Cégep de la Gaspésie et des Îles

# Documentation on Scheduling Events

**Project Coordinator**

**Ahmad Reza Madabadi**

## Team 6

| **Team Members** | **Student ID** |
|---|---|
| Sai Nikhil Deekonda | 1994162 |
| Naga Sravani Kurelli | 2091351 |
| Manoj Kumar Bushigampala | 2091243 |
| Sravan Kumar Adulapuram | 2091226 |
| Rajesh Gurram | 1994148 |

# Table of Contents

# 1. Introduction

This Android mobile application is “Scheduling Events” where users can create, attend, share and show their interest in the events.

## 1.1 Purpose

We need an application for creating and sharing events with teachers and students in our school. Users can attend the events and every event will be displayed with pictures, a banner, a title, description of the event, the number of attendees, the data and time of the event. Users will also have the option to show their interest whether they are interested in attending or not interested and also have the feasibility of commenting on the event. Users can also search for the specific event and also view the list of upcoming events.

## 1.2 Scope

1. The name of the application would be “Scheduling Events”.

2. This mobile application is used to schedule events like technical and cultural events. Where users can create new events, share events, view upcoming events and search for events and view all the information on the home screen. For accessing all this, the user needs to login/register to the application. If the user doesn’t register/login to the application, the user won’t be able to use any features of the application.

3. The main goal of this application is to have a user-friendly, efficient access to the application at the user’s convenience. Especially, due to this pandemic, to avoid more gatherings – teachers can schedule events and students can attend them at their comfort wherein teachers can also know the percentage of the people interested in attending the events. Similarly, students can also use this application and based on the majority of the students attending we can plan the next event which can help an increase in the head count attending the future events.

# 2. General Description

## 2.1 Product Functions

- Login

User can login as a student or teacher or admin by entering correct credentials such as Email Id and Password

- Register

Admin will create/edit/delete user(student/teacher).

- Logout

User can logout of the application

- Forgot Password

User will receive his password to email ID if he forgets his password by entering email ID.

- Add Event

Student/teacher can add a new event by entering all the information.

- Event Info

User can view all the event information such as banner, title, picture, date, time and place of the event.

- Attendees Count

User can see number of people who are attending the event.

- Edit/Remove Event

Admin/Student/Teacher can remove an event

- Comment Event

Student/Teacher can comment on the event.

- Delete Comment

Admin can delete any comment if he finds it not secure.

- Search Event

Student/teacher can search for any event from the list of events.

- Update student/teacher

Admin can update student/teacher profile if necessary whereas student/teacher can just view their information from their profile.

- Upcoming Events

User can see list of all the upcoming events.

- Share Event

Student/teacher can share event through Gmail and text message.

## 2.2 Actors

**Primary Actors**:

- Student / Teacher
- Admin

**Secondary Actors**:

- Database

## 3. Specific Requirements

### 3.1 Functional Requirements

| Requirement ID | Requirement Statement | Must/Should/ Could/ Would | Comments |
|---|---|---|---|
| FR001 | Login | Must | Admin/Student/Teacher must log into the application using his credentials. |
| FR002 | Forgot Password | Could | Student/Teacher will receive his/her password to email ID |
| FR003 | Profile | Could | Student/Teacher can update user details like name, password, phone number in user profile |
| FR004 | Add Event | Should | Student/Teacher can add events |
| FR005 | Event Details | Should | Admin/Student/Teacher can view the details of event |
| FR006 | My events | Should | Student/Teacher can check the events added by him/her |
| FR007 | Favourite Events | Could | Student/Teacher can check the events he/she attending |
| FR008 | Invited people | Could | Student/Teacher will check the people who are attending to the event |
| FR009 | Edit Event | Could | Student/Teacher can edit events posted by them with updated people |
| FR010 | Admin add Teachers/Students | Could | Admin can add users |
| FR011 | Admin edit Teachers/Students | Could | Admin can Edit users |
| FR012 | Admin delete Teachers/Students | Could | Admin can delete users |

|  |  |  |  |
|---|---|---|---|
| FR013 | Manage events | Could | Admin can remove events which are improper |
| FR014 | Manage Comments | Could | Admin can delete comments made by the students/Teacher |

## 3.2 Non-Functional Requirements

| **Requirement ID** | **Requirement Statement** | **Must/Should/ Could/Would** | **Comments** |
|---|---|---|---|
| NFR 001 | Performance | Should | The application should be easy to handle and should navigate in the most efficient and expected way. |
| NFR 002 | Portability | Must | The application must be compatible and should run on android devices. |
| NFR 003 | Security | Should | The application should provide access to only registered users. |
| NFR 004 | Scalability | Could | Databases of an application can handle large volumes of data at high. |
| NFR 005 | Usability | Should | The Application should easily adapt to different screen sizes. |

## 3.3 Other Requirements

### 3.3.1. Software Requirements

For developing the application, the following are the software requirements

1. Android Development Tool version 4.2.1
2. Android SDK and Eclipse Plug-ins for Android ADT (Recent versions)

**Technologies and Languages used to Develop**

1. Android
2. Java
3. Front end: XML
4. Back-end Database: MySQL

**Debugger and Emulator**

1. Android Dalvik Debug Monitor service
2. Android Emulator (Android Virtual Device)

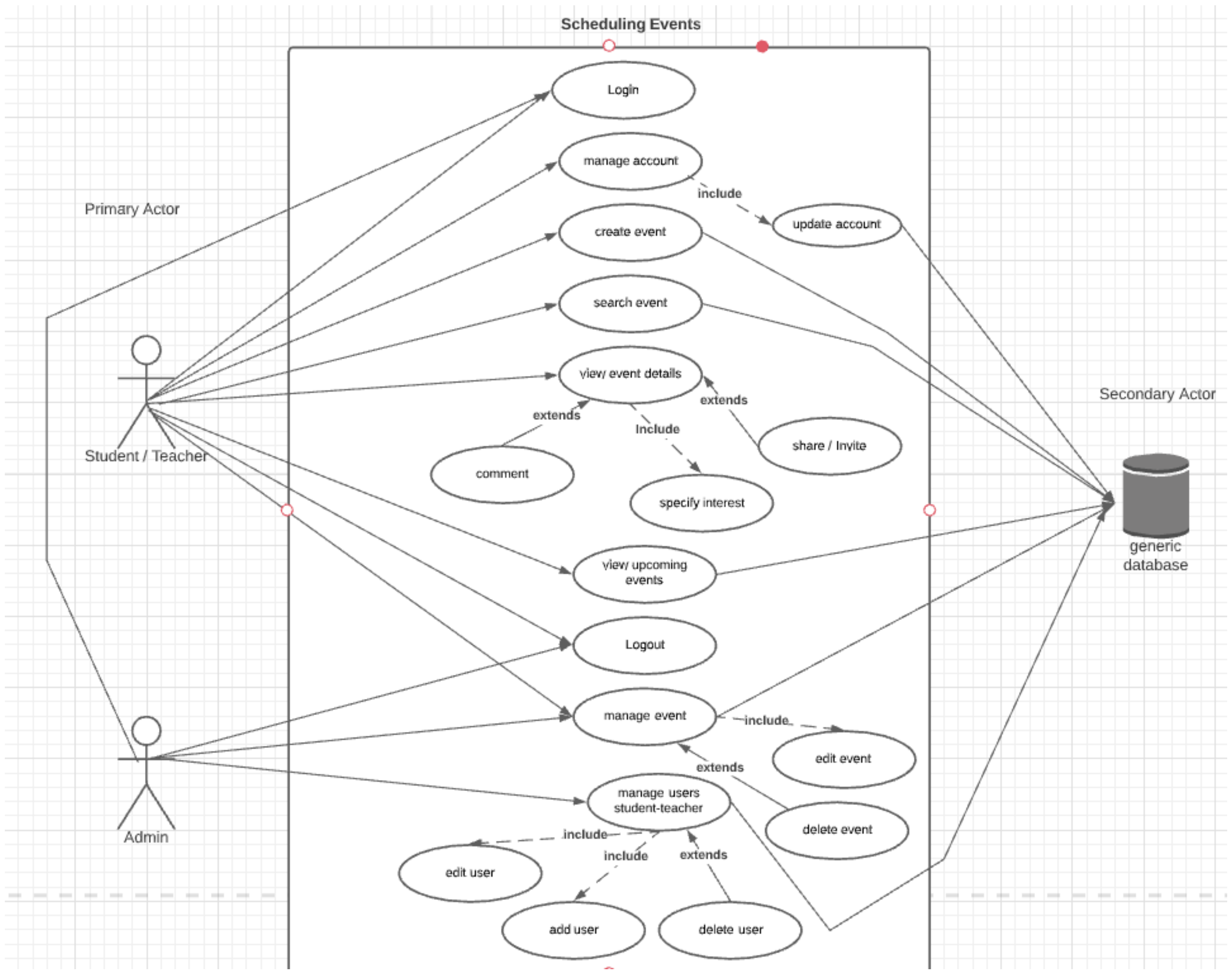
### 3.3.2. Hardware Requirements

For developing the application, the following are the Hardware Requirements:

