COMS 4156
First Iteration Report

Team Members:
Asmita Kumar ak4581
Shikha Asrani sa3864
Soamya Agrawal sa3881
Vani Jain vj2245

1. **Github Repository**
   https://github.com/SoamyaAgrawal17/SAVS

2. **API Documentation**
   https://app.swaggerhub.com/apis/savs2/SAVS/1.0.0

3. **Integration Testing Documentation**
   https://documenter.getpostman.com/view/14290543/UVC9gkKY

4. **Unit Testing Report**
   Below are the user stories that we have implemented, and for most of them, unit and system tests that have been handled as part of the first iteration.

   Student

- **Signing up as a new user**
  Acceptance Test for pass cases:
  - Input would be all required and valid user information and the result expected would be that the student is able to sign up.

  Acceptance Test for fail cases:
  - Input would be invalid form field(s) such as invalid email address, missing name, etc and the result would be an error message. The user would also not be able to register in this case.

- **View all events and their details**
  Acceptance Test for pass cases:
  - The result would be that the user would be able to view all the events and corresponding details of all the events.

- **View details of a specific event**
  Acceptance Test for pass cases:
    - Input would be event_id and the result would be that the user would be able to view all details of that event

  Acceptance Test for fail cases:
    - Input would be invalid event_id and the result would be an empty response as event does not exist

- **View all upcoming events**
  Acceptance Test for pass cases:
    - Input would be the Student ID and the expected result is that the user will receive a list of all upcoming events.

- **Register for an event**
  Acceptance Test for pass cases:
    - Input would be the Event ID and Student ID and the result would be that the user gets registered for the event and the number of participants registered should get incremented.

  Acceptance Test for fail cases:
    - Input would be the Student ID and the Event ID for an already registered event and the result would be that the user is informed that they're already registered.

- **View my registered events**
  Acceptance Test for pass cases:
    - Input would be the Student ID and the result would be the list of all past, upcoming, and cancelled registered events for this user.

  Acceptance Test for fail cases:
    - Input would be an invalid Student ID and the result would be an error message.

- **Create a new club**
  Acceptance Test for pass cases:
    - Input would be all required and valid club information and the result expected would be that the student is able to create a new club. The student should be assigned as the Club Head for this club.

  Acceptance Test for fail cases:
    - Input would be missing information and the result would be an error message.

- **View all clubs and my roles**
  Acceptance Test for pass cases:
    - Input would be Student ID and the result expected would be a list of all clubs and their corresponding role in that club.

  Acceptance Test for fail cases:
    - Input would be an invalid Student ID and the result would be an error message.

Club Member
All test cases for students also apply for a club member.

- **Propose an Event**
  Acceptance Test for pass cases:
    - Input would be all required and valid event information and the result expected would be that the club member is able to create a new event. The event should have status as “Proposed”.

  Acceptance Test for fail cases:
    - Input would be invalid missing information and the result would be an error message.
    - A student who is not a member or club head of the club cannot propose an event for that club

- **Edit event**
  Acceptance Test for pass cases:
    - Input would be Event ID and allowed editable field values, and the result expected would be the event gets updated with new values.
- Acceptance Test for fail cases:
    - Input would not include required values like “club_id”, this would throw an error
    - Input would be an invalid Event ID (event not proposed by the user, past event, non-existent event), and the result would be an error message
    - A student who is not a member or club head of the club cannot edit events for that club

- **See all the events I proposed**
  Acceptance Test for pass cases:
    - Input would be the Student ID and the result expected would be a list of events that the user proposed and their corresponding status.

  Acceptance Test for fail cases:

- Input would be an invalid Student ID, and the result would not return any events

Club Head
All test cases for club members and students also apply for a club head.

- **Edit club details**
  Acceptance Test for pass cases:
  - Input would be any permutation of valid club information and the result expected would be that the club details are changed.

  Acceptance Test for fail cases:
  - Input would be invalid club information (missing information for the head, invalid club name) and the result would be an error message informing the club head about the invalid information.
  - Anyone other then club head trying to edit details of the club

- **Delete club**
  Acceptance Test for pass cases:
  - Input would be the club name and the expected result is that the club is deleted along with any members of the club.

  Acceptance Test for fail cases:
  - Trying to delete a club that doesn't exist.
  - Anyone other than the club head trying to delete the club.

- **Add member**
  Acceptance Test for pass cases:
  - Input would be valid club information (club id or student id), the expected result is that the member has the rights of a club member.

  Acceptance Test for fail cases:
  - Input would be invalid information (invalid club id or invalid student id) and the result would be an error message informing the club head about the invalid information.
  - Trying to add a member who is not a student.
  - Add the club member to a club that doesn't exist.

5. **Multiple user functionality**
   As part of this iteration, we haven't implemented authorization service, but we have used emailId as a body parameter in APIs, to validate and distinguish between the clients.

The functionality of the application can be extended to multiple users and use cases. Below are some use cases where this application could be extended:

- In a classroom setting, TAs could be equivalent to Club Members and Instructor could be equivalent to Club Head. The functionality could be for students, TAs and Instructors to create and manage assignments for the course.
- An event scraper could use our APIs to retrieve information regarding events and clubs.
- The application is agnostic to organization and can be extended to other universities, schools, companies looking to manage clubs and events.