SENATE MANAGEMENT SYSTEM

A PROJECT REPORT

submitted

in the partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

K ANKITHA (20B81A05R1)

MANDA ABHINAV (20B81A05Q2)

AKAVARAM SRAGVI (20321A05A0)

Under the guidance of

Mr. U. Jhashuva

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

May 2023

SENATE MANAGEMENT SYSTEM

A PROJECT REPORT

submitted

in the partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

K ANKITHA (20B81A05R1)

MANDA ABHINAV (20B81A05Q2)

AKAVARAM SRAGVI (20321A05A0)

Under the guidance of

Mr. U. Jhashuva

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CVR COLLEGE OF ENGINEERING

(*An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana- 501 510

May 2023

CVR College of Engineering

ACCREDITED BY NATIONAL BOARD OF ACCREDITATION, NAAC

(Approved by AICTE & Govt. of Telangana and Affiliated by JNT University)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled **"SENATE MANAGEMENT SYSTEM"** being submitted by **K ANKITHA (20B81A05R1), MANDA ABHINAV (20B81A05Q2), AKAVARAM SRAGVI (20321A05A0)** in partial fulfilment for the award of Bachelor of Technology in Computer Science and Engineering to the CVR College of Engineering, is a record of bona fide work carried out by them under my guidance and supervision during the year 2022-2023.

The results embodied in this project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of the project guide,	Signature of the HOD
Mr. U. Jhashuva	**Dr. A Vani Vathsala**
Assistant Professor	**Department of CSE**

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

It is a great pleasure to convey our profound sense of gratitude to our principal **Dr. K. Ramamohan Reddy, Dr. A. Vani Vathsala**, Head of CSE Department, CVR College of Engineering, for having been kind enough to arrange necessary facilities for executing the project in the college.

We deem it a pleasure to acknowledge our sense of gratitude to our project guide **Mr. U. Jhashuva** under whom we have carried out the project work. His inclusive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work.

We wish a deep sense of gratitude and heartfelt thanks to management for providing excellent lab facilities and tools. Finally, we thank all those whose guidance helped us in this regard.

ABSTRACT

The Senate Management System is a software solution that offers numerous benefits to a college. The system streamlines administrative tasks, providing a centralized platform for managing meetings, agendas, and documents. The system helps the college senate to schedule meetings, create and distribute agendas, and manage the members of the senate efficiently. The system's customizable features allow for different access levels for users, providing secure access to the necessary information.

The Senate Management System improves the efficiency of the college senate by eliminating the need for manual processes and reducing the risk of errors. The system enables the senate to make informed decisions quickly by providing easy access to essential information. The system also provides a record of all meetings, agendas, and documents, making it easier to retrieve information and track progress.

Overall, the Senate Management System offers an effective and efficient solution for managing the administrative tasks of a college senate. It provides numerous benefits that help improve the efficiency and effectiveness of the college senate.

Table of Contents

LIST OF FIGURES

ABREVIATIONS

MVC	Model View Controller
RDBMS	Relational Data Base
CSS	Cascading Style Sheet
HTML	Hyper Text Markup Language
UI	User Interface
STS	Spring Tool Suite
POM	Project Object Model
XML	Extensible Markup Language

CHAPTER 1
INTRODUCTION

1.1 Problems Identified:

Many Universities lack an efficient Senate Management System to manage its administrative processes. The current system is manual and time-consuming, causing delays in decision-making, errors in record-keeping, difficulties in tracking progress, and inefficient use of resources. This problem is negatively impacting the Senate members, faculty, staff, and the institution as a whole. Without an efficient Senate Management System, the institution's Senate is struggling to manage the records of faculty, and staff, schedule meetings and events, and allocate resources. As a result, stakeholders are experiencing frustration and delays in the decision-making process, leading to an overall negative impact on the institution's productivity and efficiency. Therefore, there is a pressing need for a Senate Management System to streamline administrative processes and improve the overall functioning of the institution's Senate.

1.2 Methodology for problem solving:

For solving the present problems in present senate management system, we can provide various features which can be enabled by the organizer in order to avoid running into those obstructions. Many of these features provided are user friendly and does not require much effort and can be easily utilized for their benefit.

1.3 Outlines of the result:

This project helps different administrations to organize their meetings in a much easy manner and does not require much effort and utilizes different features presented and able to keep the track of the meetings, members and allocate agendas to different meetings.

1.4 Significance of the project

- Easy to schedule a meeting and manage it.
- Able to track the status of the meeting.
- Provides users from unauthorized logins.
- Presents each meeting with ‘n’ number of agendas.

- Provides the complete information about the members of the senate.
- Has a well-defined database which is no hassle.
- Allows to send invites to the members of the senate.

CHAPTER 2
LITERATURE SURVEY

The Senate of an educational institution plays a critical role in academic decision-making, resource allocation, and overall governance. The effective management of the Senate's administrative processes is essential for the smooth functioning of the institution. In recent years, several studies have been conducted on the use of information technology and automated systems to manage the administrative processes of the Senate. This literature survey aims to review the existing literature related to Senate Management Systems.

Several studies have highlighted the benefits of using information technology to manage Senate processes. For example, in a study conducted by Khan and Khan (2016), it was found that the use of an automated Senate Management System improved the efficiency of the Senate's administrative processes, reduced errors, and saved time. Similarly, in a study conducted by Hassan et al. (2018), it was found that a Senate Management System helped to streamline the decision-making process, improve communication, and enhance the transparency of the Senate's operations.