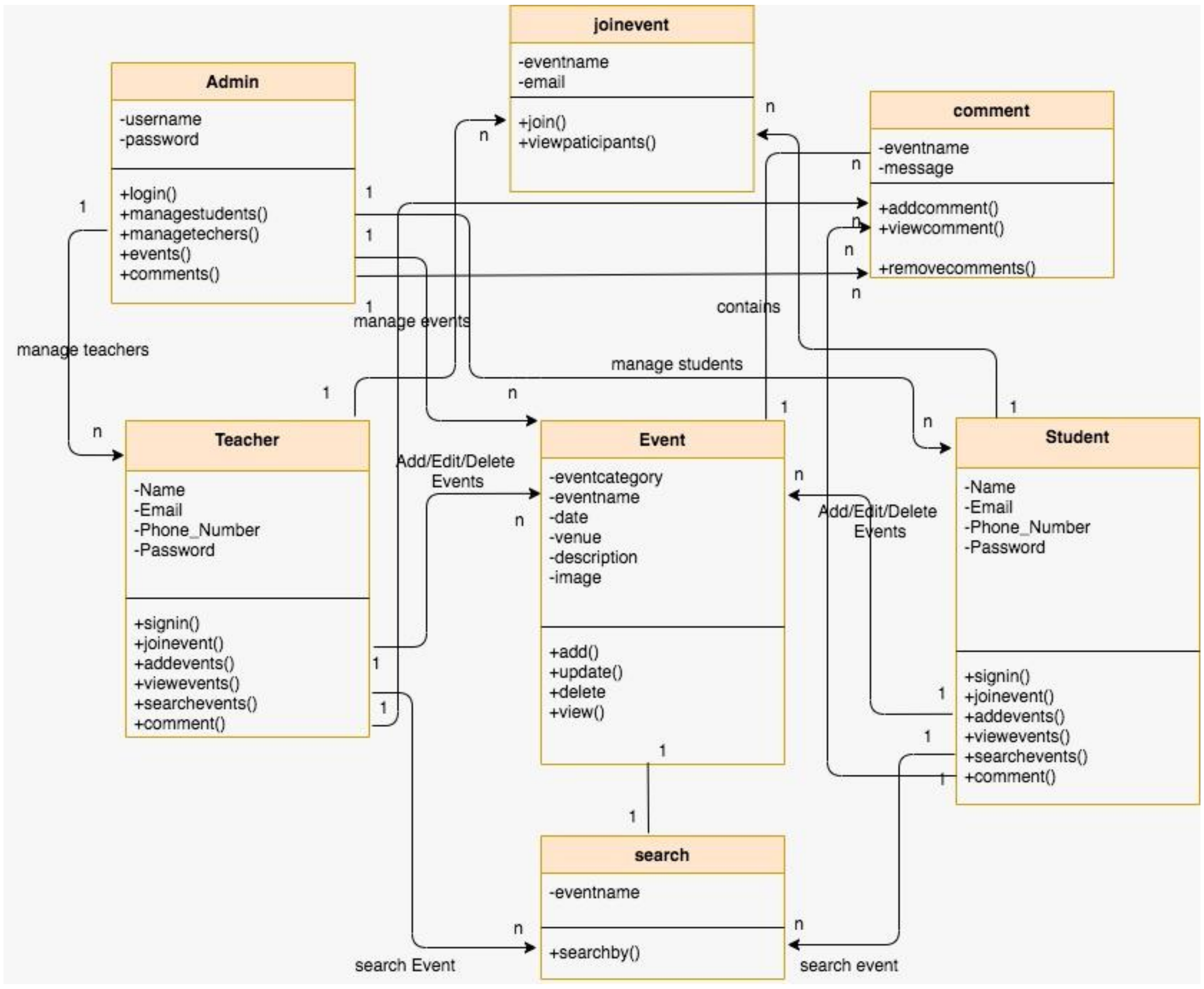
- Processor: Pentium IV or higher
- RAM: 256 MB
- Space on Hard Disk: minimum 512MB

## 4. Analysis Models

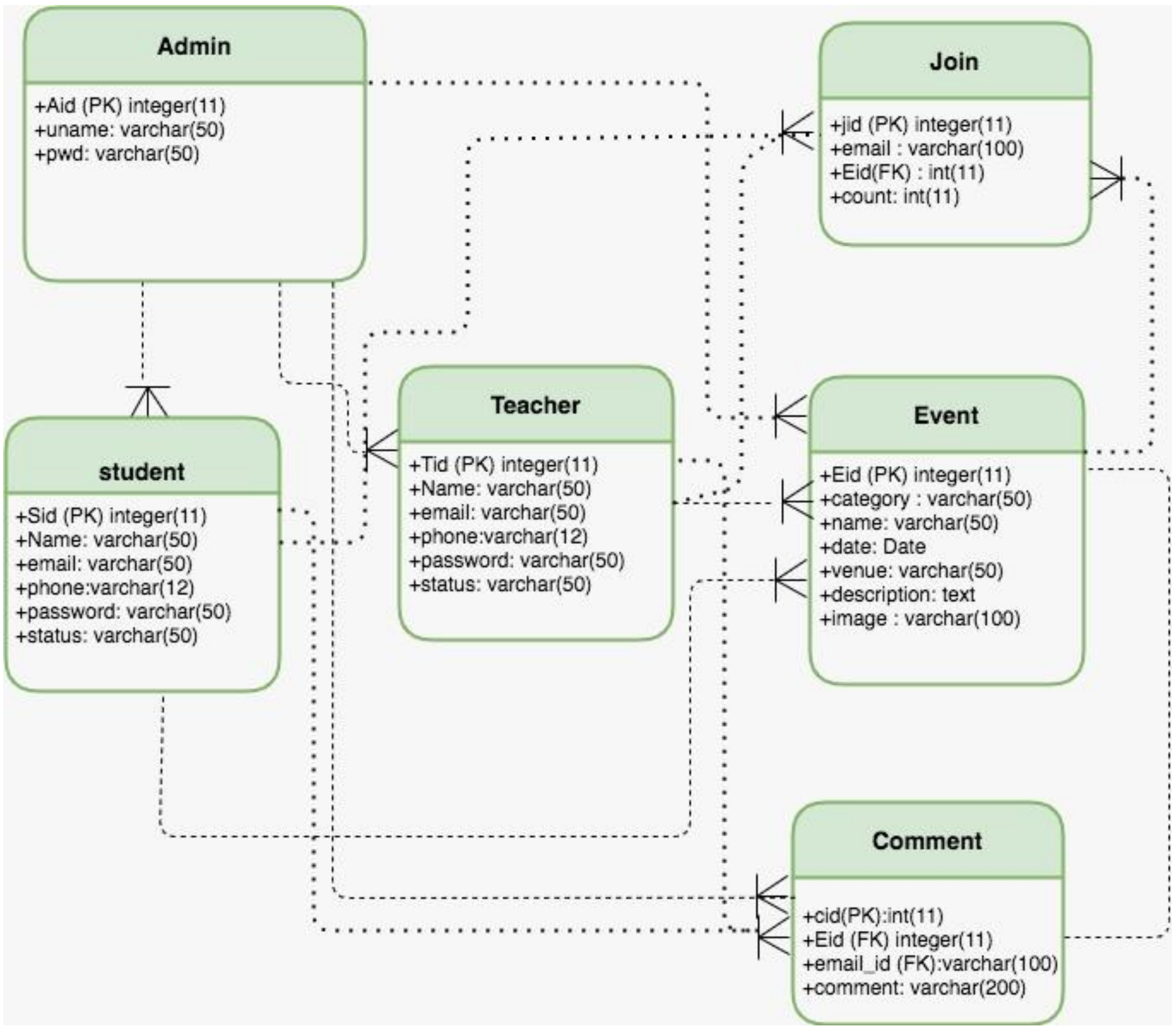
### 4.1 Use Case Diagram:



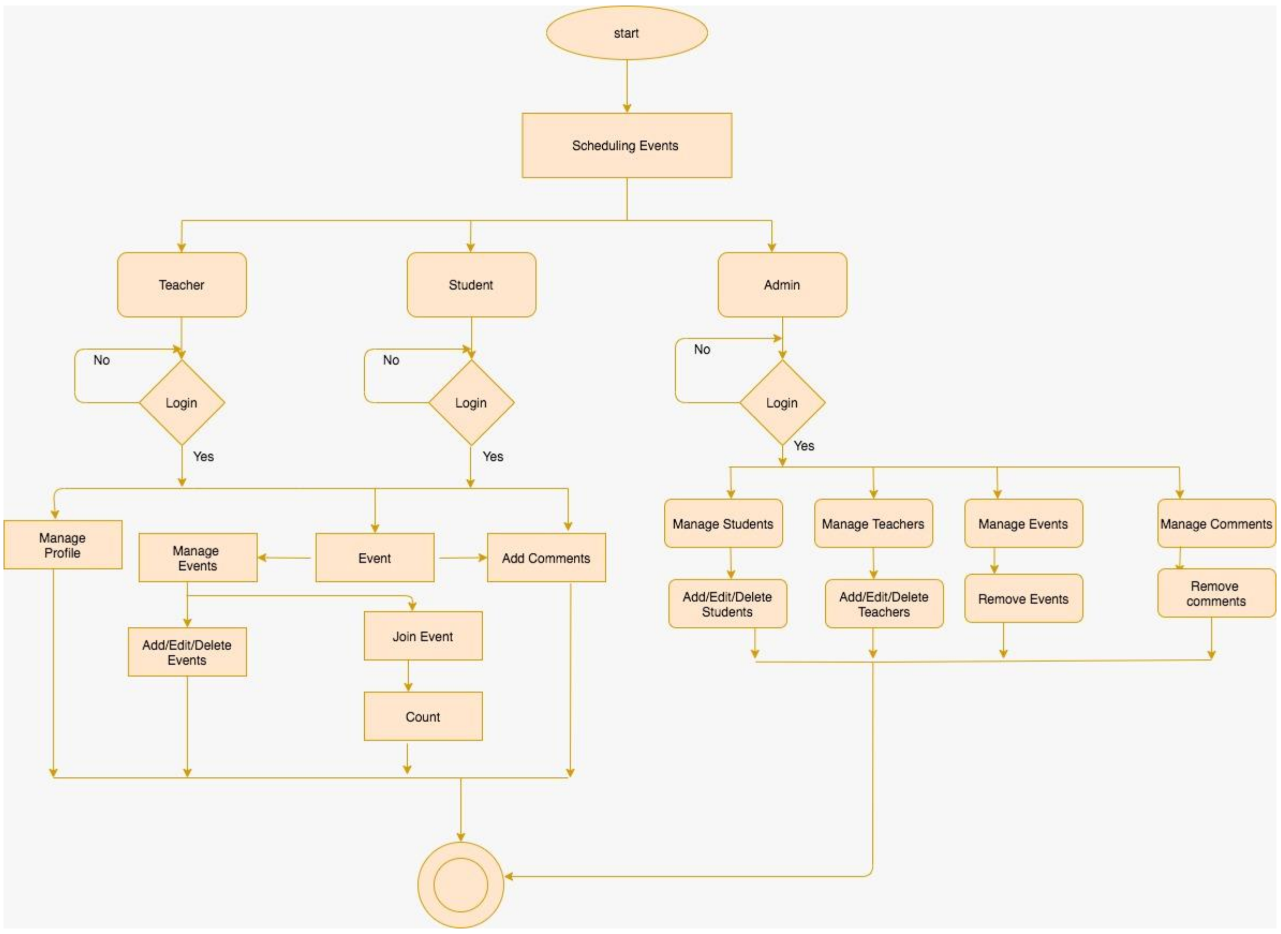
## 4.2 Class Diagram:



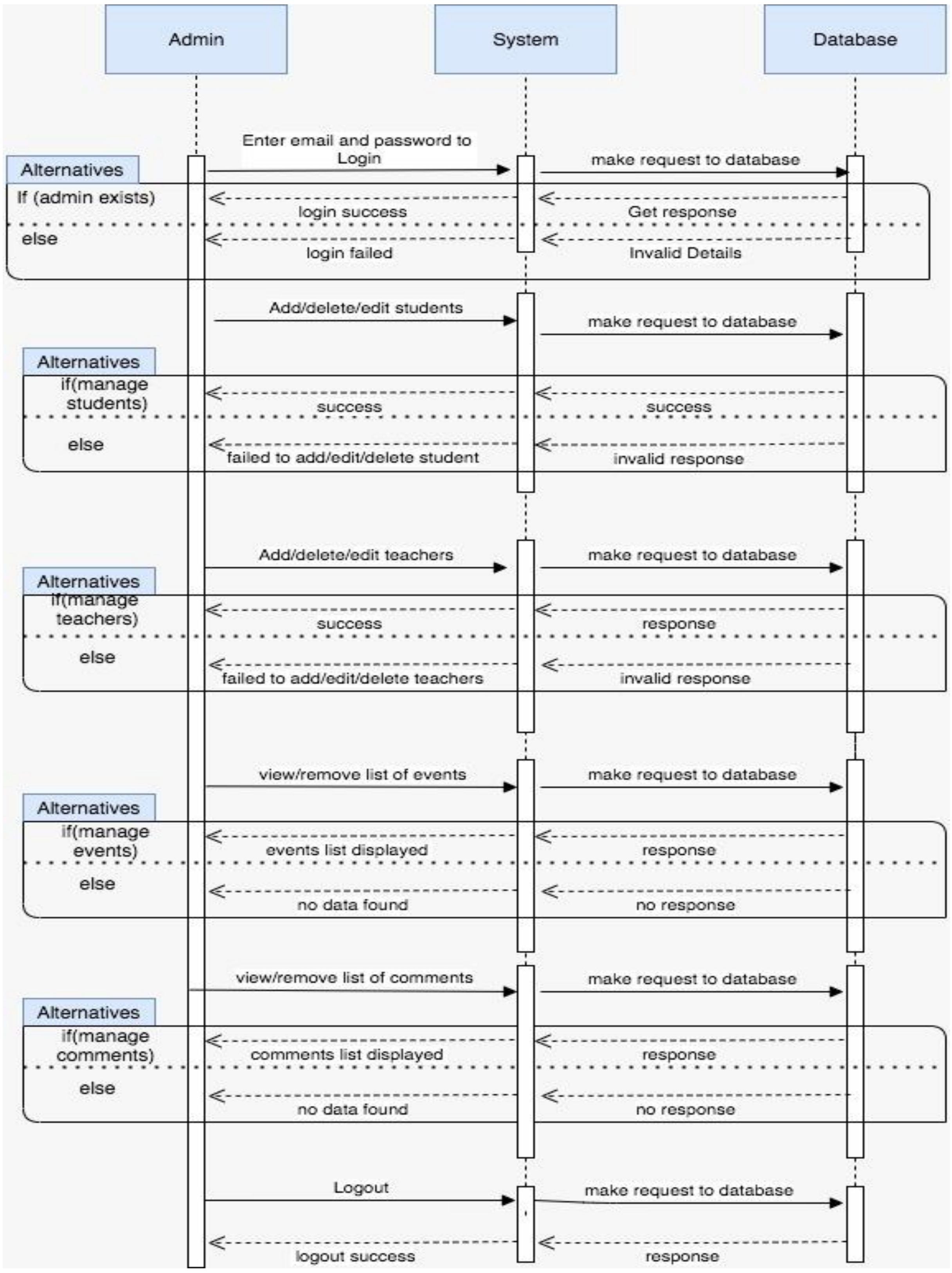
## 4.3 Database Schema



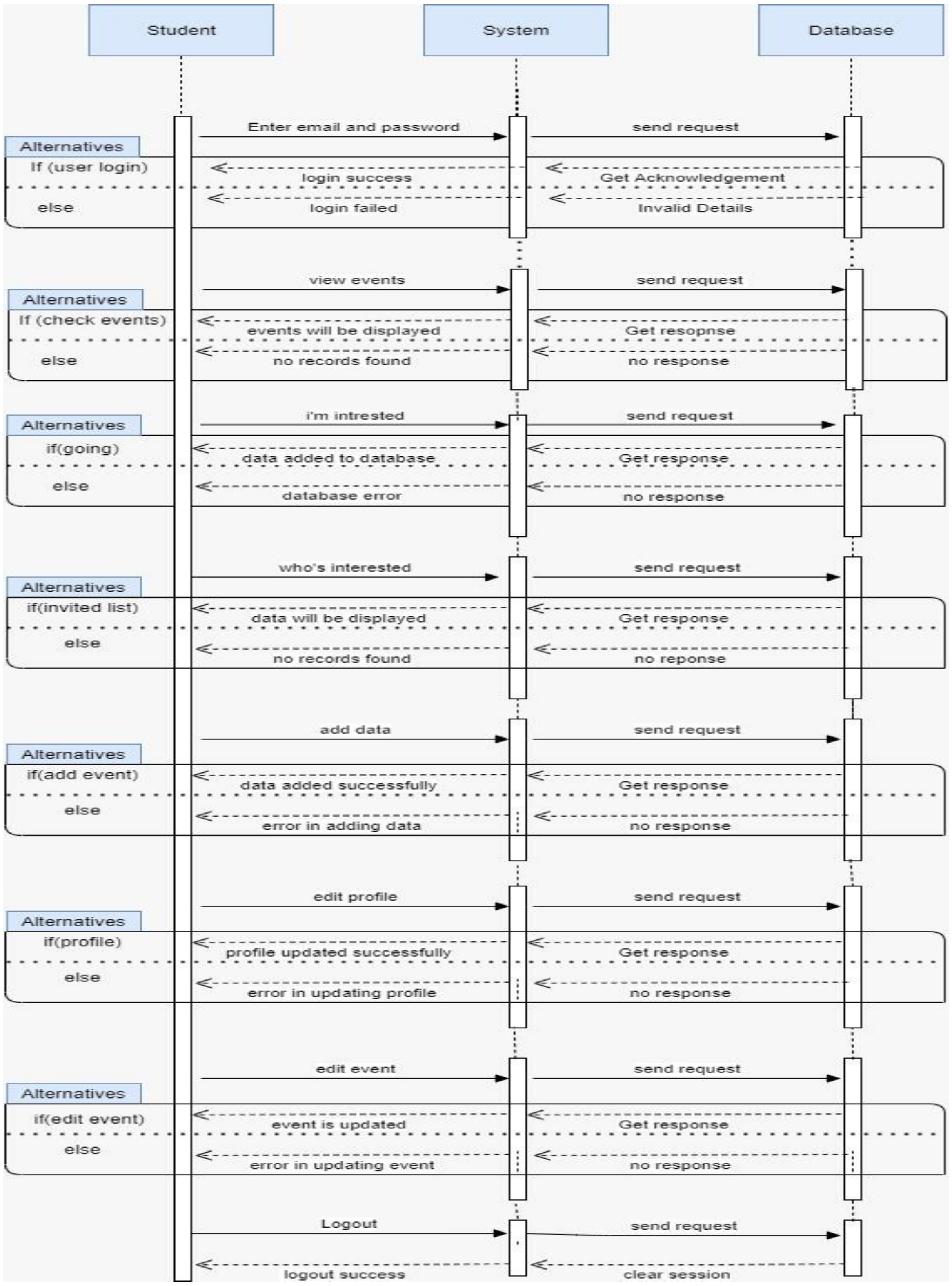
## 4.4 Activity Diagram



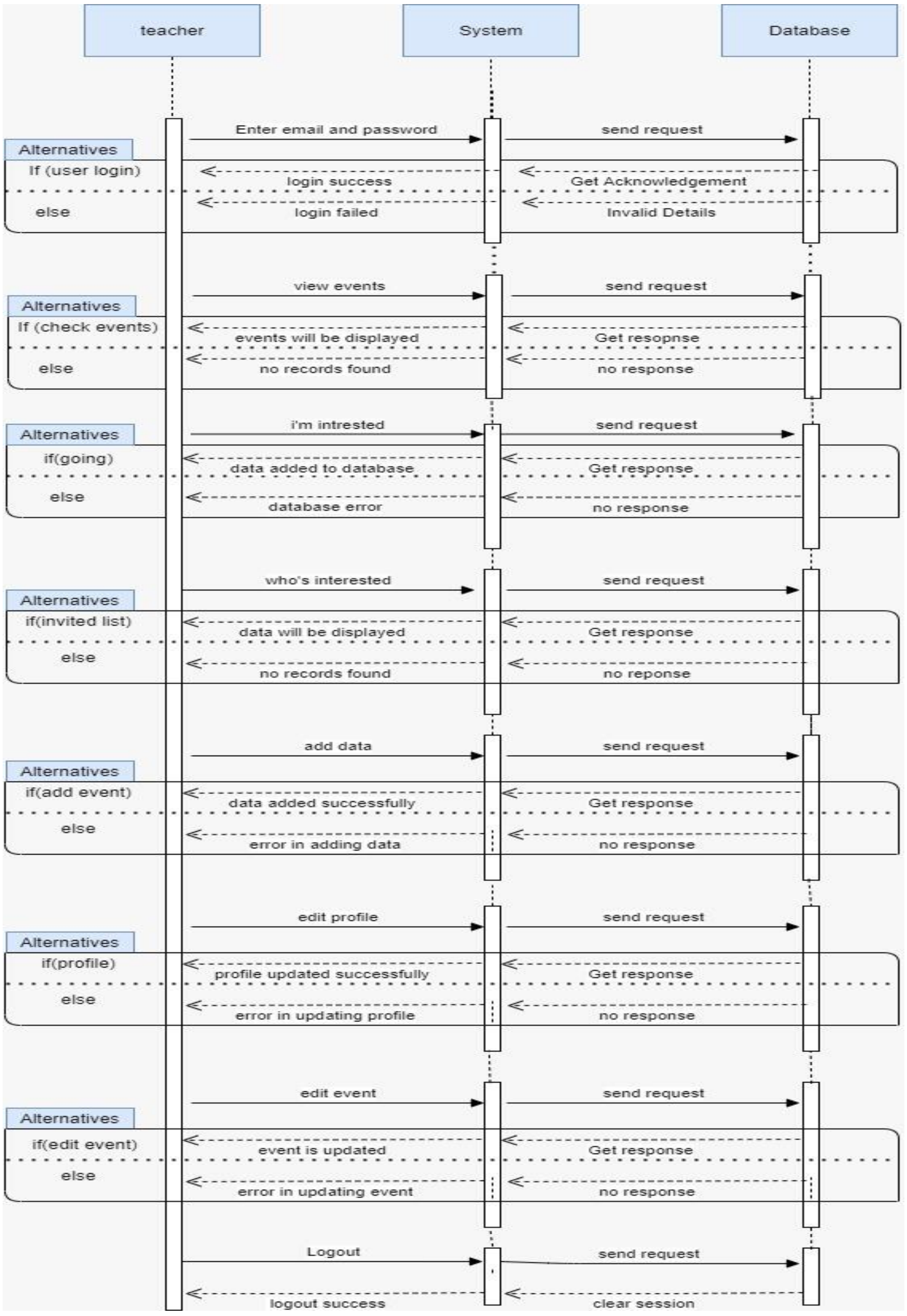
## 4.5 Sequence Diagram – Admin



Sequence Diagram – Student



## Sequence Diagram – Teacher



## 4.6 Database

A database is an organized collection of data, generally stored and accessed electronically from a computer system. These model data as rows and columns in a series of tables, and the vast majority use SQL for writing and querying data.

