team is carrying out all the processes correctly.*
* *This job is of great responsibility as the software testing manager is the one who selects, introduces and implement various tools for testing.*
* *A software test manager is responsible for finalizing templates for testing documents, test reports and other procedures.*

**Number of person required: 2**

### 4.1.6. Project Manager:

*This is the person who has the overall responsibilities for making decisions large and small. Every decision the project manager makes must directly benefit their project. Project manager can produce reports and compared with the several hours.*

**Responsibilities:**

* *Planning and Defining scope*
* *Activity Planning and sequencing*
* *Resource Planning*
* *Developing Schedules*
* *Time estimating*
* *Cost estimating*
* *Developing a budget*
* *Team leadership*
* *Business partnering*
* *Scalability, Portability Analysis.*

**Background:**

* *Experience with process improvement and inventory control.*
* *PMP certification*
* *Advanced time management and analytical skills.*
* *Minimum of bachelor’s degree in management business*.

**Number of persons required:** 1

### 4.1 Project deliverables:

*A deliverable is a real or imagination good or service produced as a result of a project that is intended to be delivered to a customer (either internal or external). A deliverable could be a report, a document, a software product, a server upgrade or any other building block of an overall project.*

*There are five deliverables in a project. They are*

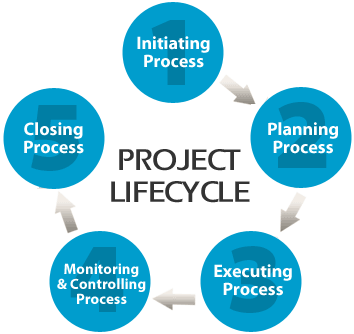
*1. Project Initiation*

*2. Project planning*

*3. Project execution*

*4. Project monitoring closing process*

*5. Project Closure*

****

**Fig 4.1: Cycle of project deliverables**

**1. Project Initiation:**

*The initiating phase is the first phase, where the project is kicked off, both with your team and with any clients and stakeholders. Any information you have (e.g. from the pitch or RFP (request for proposal) stage, from the client, from any background research) is gathered together in order to set and define the project’s scope, timings and cost. This is the core set up for your project where you identify the stakeholders, the team, goals and objectives and deliverables.*

*Once the recommended solution is approved, a project is initiated to deliver the approved solution and a project manager is appointed. The major deliverables and the participating work groups are identified, and the project team begins to take shape. Approval is then sought by the project manager to move onto the detailed planning phase*.

*In project initiation the manager needs to have team plan, how to build the correct team to success a project. The manager have to check whether having skills, stake holders, experience, budget.*

**2. Project Planning:**

*The next phase, the planning phase, is where the project solution is further developed in as much detail as possible and the steps necessary to meet the project’s objective are planned. In this step, the team identifies all of the work to be done. The project’s tasks and resource requirements are identified, along with the strategy for producing them. This is also referred to as “scope management.” A project plan is created outlining the activities, tasks, dependencies, and timeframes. The project manager coordinates the preparation of a project budget by providing cost estimates for the labor, equipment, and materials costs. The budget is used to monitor and control cost expenditures during project implementation.*

*Once the project team has identified the work, prepared the schedule, and estimated the costs, the three fundamental components of the planning process are complete. This is an excellent time to identify and try to deal with anything that might pose a threat to the successful completion of the project. This is called risk management. In risk management, “high-threat” potential problems are identified along with the action that is to be taken on each high-threat potential problem, either to reduce the probability that the problem will occur or to reduce the impact on the project if it does occur. This is also a good time to identify all project stakeholders and establish a communication plan describing the information needed and the delivery method to be used to keep the stakeholders informed.*

*Finally, you will want to document a quality plan, providing quality targets, assurance, and control measures, along with an acceptance plan, listing the criteria to be met to gain customer acceptance. At this point, the project would have been planned in detail and is ready to be executed.*

**3. Project Execution:**

*During the third phase, the implementation phase, the project plan is put into motion and the work of the project is performed. It is important to maintain control and communicate as needed during implementation. Progress is continuously monitored and appropriate adjustments are made and recorded as variances from the original plan. In any project, a project manager spends most of the time in this step.*

*During project implementation, people are carrying out the tasks, and progress information is being reported through regular team meetings. The project manager uses this information to maintain control over the direction of the project by comparing the progress reports with the project plan to measure the performance of the project activities and take corrective action as needed.*

*The first course of action should always be to bring the project back on course (i.e., to return it to the original plan). If that cannot happen, the team should record variations from the original plan and record and publish modifications to the plan. Throughout this step, project sponsors and other key stakeholders should be kept informed of the project’s status according to the agreed-on frequency and format of communication. The plan should be updated and published on a regular basis.*

*Status reports should always emphasize the anticipated end point in terms of cost, schedule, and quality of deliverables. Each project deliverable produced should be reviewed for quality and measured against the acceptance criteria. Once all of the deliverables have been produced and the customer has accepted the final solution, the project is ready for closure*

**4. Project Monitoring and Control:**

*This is where it can get tough. In parallel with the project execution, as a project manager, you report performance, and monitor and control the project. That means monitoring the project life to ensure the project is going according to plan, and if it isn’t, controlling it by working out solutions to get it back on track.*

*First, that means ensuring you capture the data (usually derived from timesheets and tasks completed) to track progress effectively against the original plan. Secondly, it means taking the data and comparing the task completion, budget spend and timeline allocated in the original plan. By comparing the plan against actuals you can establish whether or not you’re hitting the objectives for timeline, cost, and quality and success metrics.*

**5. Project Closure:**

*During the final closure, or completion phase, the emphasis is on releasing the final deliverables to the customer, handing over project documentation to the business, terminating supplier contracts, releasing project resources, and communicating the closure of the project to all stakeholders. The last remaining step is to conduct lessons-learned studies to examine what went well and what didn’t. Through this type of analysis, the wisdom of experience is transferred back to the project organization, which will help future project teams.*

**SUMMARY:**

* 1. Backend -Kavya, Mahender, Sai
  2. Web Services
* Bala Kavya: Login, Set Appointment, Update Appointments, Accepted Appointments, appointment history
* Mahender Reddy: Set Availability, View Availability, Update Profile, forgot password.
* Sai Kumar Sama: View Profile, appointment request, current appointments.
  1. Frontend(Design & code)
* Aravind: Login page, Set Appointments
* Vijayender: Current appointments, Update appointments, Set Availability.
* Mahender: Appointment requests
* Sirisha: Appointment history, Update profile, Homepage
* Jayram: View profile, Accepted request, logout.
  1. Documentation:

Bala Kavya, Sirisha, Vijayender, Mahender, Sai, Aravind, Jayram