Other studies have also explored the different features and functionalities that are required in a Senate Management System. For example, in a study conducted by Kaur et al. (2017), it was found that a Senate Management System should have features such as agenda management, voting mechanisms, document management, and reporting functionalities. Similarly, in a study conducted by Singh and Kumari (2018), it was found that a Senate Management System should have features such as online voting, attendance management, and scheduling functionalities.

However, there are also challenges associated with the implementation of Senate Management Systems. For example, in a study conducted by Idowu et al. (2018), it was found that the implementation of a Senate Management System required significant investment in terms of technology infrastructure, training, and support. Similarly, in a study conducted by Batool et al. (2019), it was found that the implementation of a Senate Management System required careful consideration of security and privacy concerns.

Overall, the literature suggests that a Senate Management System can improve the efficiency, transparency, and communication of the Senate's administrative processes. However, careful consideration must be given to the features and functionalities required, as well as the challenges associated with implementation. Further research is needed to explore the effectiveness of Senate Management Systems in different contexts and to identify best practices for implementation.

CHAPTER 3
SOFTWARE & HARDWARE SPECIFICATIONS

3.1 Software Requirements

3.1.1 Functional Requirements

- **User Management:** The system allows the creation, modification, and deletion of user accounts, and provide access control to different parts of the system based on user roles and permissions.
- **Meeting Scheduling:** The system allows the scheduling of meeting, and should provide access to the update, postpone and delete meetings.
- **Agenda Management:** The system allows to map different agendas to a particular meeting and also it also has the ability to view the agendas.
- **Entity profile authorization:** Password protected user accounts.

3.2.2 Non-Functional Requirements

- User interface is only in English. No other language is available.
- Internet Connection is required to use the system.
- **Usability:** The system is user-friendly and intuitive, with clean and simple user interface that is easy to navigate and understand.
- **Performance:** The system is fast and responsive, with minimal downtime and the ability to handle a large number of users and data.

These functional and non-functional requirements will ensure that the Senate Management System is able to meet the needs of the educational institutions and its users.

3.2 System Specifications

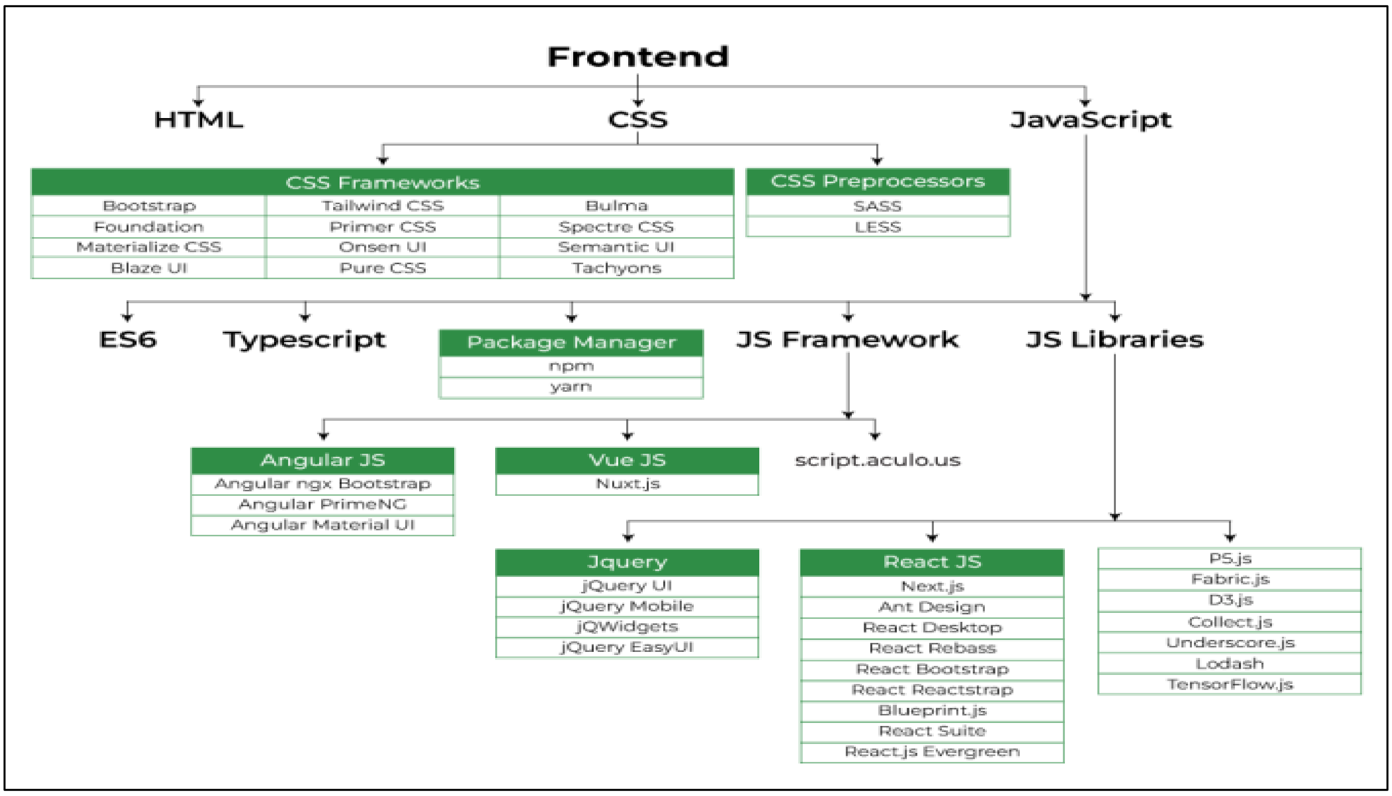
3.2.1 Hardware Specifications

Minimum hardware requirements are very dependent on the particular software being developed. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

- **Operating Systems:** Windows, Linux
- **Processor:** minimum intel i3
- **Ram:** minimum 4GB
- **Hard disk:** minimum 250gb
- **Printers/Scanners:** This management system may require the printing or scanning of documents such as meeting agendas, reports, etc.