### 4.6.1 Data Dictionaries

| **Admin** | | | | | |
|---|---|---|---|---|---|
| | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Aid (PK) | INT | 11 | N | Contains Admin id |
| 2 | uname | VARCHAR | 50 | N | Username for admin login |
| 3 | pwd | VARCHAR | 50 | N | Password for admin login |

| **Student Registration** | | | | | |
|---|---|---|---|---|---|
| | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Sid (PK) | INT | 11 | N | Contains Student id |
| 2 | Name | VARCHAR | 50 | N | First name for Student to login |
| 3 | Email | VARCHAR | 50 | N | Email of Student |
| 4 | Phone | VARCHAR | 12 | N | Phone number of Student |
| 5 | Password | VARCHAR | 50 | N | Password for student to login |
| 6 | Status | VARCHAR | 50 | N | Status of the student |

| **Teacher Registration** | | | | | |
|---|---|---|---|---|---|
| | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Tid (PK | INT | 11 | N | Contains Teacher id |
| 2 | Name | VARCHAR | 50 | N | Full name for teacher |

| | | | | | |
|---|---|---|---|---|---|
| 3 | Email | VARCHAR | 50 | N | Email of Teacher |
| 4 | Phone | VARCHAR | 12 | N | Phone number of Teacher |
| 5 | Password | VARCHAR | 50 | N | Password for Teacher |
| 6 | Status | VARCHAR | 50 | N | Status of the Teacher |

| Join | | | | | |
|---|---|---|---|---|---|
| | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Jid (PK) | INT | 11 | N | Contains unique id for record |
| 2 | email | VARCHAR | 100 | N | Email of the student/teacher |
| 3 | Eid (FK) | VARCHAR | 11 | N | Event id |
| 4 | count | VARCHAR | 11 | N | Number of people interested in the event |

| Event | | | | | |
|---|---|---|---|---|---|
| | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |
| 1 | Eid (PK) | INT | 11 | N | Unique event id |
| 2 | category | VARCHAR | 50 | N | Category of the event |
| 3 | name | VARCHAR | 50 | N | Name of the event |
| 4 | date | Date | | N | Date of the event |
| 5 | venue | VARCHAR | 50 | N | Location of the event |
| 6 | Description | Text | | N | Description of the event |
| 7 | image | VARCHAR | 100 | N | Images of the events |

| Comment | | | | | |
|---|---|---|---|---|---|
| | **Column Name** | **Data Type** | **Length** | **Nullable** | **Description** |

| | | | | | |
|---|---|---|---|---|---|
| 1 | cid (PK) | INT | 11 | N | Unique comment id |
| 2 | Eid (FK) | VARCHAR | 11 | N | Unique id of event |
| 3 | Email_id | VARCHAR | 100 | N | Email of the student/teachers |
| 4 | comment | VARCHAR | 200 | N | Comment made by the student/teacher |

## 4.6.2. Database Script

```
drop database if exists Event;
create database Event;
use Event;

/* Admin table */
 create table if not exists admin (
  a_id            int (11)          not null
  Uname           varchar (50)      not null
  Pwd             varchar (50)      not null,
primary key (a_id),
 );

/* student */
create table if not exists Student (
  sid               int (11)               not null,
  name              varchar (50)           not null,
  Email             varchar (50)           not null,
  phone             int (12)               not null,
  password          varchar (50)           not null,
  status            varchar (50)           not null,
 primary key (sid),
);

/* teacher */
create table if not exists Student (
  tid               int (11)               not null,
  name              varchar (50)           not null,
  Email             varchar (50)           not null,
  phone             int (12)               not null,
  password          varchar (50)           not null,
  status            varchar (50)           not null,
primary key (tid),
);
```

**/* Event */**

```
create table if not exists event (
  Eid (PK)          int(11)            not null,
  category          varchar (50)       not null,
  Name              varchar (50)       not null,
  date              Date               not null,
  venue             varchar (50)       not null,
  description       Text               not null,
  image             varchar (100)      not null,
primary key (Eid),
);
```

## 5. Fully Dressed Use Cases

| | | |
|---|---|---|
| **Use Case ID** | UC1 | |
| **Use Case Name** | Login | |
| **Actor** | Admin/Student/Teacher | |
| **Description** | This use case describes the login screen of the application | |
| **Triggering event** | After the splash screen Admin/ Student /Teacher redirects to login page | |
| **Preconditions** | Users must have a good internet connection. | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Admin/student /Teacher | 1. Admin/student/Teacher will enter their credentials and click on the login button<br>2. System will check the details. If the details exist the user redirects to home page |
| **Post conditions** | Actors can view home page after logging in | |
| **Exception - Conditions** | If the user enters the wrong details an error message will be displayed | |

| | |
|---|---|
| **Use Case ID** | UC2 |
| **Use Case Name** | Forgot Password |
| **Actor** | Student /Teacher |

| | | |
|---|---|---|
| **Description** | This use case describes the password resetting of the application | |
| **Triggering event** | Student /Teacher clicks on the forgot password | |
| **Preconditions** | Users should be registered to the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Student /Teacher | 1. Student /Teacher will enter the mail id<br>2. System will check the mail id and send the password to user |
| **Post conditions** | The Actor can login into the application with new password | |
| **Exception - Conditions** | If the email entered by the student/teacher is invalid then an error message will be displayed | |

| | | |
|---|---|---|
| **Use Case ID** | UC3 | |
| **Use Case Name** | Manage Profile | |
| **Actor** | Student /Teacher/Admin | |
| **Description** | This use case describes about the profile updating for student/teacher/admin | |
| **Triggering event** | Student /Teacher/Admin clicks on My profile | |
| **Preconditions** | Users should be logged into the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Student /Teacher<br>/Admin | 1. Admin will click on student/teacher profile button and update the details if required<br>2. System will update the database with new details |

| | |
|---|---|
| **Post conditions** | The Student/Teacher/Admin can view their updated profile |
| **Exception - Conditions** | Internet connection error |

| | | |
|---|---|---|
| **Use Case ID** | UC4 | |
| **Use Case Name** | Add Event | |
| **Actor** | Student /Teacher | |
| **Description** | This use case describes about adding new event | |
| **Triggering event** | Student/Teacher clicks on add event | |
| **Preconditions** | Student /Teacher should be logged into the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Student /Teacher | 1. Student/Teacher will add the event by entering data like name, venue, date, image and some description.<br>2. The date will be stored in the database |
| **Post conditions** | The Student/Teacher can see the events added by him/her | |
| **Exception - Conditions** | Internet connection error or database connection error | |

| | |
|---|---|
| **Use Case ID** | UC5 |
| **Use Case Name** | My Event |
| **Actor** | Student /Teacher |

| | | |
|---|---|---|
| **Description** | This use case describes about the event added by student/teacher | |
| **Triggering event** | Student /Teacher clicks on my event | |
| **Preconditions** | Student /Teacher should be logged into the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Student /Teacher | 1. Student/teacher will check the events added by them<br>2. System will fetch the details of the events based on a unique id |
| **Post conditions** | The Student/Teacher can see the events added by him/her | |
| **Exception - Conditions** | Internet connection error or database connection error | |

| | | |
|---|---|---|
| **Use Case ID** | UC6 | |
| **Use Case Name** | Edit Event | |
| **Actor** | Student /Teacher | |
| **Description** | This use case describes about the editing events added by the student/teacher | |
| **Triggering event** | Student /Teacher clicks on edit button in My event | |
| **Preconditions** | Student /Teacher should be logged into the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Student /Teacher | 1. Student/teacher clicks on the edit button and edit the data if there are any changes<br>2. System will store the new details by |

| | | |
|---|---|---|
| | | replacing the old ones |
| **Post conditions** | The Student /Teacher can see the events with updated details | |
| **Exception - Conditions** | Internet connection error or database connection error | |

| | | |
|---|---|---|
| **Use Case ID** | UC7 | |
| **Use Case Name** | Event Description | |
| **Actor** | Student /Teacher/Admin | |
| **Description** | This use case describes about the event description | |
| **Triggering event** | Student /Teacher/Admin clicks on event to see description | |
| **Preconditions** | Student /Teacher/Admin should be logged into the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Student /Teacher /Admin | 1. Student/teacher/Admin will click on the event to see event details<br>2. System will fetch all events from the database and shows them to student/teacher/Admin |
| **Post conditions** | The Student /Teacher can see the event descriptions | |
| **Exception - Conditions** | Internet connection error or database connection error | |

| | |
|---|---|
| **Use Case ID** | UC8 |

| | | |
|---|---|---|
| **Use Case Name** | Invited People | |
| **Actor** | Student /Teacher/Admin | |
| **Description** | This use case describes about the people interested in the event | |
| **Triggering event** | Student /Teacher/Admin clicks on who's going button | |
| **Preconditions** | Student /Teacher/Admin should be logged into the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Student /Teacher /Admin | 1. Student/teacher/admin will click on the who's going button and checks the people who are interested in the event<br>2. System will check the details from the database and shows details |
| **Post conditions** | The Student /Teacher can see the people who are interested in a particular event | |
| **Exception - Conditions** | Internet connection error or database connection error | |

| | |
|---|---|
| **Use Case ID** | UC9 |
| **Use Case Name** | Favourite events |
| **Actor** | Student /Teacher |
| **Description** | This use case describes about the favourite events of the student/teacher |
| **Triggering event** | Student /Teacher/clicks on the I'm interested button |
| **Preconditions** | Student /Teacher/Admin should be logged into the application |

| | | |
|---|---|---|
| **Flow of events** | **Actor** | **Flow of events** |
| | Student /Teacher /Admin | 1. Student/teacher who are interested in a particular event clicks on the I'm interested button<br>2. System will add the details to database |
| **Post conditions** | The Student /Teacher can see the people who are interested in a particular event | |
| **Exception - Conditions** | Internet connection error or database connection error | |

| | | |
|---|---|---|
| **Use Case ID** | UC10 | |
| **Use Case Name** | Admin add students/teacher | |
| **Actor** | Admin | |
| **Description** | This use case describes about admin adding student/teacher | |
| **Triggering event** | Admin clicks on the add student/teacher button | |
| **Preconditions** | Admin should be logged in the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Admin | 1. Admin adds the new student/teacher by entering details like name, phone email, password<br>2. System will add the details into the database |
| **Post conditions** | The admin will add the details of the new user | |
| **Exception - Conditions** | Internet connection error or database connection error | |

| | | |
|---|---|---|
| **Use Case ID** | UC11 | |
| **Use Case Name** | Admin delete students/teacher | |
| **Actor** | Admin | |
| **Description** | This use case describes about the admin deleting the student/teacher | |
| **Triggering event** | Admin clicks on the delete student/teacher button | |
| **Preconditions** | Admin should be logged in the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Admin | 1. Admin will click on the delete student/teacher button<br>2. System will delete the details of the student/teacher so that he can't able to access the application |
| **Post conditions** | The admin will delete the details of the student/teacher | |
| **Exception - Conditions** | Internet connection error or database connection error | |

| | |
|---|---|
| **Use Case ID** | UC12 |
| **Use Case Name** | Admin Manage events |
| **Actor** | Admin |
| **Description** | This use case describes about the admin managing the events |
| **Triggering event** | Admin clicks on the manage event button |

| | | |
|---|---|---|
| **Preconditions** | Admin should be logged in the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Admin | 1. Admin will manage the events added by the student/teachers |
| **Post conditions** | The admin will manage the events added by the student/admin | |
| **Exception - Conditions** | Internet connection error or database connection error | |

| | | |
|---|---|---|
| **Use Case ID** | UC13 | |
| **Use Case Name** | Admin Manage comments | |
| **Actor** | Admin | |
| **Description** | This use case describes about the admin managing the comments | |
| **Triggering event** | Admin clicks on the manage comments button | |
| **Preconditions** | Admin should be logged in the application | |
| **Flow of events** | **Actor** | **Flow of events** |
| | Admin | 1. Admin will delete the comments added by the other student/teachers if they are violating rules<br>2. System will delete the record of it |
| **Post conditions** | The admin will manage to delete the comments made by the other students/teachers | |
| **Exception - Conditions** | Internet connection error or database connection error | |