3.2.2 Software Specifications

<u>Frontend Development</u>: The part of a website that the user interacts is termed as front end. It is also referred to as the 'client side' of the application.

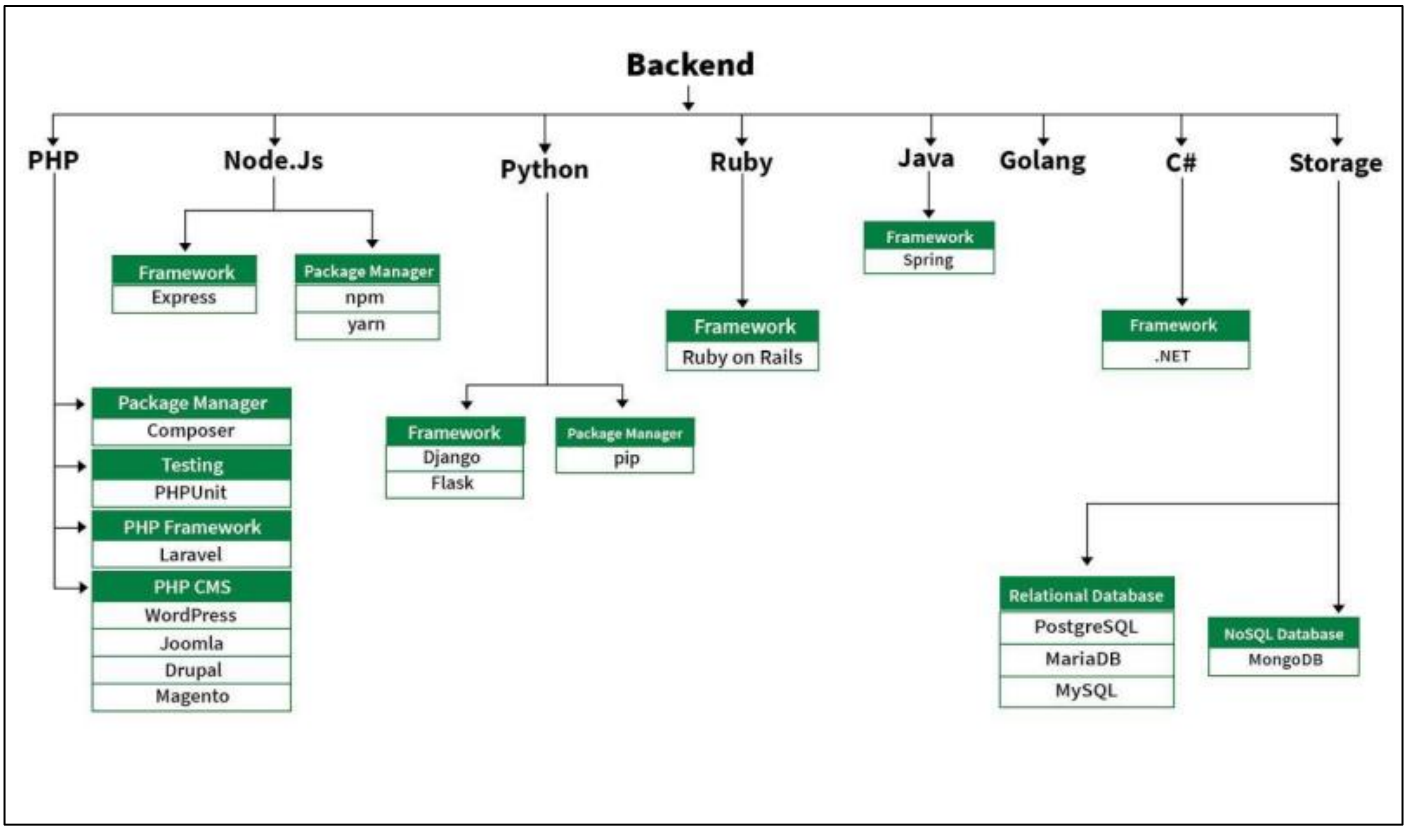


3.2.2.1 Frontend Frameworks structure

The front-end portion is built by using some of the languages which are discussed below:

- **HTML** is used to define the structure and content of the web pages that make up the Senate Management System. HTML tags are used to create headings, paragraphs, tables, forms, and other elements that make up the user interface of the system. The HTML pages are served by the Spring MVC framework, which acts as the controller and handles user requests and responses.
- **CSS** is used to define the visual appearance and layout of the web pages in the Senate Management System. CSS styles are applied to HTML elements to change the font, colour, size, and position of text and other page elements. The CSS files are linked to the HTML pages using the "link" tag in the HTML header. The Spring MVC framework also supports the use of CSS files for styling web pages.
- **Thymeleaf** is a server-side template engine that is used to generate dynamic HTML pages in the Senate Management System. Thymeleaf templates contain HTML code along with Thymeleaf expressions that can be used to insert dynamic data into the HTML pages. Thymeleaf templates are processed by the Spring MVC framework, which uses the data model and the template engine to generate the final HTML pages that are sent to the user's web browser.
- **Bootstrap** is a popular CSS framework that provides pre-built CSS classes and JavaScript components for creating responsive web pages. Bootstrap can be used to quickly and easily create a consistent and modern user interface for the Senate Management System, with features such as responsive navigation bars, grid systems, and form controls. Bootstrap can be integrated into the Senate Management System using HTML and CSS, with the Bootstrap CSS and JavaScript files linked to the HTML pages.