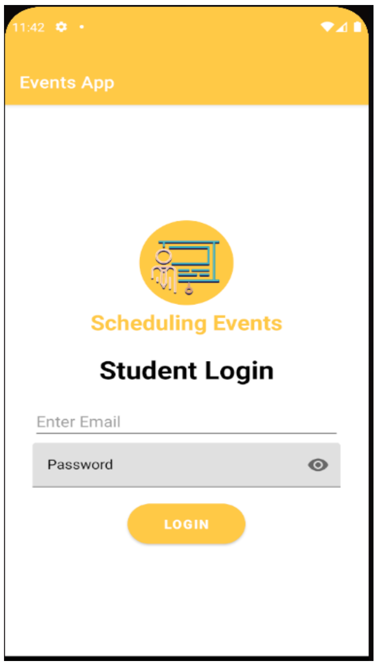
## 6. Iteration Planning

| | **Deliverables** | **Start Date** | **End Date** |
|---|---|---|---|
| Iteration1 | • Proposal that meets client requirements<br>• Functional/Non-Functional requirements<br>• User Requirements /Descriptions<br>• Proper basis for Database design<br>• Fair teamwork distribution<br>• Class Diagram<br>• Use Case Diagram<br>• Prototype<br>• Proper setup GitHub<br>• User Stories<br>• Iteration Planning | 08-062021 | 28-062021 |
| Iteration2 | • Adequate UI<br>• Efficient - Finalized Database schema<br>• Viable start to deploy code<br>• Sufficient delivery of some functionalities<br>• Adequate debugging<br>• All Sequence diagrams and Activity diagrams<br>• Fully dressed use cases<br>• Database deployment script<br>• Mobile application source code<br>• Connection to Database | 29-062021 | 12-072021 |
| Iteration3 | • Backend application source code<br>• Clear application of modifications suggested<br>• Finalized UI<br>• Logical organisation of screen flow diagrams<br>• Finalized Database<br>• Most or all functional requirements finished<br>• Finalized Backend | 13-072021 | 09-082021 |
| Iteration4 | • Finalized Front End<br>• All requested changes accomplished<br>• Fully working application<br>• minimal to no bugs/errors<br>• Complete source code (front and back end) | 10-082021 | 23-082021 |

| | | | |
|---|---|---|---|
| Iteration5 | • Finalized Front End<br>• Complete source code on GitHub<br>• All test accounts(username/passwords) included in GitHub<br>• All relevant files/information related to the project included in GitHub | 24-082021 | 30-082021 |
| Iteration6 | • Quality and aesthetic presentation of UI/UX: using multiple criteria, such as legibility, accessibility, fluidity with a structured layout, size and distance during handling, quality of written language<br>• Suitability: Appropriate and efficient response to client requests and user input<br>• Proper use of asynchronous functionalities<br>• Appropriate recovery following a programming error<br>• Respect for creative approaches: The app is characterized by originality of thought or inventiveness | 24-082021 | 30-082021 |
| Iteration7 | • Convincing demonstration of skills related to user requirements, source code, and related documentation, suggestions for improvement<br>• Relevance of the documentation presented | 24-082021 | 30-082021 |

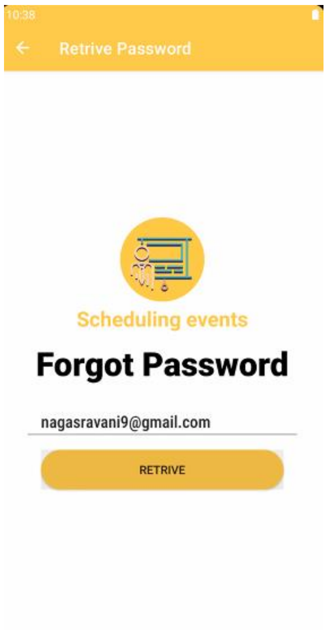
## 7. Screen Design

1. Student Login



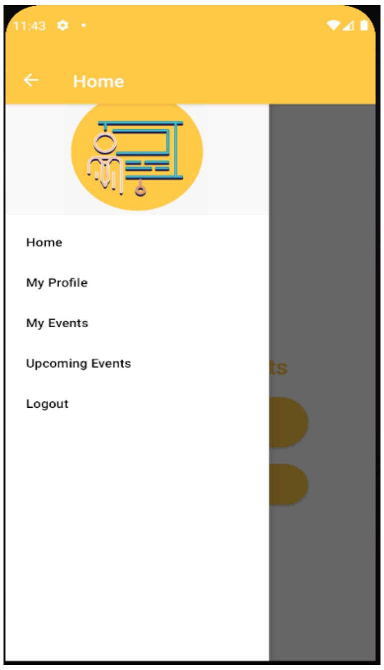
Students will login into the application

2. **Forgot password**



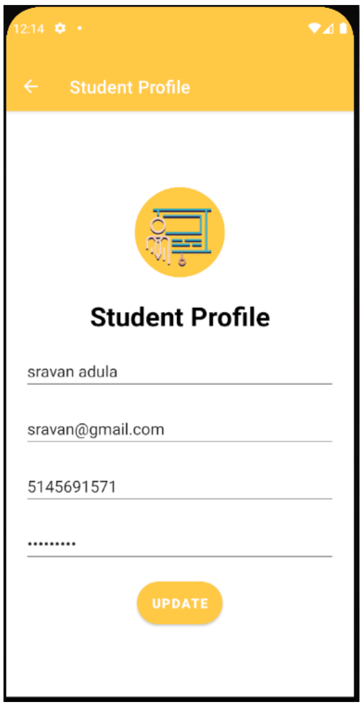
Student will reset his password

3. Student Navigation



Students will navigate in the application

4. Student profile



Student Profile

sravan adula

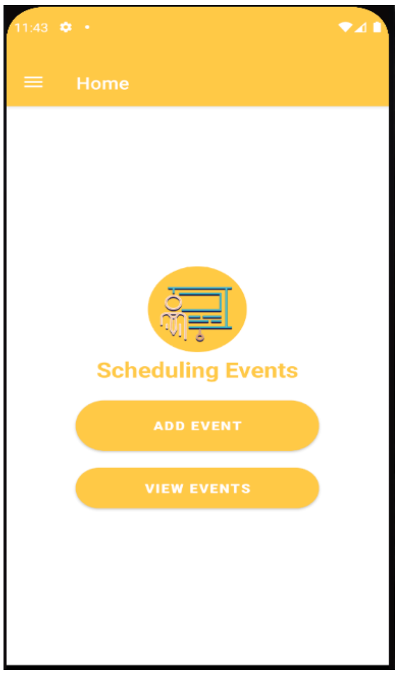
sravan@gmail.com

5145691571

•••••••••

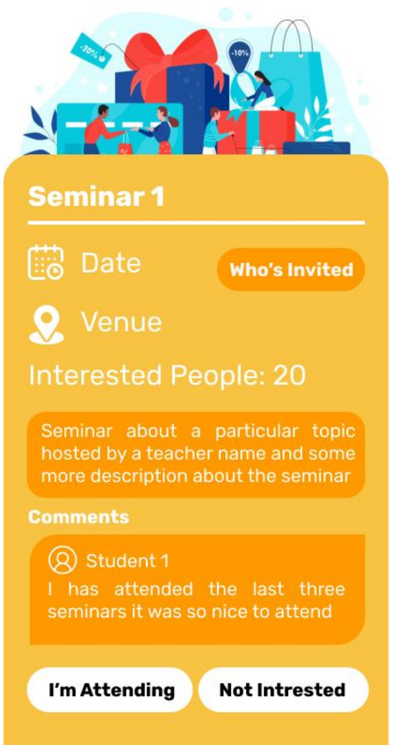
Students can update their profile

5. Student Home



Student /Teacher can add/view events

6. Event Description



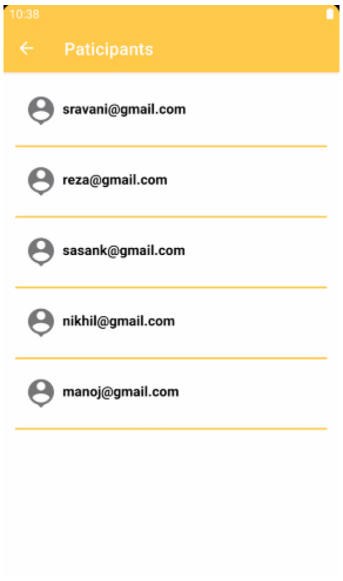
Interested People: 20

Seminar about a particular topic hosted by a teacher name and some more description about the seminar

Student 1
I has attended the last three seminars it was so nice to attend

Student /Teacher can view the event description added by the other users

7. Invited people


10:38

Paticipants

sravani@gmail.com

reza@gmail.com

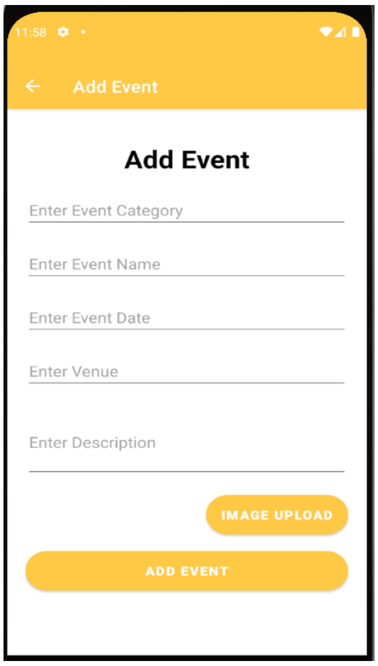
sasank@gmail.com

nikhil@gmail.com

manoj@gmail.com

Student /Teacher can check the students/Teachers list who are interested in that event

8. Add Events


11:58

Add Event

Add Event

Enter Event Category

Enter Event Name

Enter Event Date

Enter Venue

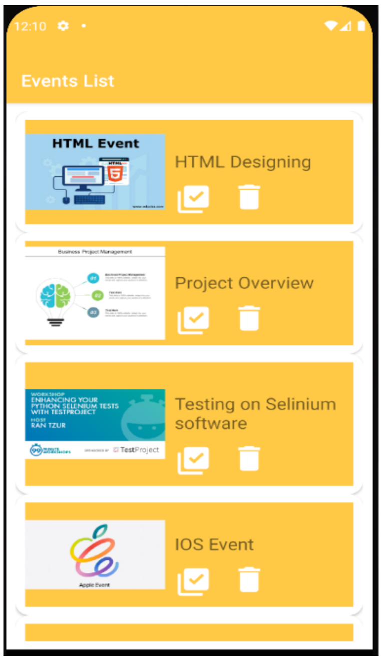
Enter Description

IMAGE UPLOAD

ADD EVENT

Student/Teacher can add new event

9. My Events



Events List

HTML Designing

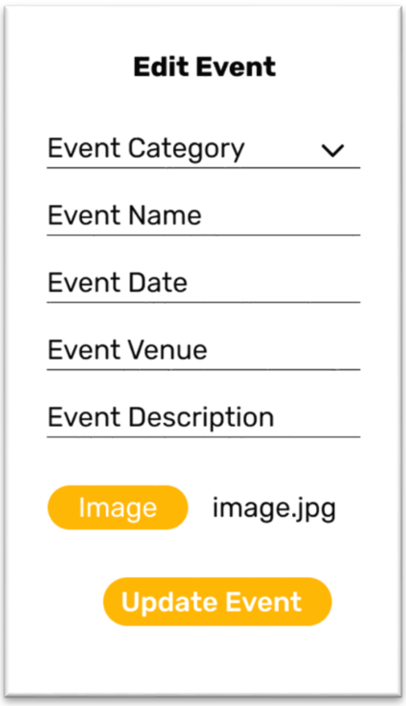
Project Overview

Testing on Selinium software

IOS Event

Students/Teachers can check the events added by them

10. Edit Event



Edit Event

Event Category

Event Name

Event Date

Event Venue

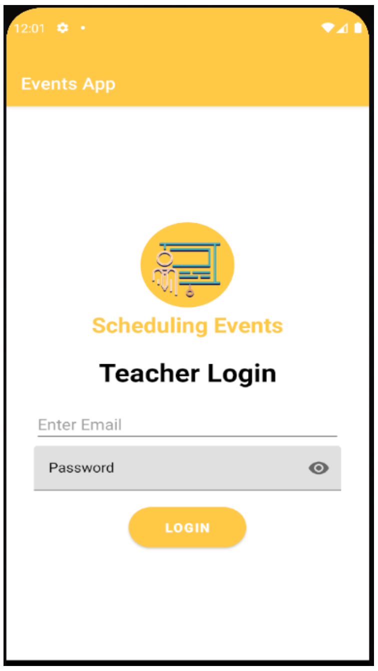
Event Description

Image image.jpg

Update Event

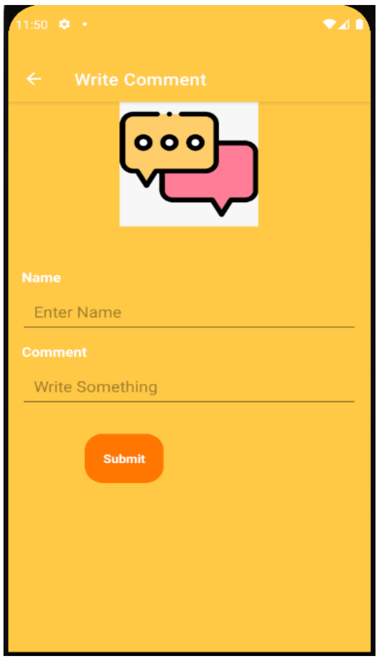
Student/Teacher can edit the events added by them

11. Teacher Login

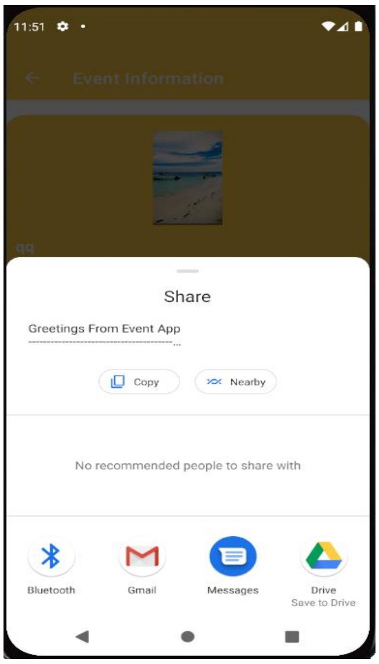


Teacher will login into the application

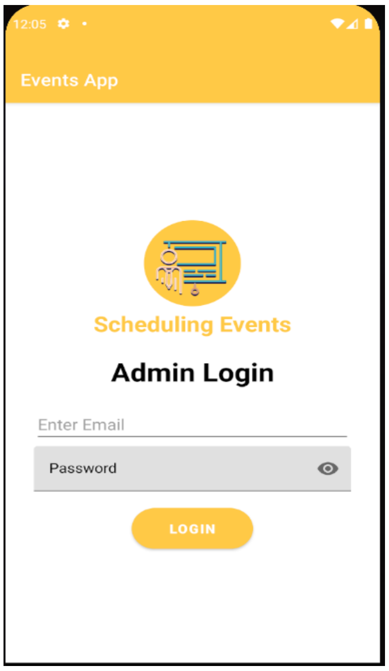
12. Write Comment (Student/Teacher)



13. Share Event

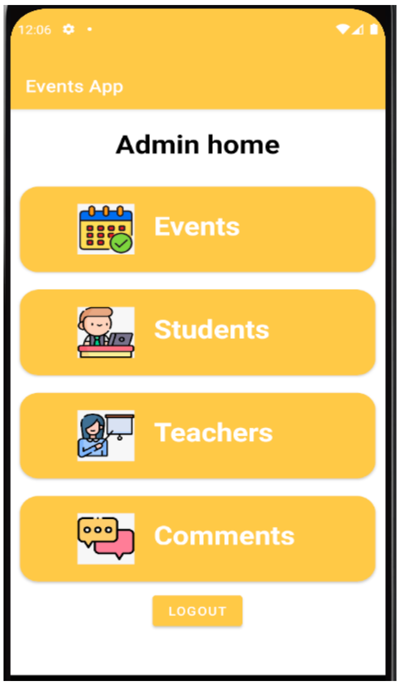


14. Admin Login



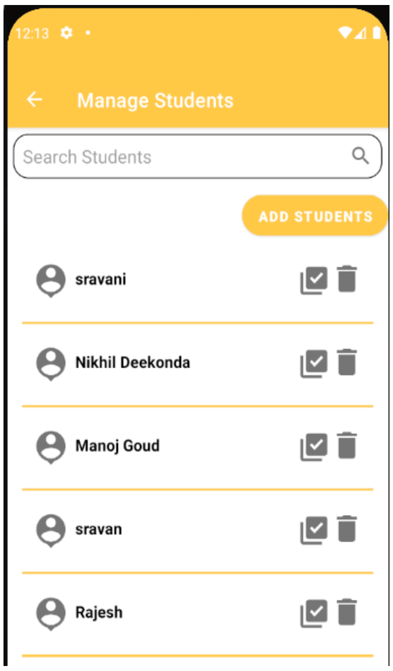
Admin will login into the application

15. Admin Navigation



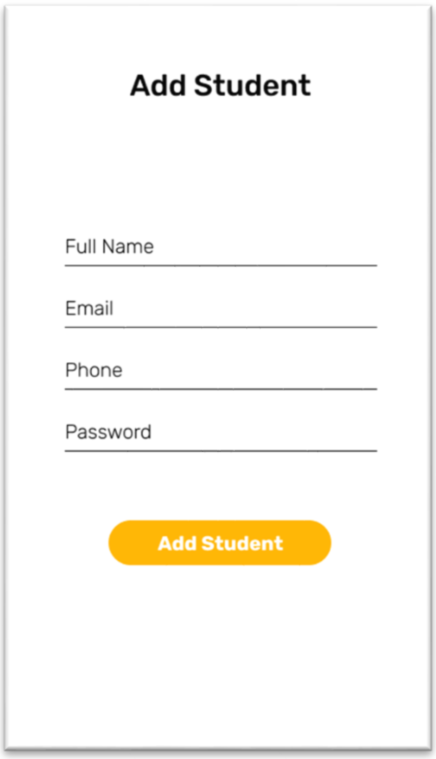
Admin will navigate in the application from here