Backend Development: Backend is the server side of the website. It is the part of the website that users cannot see and interact. It is the portion of the software that does not come in direct contact with the users. It is used to store and arrange data.



3.2.2.2 Backend design roadmap

- **Spring** is a popular Java framework that provides a comprehensive programming and configuration model for building enterprise-level applications. It is based on the principles of Inversion of Control (IoC) and Dependency Injection (DI) that help in creating loosely coupled and highly modular applications.
- **Spring Boot** is an extension of the Spring framework that makes it easier to create stand-alone, production-grade Spring-based applications with minimal configuration. It provides a range of pre-configured components and starters that simplify the process of building and deploying Spring applications.
- **DBMS:** The software which is used to manage database is called Database Management System (DBMS). Here we have MySQL as our database. The “SQL” part of “MySQL” stands for “Structured Query Language.” SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

3.3 Deployment

STS (Spring Tool Suite):

Spring Tool Suite is an IDE based on Eclipse that provides a range of features and tools for developing and deploying Spring-based applications.

Here are the steps to use STS for the Senate Management System:

1. **Download and install STS:** You can download the latest version of STS from the Spring website and install it on your computer.
2. **Create a new Spring Boot project:** In STS, create a new Spring Boot project using the Spring Boot Starter template. This will create a new project with all the necessary files and dependencies for the Senate Management System.
3. **Configure the application**: In the project settings, configure the application properties such as the database connection details, server port, security settings, etc.
4. **Develop the application:** Use the eclipse editor to write the Java code for the Senate Management System, including the controllers, services, repositories, and models.
5. **Test the application:** Use the STS built-in testing tools to test the functionality of the Senate Management System, including the unit and integration tests.
6. **Deploy the application:** Once the application is developed and tested, use the STS tools to package the application as a JAR or WAR file and deploy it to a server or cloud platform.

3.4 MySQL (Relational Database used in this project)

MySQL Database is a client/server system that consists of a multithreaded SQL server that supports different back ends, several different client programs and libraries, administrative tools, and a wide range of application-programming interfaces (APIs). We also provide MySQL as an embedded multithreaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

Here are the steps to use MySQL in the Senate Management System:

1. Install MySQL: You can download and install MySQL on your local computer or a remote server. Ensure that you have the necessary privileges to create a database and create tables.

2. Create a database: Using the MySQL command line or a MySQL client like **MySQL Workbench**, create a new database for the Senate Management System. The database should have a name and appropriate privileges for a MySQL user.

3. Configure the application: In the application configuration files, specify the MySQL database connection details, including the database URL, username, and password.

4. Create tables: Using SQL scripts or an ORM framework like Hibernate, create the necessary tables and relationships for the Senate Management System. The tables should map to the entities in the application, such as Admin, Members, Agendas etc.

5. Use MySQL queries: Use MySQL queries to retrieve data from the database and display it in the Senate Management System's views. You can use MySQL's rich set of query functions to filter, sort, and aggregate data.

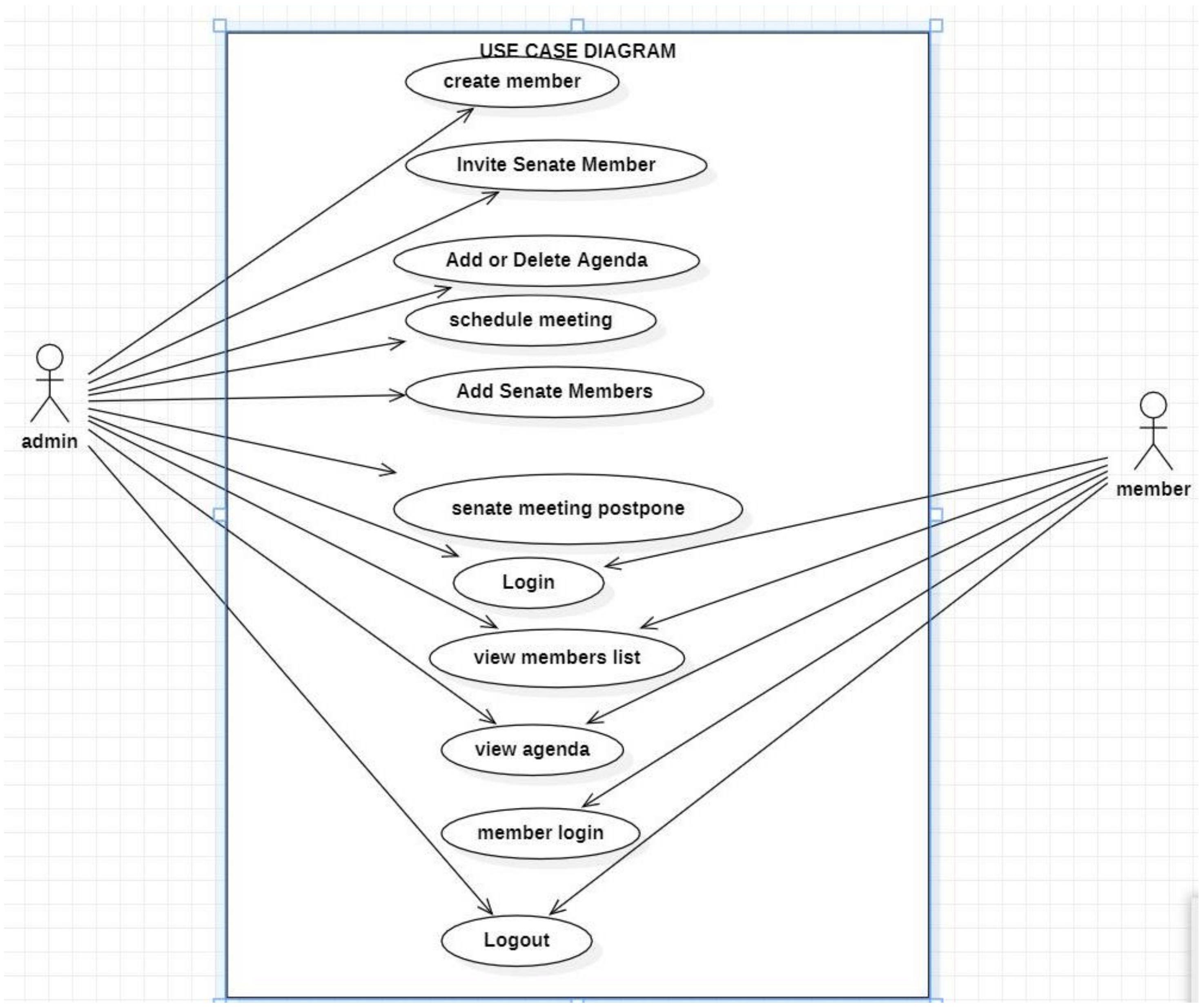
6. Store data: Use MySQL queries or ORM frameworks to store data entered in the Senate Management System.

By using MySQL in the Senate Management System, you can create a robust and scalable data management system that can store and retrieve large amounts of data efficiently. MySQL also provides features like data replication, backup and recovery, and performance tuning, making it a reliable choice for enterprise-grade applications.

CHAPTER 4

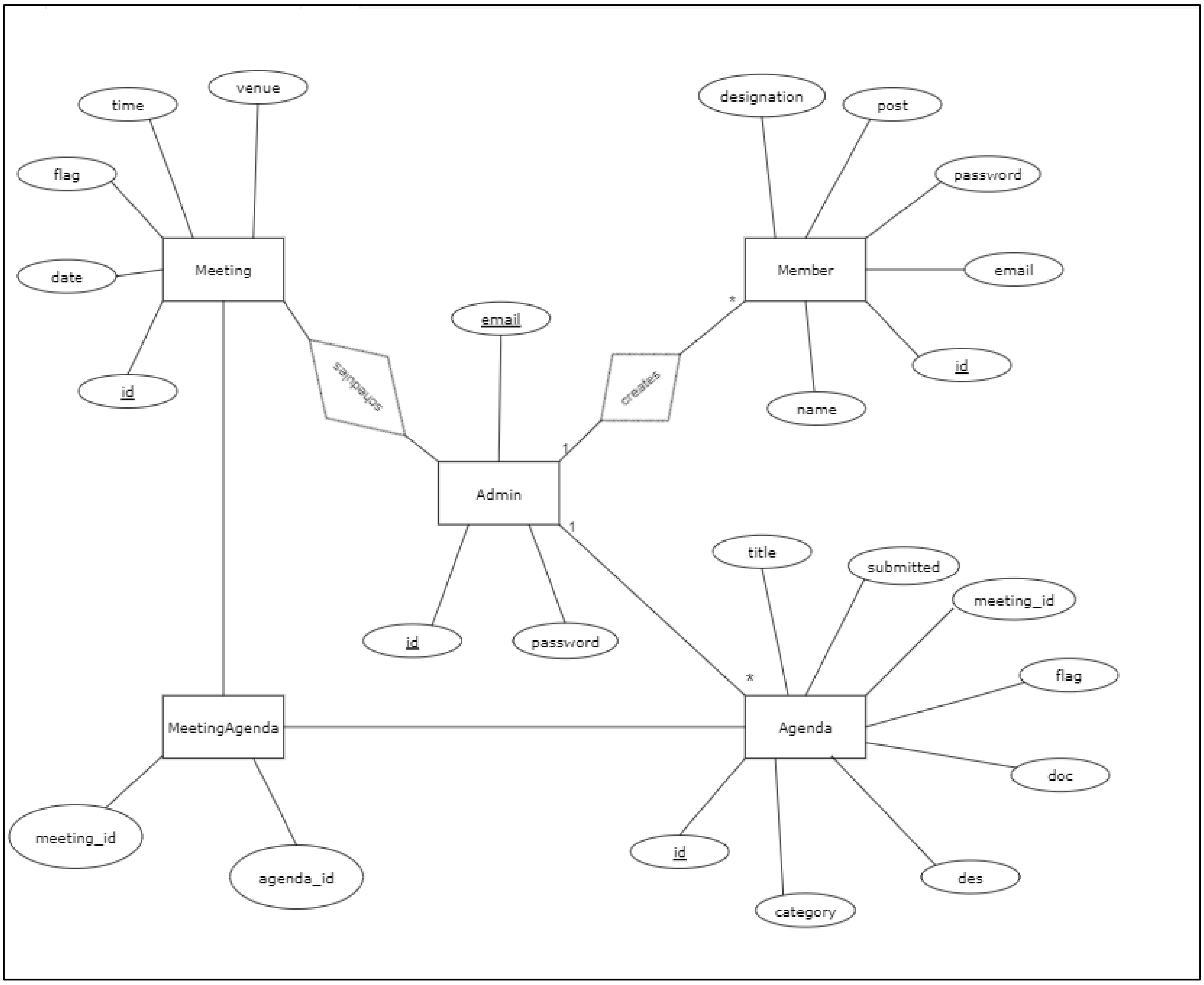
SYSTEM DESIGNING

4.1 Use Case diagram



4.1.1 Use case diagram

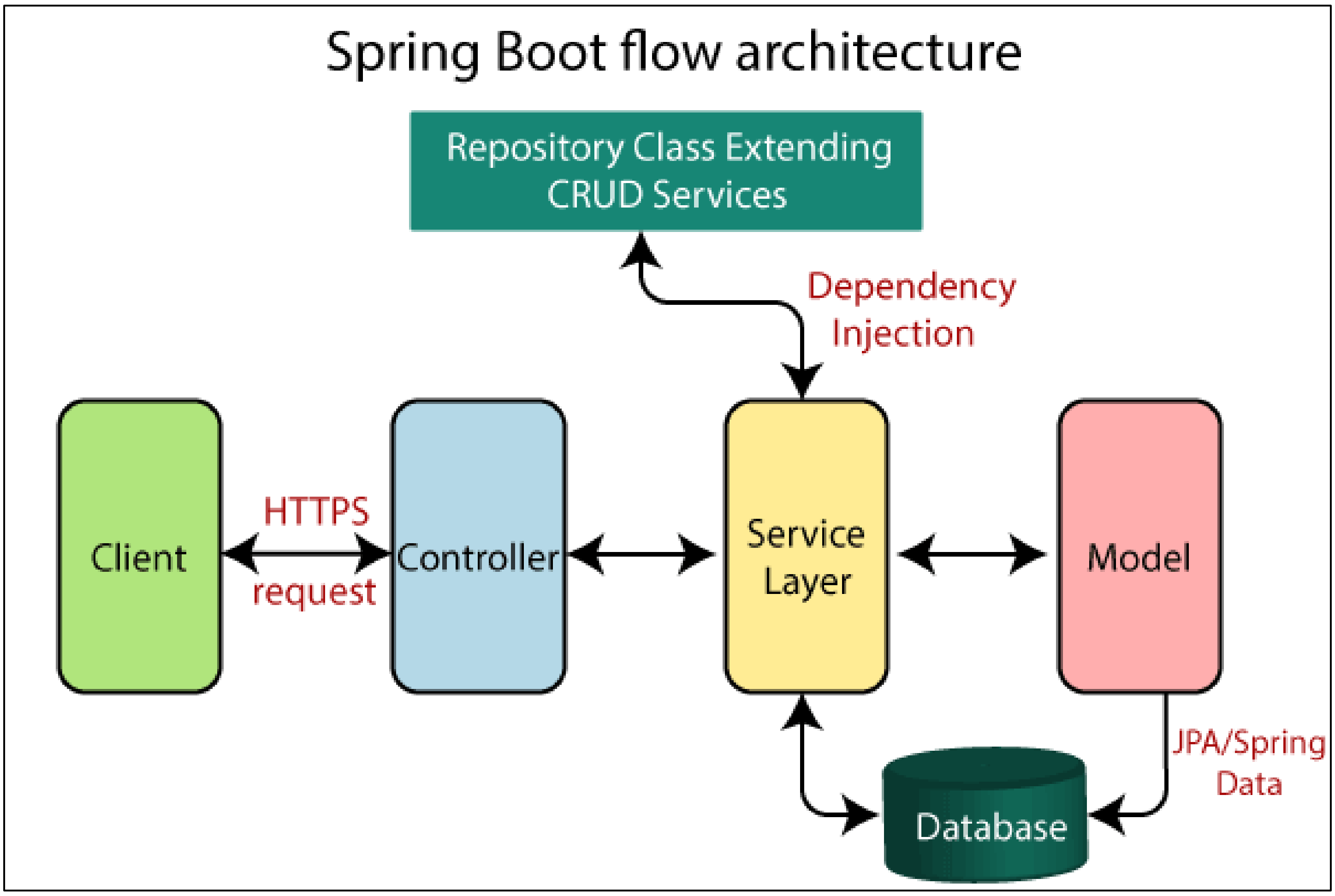
4.2 Entity Relationship diagram (ER diagram)



4.2.1 ER Diagram

4.3 Spring Framework Architecture Diagram

Spring Framework is an open-source application framework for building enterprise-grade Java applications. It provides comprehensive support for developing Java-based web applications, services, and microservices. Spring offers a broad set of features, including inversion of control, aspect-oriented programming, and support for popular data persistence frameworks like Hibernate and JPA.