16. Manage Student



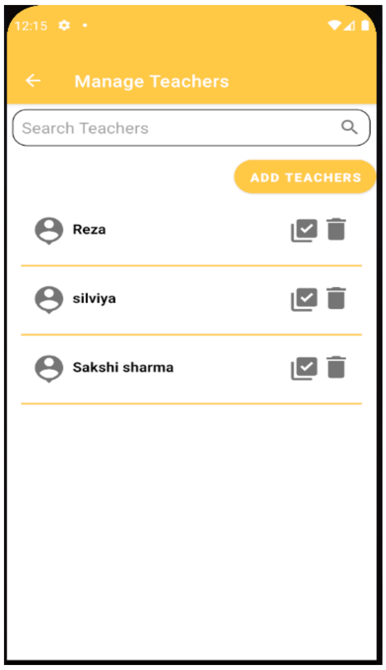
Admin will Manage students

17. Add Students



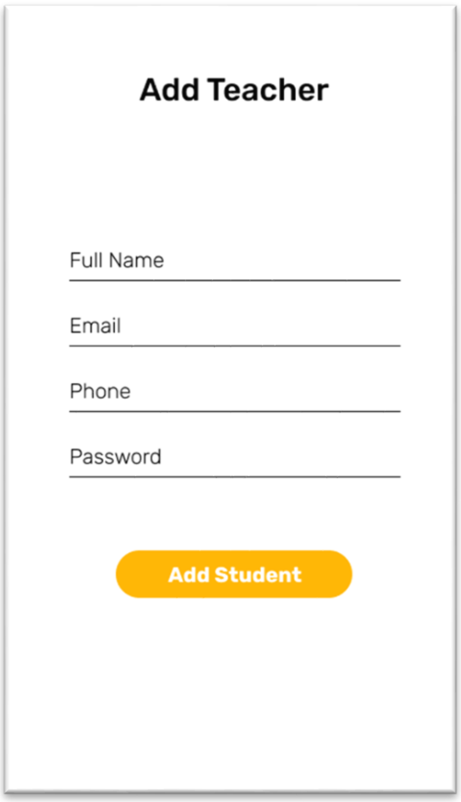
Admin can add new students

18. Manage Teachers



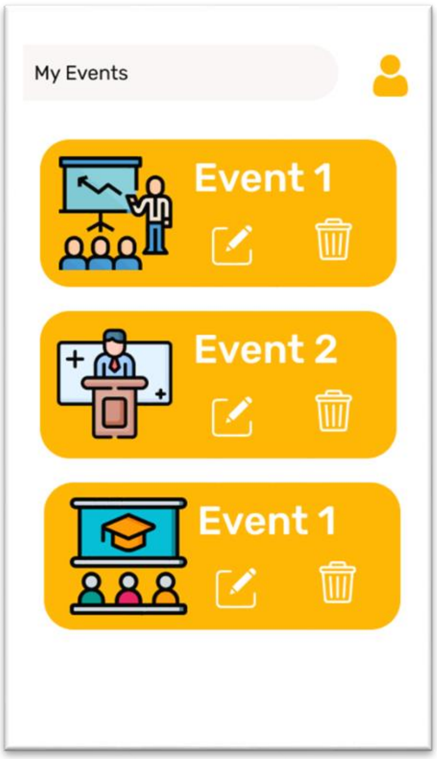
Admin manages teachers list

19. Add Teacher



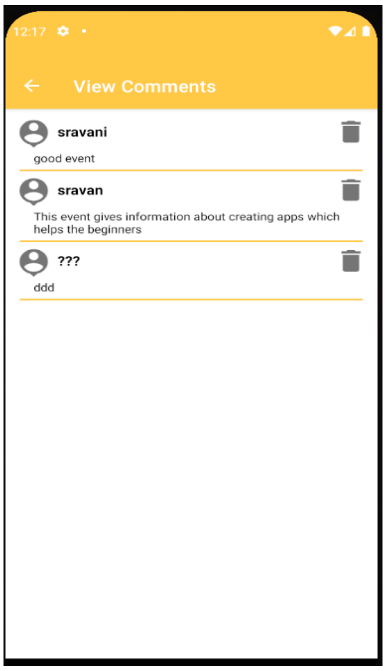
Admin can add new teachers

20. Manage all events



Admin can manage all events posted by the students/teachers

21. Manage Comments


12:17

View Comments

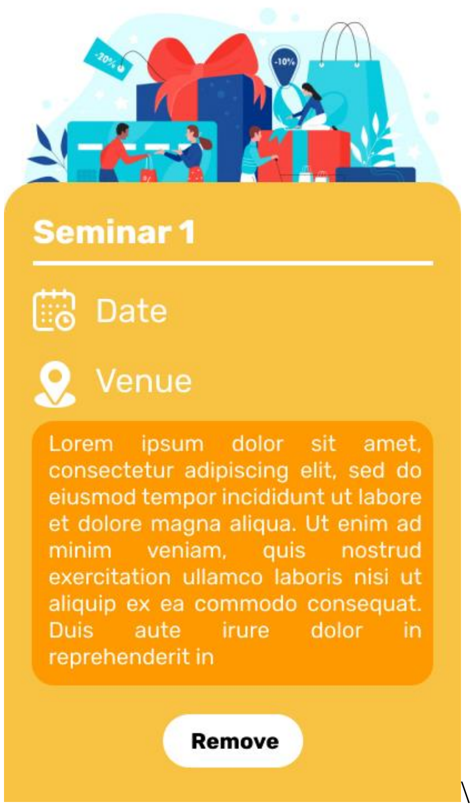
sravani
good event

sravan
This event gives information about creating apps which helps the beginners

???
ddd

Admin can manage comments made by the students/teachers

22. View/Delete Event


Seminar 1

Date

Venue

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in

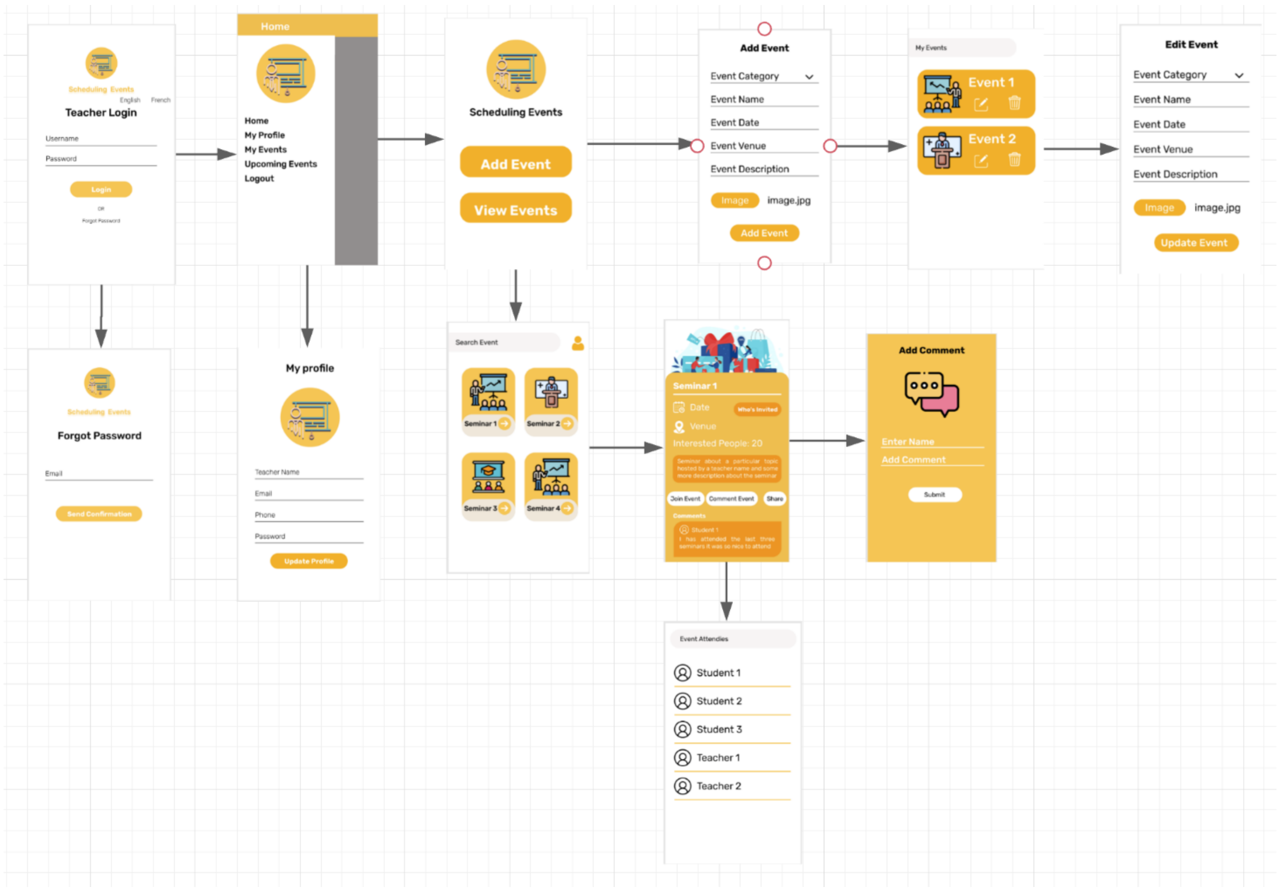
Remove

\

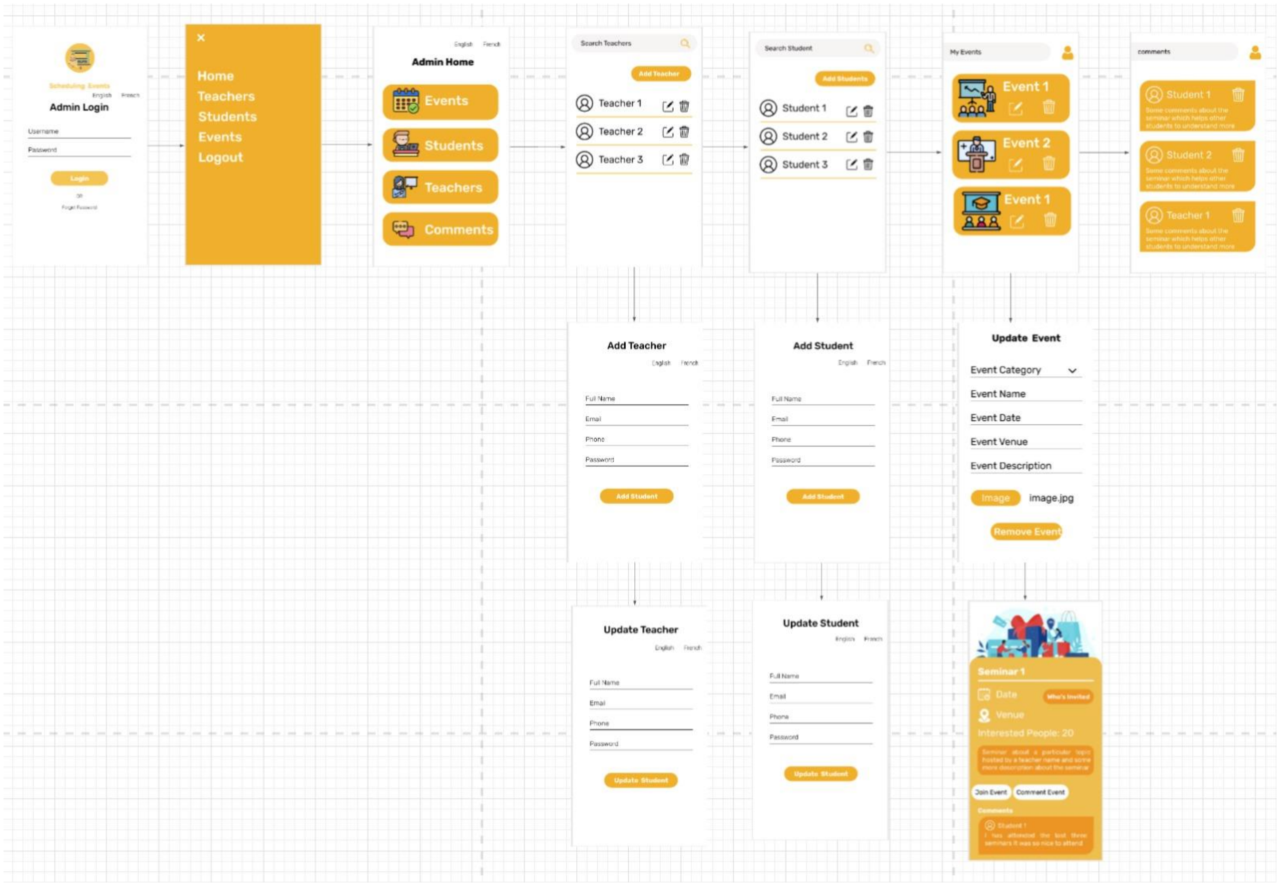
Admin can manage events

# 8.Screen Flow

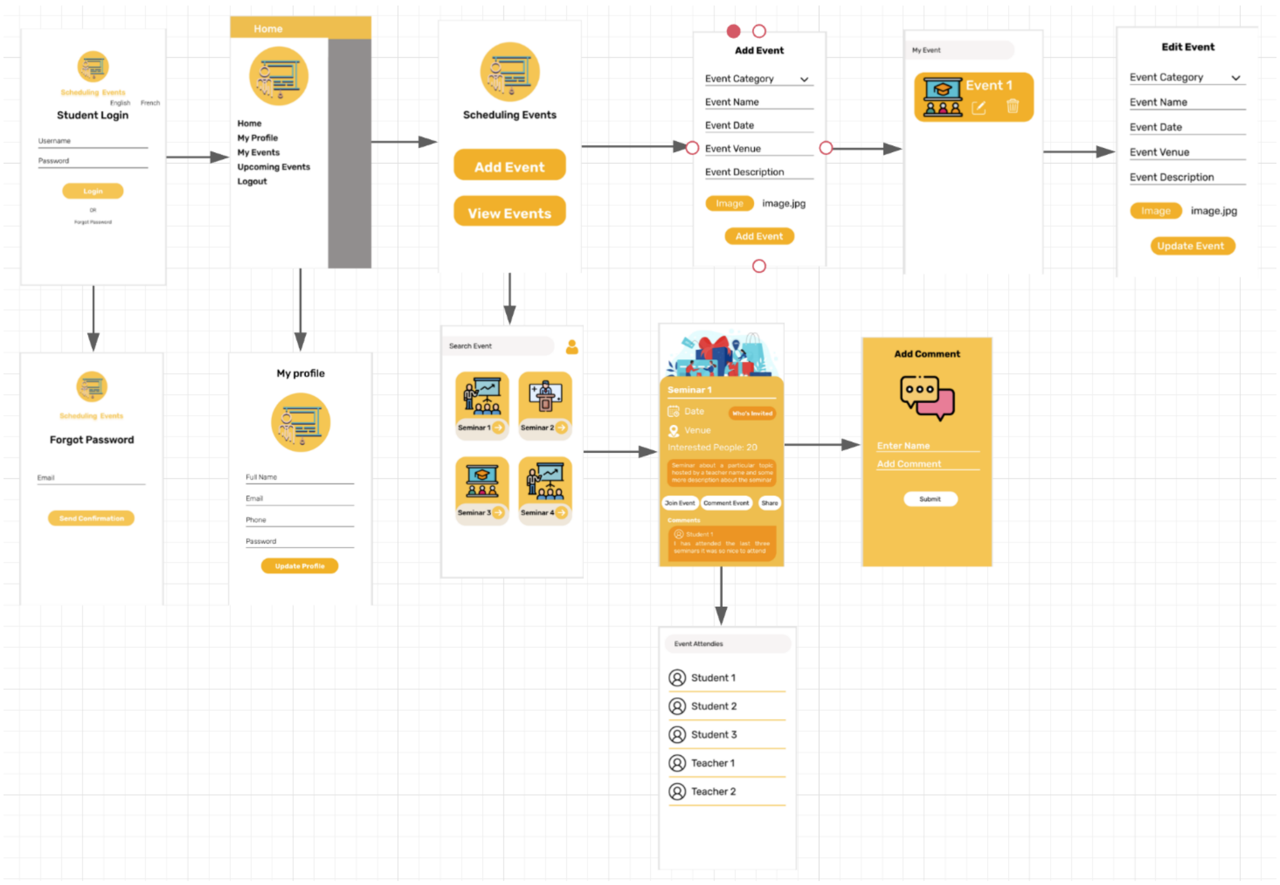
## Teacher Screen Flow



## Admin Screen Flow



Student Screen Flow



Seminar about a particular topic hosted by a teacher name and some more description about the seminar

I has attended the last three seminars it was so nice to attend

## 9. Web Service Calls

| **UC ID – 01** | Admin login |
|---|---|
| URL | http://paytracker.ca/events/adminlogin.php |
| Method | GET |
| Parameters | |

| | | |
|---|---|---|
| uname | Username of admin | String |
| pwd | Password of admin | string |

| | |
|---|---|
| Response | Example 1:<br>admin login failed<br>{<br>{"message":"Invalid username / password","status":"false"}<br>}<br>Example 2:<br>Admin login successful<br>{<br>{"message":"Login successful","status":"true"}<br>} |

| **UC ID – 02** | Get students |
|---|---|
| URL | http://paytracker.ca/events/getstudents.php |
| Method | GET |
| Parameters | |
| Response | Example 1:<br>status:false<br>{<br><br>Error fetching details<br>}<br>Example 2:<br>Status:true<br>{ |

| | |
|---|---|
| | [{"sid":"1","name":"sravani","phone":"512312312", "pass":"123","email":"sravani@gmail.com"}, {"sid":"2","name":"Reza","phone":"51212312312", "pass":"123","email":"reza@gmail.com"}] <br><br> } |

| **UC ID – 03** | Get teacher |
|---|---|
| URL | http://paytracker.ca/events/getteachers.php |
| Method | GET |
| Parameters | |
| Response | Example 1: status:false <br> { <br><br> Error fetching details <br> } <br> Example 2: <br> Status:true <br> { <br> [{"tid":"1","name":"Reza","phone":"123","pass":"512312312","email":"R@gmail.com"}] <br><br> } |

| **UC ID – 04** | Add student |
|---|---|
| URL | http://paytracker.ca/events/studentadd.php |
| Method | GET |
| Parameters | |

| | | |
|---|---|---|
| Name | Name of the student | String |
| Email | Email of the student | string |
| Phone | Phone of the student | Integer |
| Password | Password for the student | string |

| | |
|---|---|
| | |
| Response | Example 1:<br>status:false<br>{<br>{"message":"Student is already exist.","status":"false"} }<br>Example 2:<br>Status:true<br>{<br>{"message":"Student Added successfully.","status":"true"}<br><br>} |

| **UC ID – 05** | Add teacher |
|---|---|
| URL | http://paytracker.ca/events/teacheradd.php |
| Method | GET |
| Parameters | <table><tr><td>Name</td><td>Name of the teacher</td><td>String</td></tr><tr><td>Email</td><td>Email of the teacher</td><td>string</td></tr><tr><td>Phone</td><td>Phone of the teacher</td><td>Integer</td></tr><tr><td>Password</td><td>Password for the teacher</td><td>string</td></tr></table> |
| Response | Example 1:<br>status:false<br>{<br>{"message":"Teacher is already exist.","status":"false"}<br>}<br>Example 2:<br>Status:true<br>{<br>{"message":"Teacher Added successfully.","status":"true"}<br>} |

| **UC ID – 06** | Student login |
|---|---|
| URL | http://paytracker.ca/events/studentlogin.php |

| | |
|---|---|
| Method | GET |
| Parameters | |

| | | |
|---|---|---|
| uname | username of the student | string |
| Password | Password for the student | string |

| | |
|---|---|
| Response | Example 1:<br>status:false<br>{<br>{"message":"Invalid username / password","status":"false"}<br>}<br>Example 2:<br>Status:true<br>{<br>{"message":"Login successful","status":"true"}<br>} |

| | |
|---|---|
| **UC ID – 07** | Student profile |
| URL | http://paytracker.ca/events/studentprofile.php?email=sravani@gmail.com |
| Method | GET |
| Parameters | |

| | | |
|---|---|---|
| email | email of the student | string |

email= sravani@gmail.com

| | |
|---|---|
| Response | Example 1:<br>status:false<br>{<br>Error fetching details<br>}<br>Example 2:<br>Status:true<br>{<br>[{"name":"sravani","phone":"512312312","pass":"123","email":"sravani@gmail.com"}] |

|  |  |
|---|---|
|  | } |

| **UC ID – 08** | Student update profile |
|---|---|
| URL | http://paytracker.ca/events/studentupdateprofile.php |
| Method | GET |
| Parameters | <table><tr><td>Name</td><td>Name of the student</td><td>String</td></tr><tr><td>Email</td><td>Email of the student</td><td>string</td></tr><tr><td>Phone</td><td>Phone of the student</td><td>Integer</td></tr><tr><td>Password</td><td>Password for the student</td><td>string</td></tr></table> |
| Response | Example 1:<br>status:false<br>{<br>Error updating details<br>}<br>Example 2:<br>Status:true<br>{<br>{"message":"Updated successfully.","status":"true"}<br>} |

| **UC ID – 09** | Teacher login |
|---|---|
| URL | http://paytracker.ca/events/teacherlogin.php |
| Method | GET |
| Parameters | <table><tr><td>uname</td><td>uname of the teacher</td><td>String</td></tr><tr><td>Password</td><td>Password for the teacher</td><td>string</td></tr></table> |
| Response | Example 1:<br>status:false<br>{<br>{"message":"Invalid username / password","status":"false"} |

| | |
|---|---|
| | }<br>Example 2:<br>Status:true<br>{<br>{"message":"Login successful","status":"true"}<br>} |

| **UC ID – 10** | teacher profile |
|---|---|
| URL | http://paytracker.ca/events/studentprofile.php?email=sravani@gmail.com |
| Method | GET |
| Parameters | <table><tr><td>email</td><td>email of the student</td><td>string</td></tr></table><br>email= sravani@gmail.com |
| Response | Example 1:<br>status:false<br>{<br>Error fetching details<br>}<br>Example 2:<br>Status:true<br>{<br>[{"name":"sravani","phone":"512312312","pass":"123","email":"sravani@gmail.com"}]<br>} |

| **UC ID – 11** | teacher update profile |
|---|---|
| URL | http://paytracker.ca/events/studentupdateprofile.php |
| Method | GET |
| Parameters | <table><tr><td>Name</td><td>Name of the student</td><td>String</td></tr><tr><td>Email</td><td>Email of the student</td><td>string</td></tr></table> |

| | | |
|---|---|---|
| Phone | Phone of the student | Integer |
| Password | Password for the student | string |

| | |
|---|---|
| Response | Example 1:<br>status:false<br>{<br>Error updating details<br>}<br>Example 2:<br>Status:true<br>{<br>{"message":"Updated successfully.","status":"true"}<br>} |

| **UC ID - 12** | Add event |
|---|---|
| URL | http://paytracker.ca/events/addevent.php |
| Method | GET |
| Parameters | |

| | | |
|---|---|---|
| email | Email to contact | String |
| msg | Description of the event | string |
| type | Type of the event | String |
| Acno | User id of event creator | Integer |
| Status | Status of the event | string |

| | |
|---|---|
| Response | Example 1:<br>event added successfully<br><br>{<br>{"message": "Event Added successfully.","status":"true"}<br>}<br><br>Example 2 error in adding successfully |

| | |
|---|---|
| | { <br> Error in adding events <br> } |