4.3.1 Architecture diagram of Spring

CHAPTER 5
IMPLEMENTATION, TESTING AND RESULTS

5.1 Application Configuration

```
server.port=8089

spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html

#DataBase Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/senatemain
spring.datasource.username=root
spring.datasource.password=root@123
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.hibernate.ddl-auto=update

#Email Configuration
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=kankitha2110@gmail.com
spring.mail.password=hxtkzxdeclgivglh
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

After starting the application, we have to type the following URL in the browse:

http://localhost:8089

The port number used for the purpose of this project is 8089. It can be changed accordingly in the **application.properties .**

Then the website will be displayed.

5.2 pom.xml

The pom.xml file is an XML file that contains the configuration and dependencies for a Maven-based Spring Boot project. Maven is a popular build automation tool that manages dependencies, compiles code, runs tests, and packages the application into a JAR or WAR file. Here are some key features of the pom.xml file in Spring Boot:

1. Project configuration: The pom.xml file specifies the project configuration, such as the project name, version, description, and packaging type. This information is used by Maven to generate the project structure and manage the build process.

2. Dependencies: The pom.xml file lists all the dependencies required by the Spring Boot project. These dependencies can be Spring Boot starter dependencies, third-party libraries, or other modules within the same project.

3. Build plugins: The pom.xml file includes Maven plugins that perform tasks such as compiling code, running tests, and packaging the application into a JAR or WAR file. Spring Boot provides several Maven plugins that simplify the build process, such as the spring-boot-maven-plugin.

4. Parent POM: The pom.xml file can also inherit from a parent POM, which contains common configuration and dependencies for multiple projects. This allows developers to maintain a consistent build process and avoid duplication across projects.

5. Profiles: The pom.xml file can also define Maven profiles, which are sets of configuration and dependencies that can be activated based on the environment or build target. For example, a profile can define different database connections for development, testing, and production environments.

In summary, the pom.xml file in Spring Boot is a crucial component that defines the project's configuration, dependencies, and build process. It is used by Maven to manage the project's lifecycle and produce a deployable artifact.

```
<dependencies>
            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-web</artifactId>
            </dependency>

            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-test</artifactId>
                  <scope>test</scope>
            </dependency>
            <!--
https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-
jasper -->
            <dependency>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-jasper</artifactId>
            </dependency>

            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-data-jpa</artifactId>
            </dependency>
            <dependency>
                  <groupId>com.mysql</groupId>
                  <artifactId>mysql-connector-j</artifactId>
                  <scope>runtime</scope>
            </dependency>

            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-thymeleaf</artifactId>
            </dependency>
            <!--
https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-
devtools -->
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-devtools</artifactId>
            </dependency>

            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-validation</artifactId>
            </dependency
            <!--
https://mvnrepository.com/artifact/org.hibernate/hibernate-validator -->
            <dependency>
                <groupId>org.hibernate</groupId>
                <artifactId>hibernate-validator</artifactId>
                <version>8.0.0.Final</version>
            </dependency>

            <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-mail</artifactId>
            </dependency>
      </dependencies>
```

5.3 Controller Classes

5.3.1 Member Controller

```
@Controller
public class MemberController {

      @Autowired MemberRepository memberRepository;

      private String generatePassword() {
            int passwordLength=8;
            String
passwordChars="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ12345678
90";
            StringBuilder passwordBuilder=new StringBuilder();
            Random random=new Random();
            for(int i=0;i<passwordLength;i++) {
                  int index=random.nextInt(passwordChars.length());
                  passwordBuilder.append(passwordChars.charAt(index));
            }
            return passwordBuilder.toString();
      }

      //Admin Urls
      @GetMapping("/addmembers")
      public String showAddMemberForm(Model model,HttpSession session) {

            if(!session.getAttribute("privilege").equals("admin")) {
                  System.out.println(session.getId());
                  return "/";
            }

            System.out.println("Add members");
        model.addAttribute("member", new Member());
        return "add-member-form";
```

```
    }

      @PostMapping("/addmembers")
      public String addMember(@RequestParam(name = "status",required =
false) String status,@ModelAttribute("member") Member member,Model model,
HttpSession session) {

                  //check if user is authorized
                  if(!session.getAttribute("privilege").equals("admin")) {
                        System.out.println(session.getId());
                        return "/";
                  }

                  //check status message
                  if(status!=null) {
                        switch (status) {
                        case "empty":
                              model.addAttribute("message","Please fill out
all the fields.");
                              break;
                        case "unique":
                              model.addAttribute("message", "Email address
is already in the list.");
                      break;
                        case "success":
                              model.addAttribute("status", "Member added
successfully.");
                      break;
                        }
                  }

                  //Add member to database
                  else {
                        if (member.getName().isEmpty() ||
member.getEmail().isEmpty() || member.getDesignation().isEmpty() ||
member.getPost().isEmpty()) {
                        return "redirect:/addmembers?status=empty";
                    }
                    if
(memberRepository.findByEmail(member.getEmail())!=null) {
                        return "redirect:/addmembers?status=unique";
                    }
                    String password=generatePassword();
                    member.setPassword(password);
                    memberRepository.save(member);
                    return "redirect:/addmembers?status=success"; //
redirect with success message
                  }

                  // Return form with status message
                  return "add-member-form";
      }


      @GetMapping("/memberlist")
      public String getMembers(Model model, HttpSession session) {

            if(!session.getAttribute("privilege").equals("admin")) {
                  System.out.println(session.getId());
                  return "/";
            }
            List<Member> members=memberRepository.findAll();
            model.addAttribute("members", members);
            return "memberlist";//the name of the view file
      }

      @GetMapping("/removemember")
      public String removeMember(@RequestParam("id") Integer id,
HttpSession session) {

            if(!session.getAttribute("privilege").equals("admin")) {
                  System.out.println(session.getId());
                  return "/";
            }

            Optional<Member> member=memberRepository.findById(id);
            if(member.isPresent()) {
                  memberRepository.delete(member.get());
            }
            return "redirect:/memberlist";
      }

      @GetMapping("/updatemember")
      public String showUpdateForm(@RequestParam("id") Integer id,Model
model, HttpSession session) {

            if(!session.getAttribute("privilege").equals("admin")) {
                  System.out.println(session.getId());
                  return "/";
            }

            System.out.println("Show Update Form");
            Optional<Member> member=memberRepository.findById(id);
            if(member.isPresent()) {
                  model.addAttribute("member", member.get());
                  return "update-member-form";
            }
            else {
                  return "redirect:/memberlist";
            }
      }

      @PostMapping("/updatemember")
      public String processUpdateForm(@ModelAttribute("member") Member
member, HttpSession session) {

            if(!session.getAttribute("privilege").equals("admin")) {
                  System.out.println(session.getId());
                  return "/";
            }

            System.out.println("Process Update Form Begin");
            Member
existingMember=memberRepository.findById(member.getId());
            if(existingMember!=null) {
                  //System.out.println(existingMember.toString());
                  existingMember.setName(member.getName());
                  //System.out.println(existingMember.toString());
                  existingMember.setEmail(member.getEmail());
                  existingMember.setDesignation(member.getDesignation());
                  existingMember.setPost(member.getPost());
```

```
				memberRepository.save(existingMember);
				System.out.println("Process Update Form End");
				return "redirect:/memberlist";
			}
			else {
				return "redirect:/memberlist";
			}
	}

	//Member Urls
	@GetMapping("/memberlistm")
	public String showMemberList(Model model) {
		List<Member> members=memberRepository.findAll();
		model.addAttribute("members", members);
		return "memberlistm";//the name of the view file
	}

	@GetMapping("/memberpage")
	public String showMemberPage(HttpServletRequest request,Model
model,HttpSession session) {

		session=request.getSession();
		if(session.getAttribute("privilege")!=null &&
session.getAttribute("privilege").equals("member")){
			int memberId=(int) session.getAttribute("memberId");
			System.out.println("Member Page: "+memberId);
			System.out.println(session.getId());
			Member member=memberRepository.findById(memberId);
			model.addAttribute("member", member);
			return "memberpage";
		}
		else {
			return "redirect:/";
		}
	}
}
```

5.3.2 Meeting Controller

```
@Controller
public class MeetingController {

	@Autowired
	private MeetingService meetingService;

	@Autowired
	private MeetingRepository meetingRepository;

	@GetMapping("/adminpage")
	public String showAdminPage(Model model,HttpSession session) {
		//check user privilege
		if(!session.getAttribute("privilege").equals("admin")) {
			return "/";
		}

		//Fetch next meeting
		Optional<Meeting> meeting=meetingRepository.findByFlag(0);
		if(!meeting.isPresent()) {
			model.addAttribute("message","No meeting exists");
			return "adminpage";
```

```
            }

            //Calculate days left
            Meeting nextMeeting=meeting.get();
            LocalDate today=LocalDate.now();
            LocalDate meetingDate=nextMeeting.getDate();
            long daysLeft=ChronoUnit.DAYS.between(today, meetingDate);

            //Add model attributes
            model.addAttribute("meeting", nextMeeting);
            model.addAttribute("daysLeft", daysLeft);

            return "adminpage";
      }

      @GetMapping("/schedulemeeting")
      public String showScheduleMeetingForm(Model model, HttpSession
session) {

            if(!session.getAttribute("privilege").equals("admin")) {
                  System.out.println(session.getId());
                  return "/";
            }
            Optional<Meeting>
existingMeetings=meetingRepository.findByFlag(0);
            if(!existingMeetings.isEmpty()) {
                  model.addAttribute("meetingExists",true);
            }
            return "schedulemeeting";
      }

      @PostMapping("/schedulemeeting")
      public String submitScheduleMeetingForm(@ModelAttribute("meeting")
Meeting meeting,BindingResult bindingResult, Model model, HttpSession
session) {

            if(!session.getAttribute("privilege").equals("admin")) {
                  System.out.println(session.getId());
                  return "/";
            }

            if(bindingResult.hasErrors()) {
                  return "schedulemeeting";
            }
            try {
                  meetingService.addMeeting(meeting);
                  return "redirect:/schedulemeeting?status=success";
            }
            catch(RuntimeException ex) {
                  model.addAttribute("error", ex.getMessage());
                  return "schedulemeeting";
            }
      }

      @GetMapping("postpone")
      public String getPostponeMeetingForm(Model model, HttpSession
session) {

            if(!session.getAttribute("privilege").equals("admin")) {
                  System.out.println(session.getId());
                  return "/";
```

```
		}

		Optional<Meeting> meetings=meetingRepository.findByFlag(0);
		if(meetings.isEmpty()) {
			model.addAttribute("message","No meeting exists");
		}
		else {
			model.addAttribute("meetings",meetings);
			model.addAttribute("meeting", meetings.get());
		}
		return "postpone";
	}

	@PostMapping("postpone")
	public String postponeMeeting(@RequestParam("id") int id,
			@RequestParam("date") LocalDate date,
			@RequestParam("venue") String venue,
			@RequestParam("time") LocalTime time,Model model,
HttpSession session) {

		if(!session.getAttribute("privilege").equals("admin")) {
			System.out.println(session.getId());
			return "/";
		}

		Meeting meeting=meetingRepository.findById(id);
		meeting.setDate(date);
		meeting.setVenue(venue);
		meeting.setTime(time);
		meetingRepository.save(meeting);
		model.addAttribute("message","success");
		return "redirect:/postpone";
	}

	@GetMapping("meetings-list")
	public String getMeetings(Model model, HttpSession session) {

		if(!session.getAttribute("privilege").equals("admin")) {
			System.out.println(session.getId());
			return "/";
		}

		Optional<Meeting> meetings=meetingRepository.findByFlag(0);
		if(!meetings.isPresent())
			model.addAttribute("message","No Upcoming Meetings
Exist");
		else
			model.addAttribute("meetings", meetings.get());
		return "meetings-list";
	}

	@GetMapping("/completed")
	public String meetingCompleted(Model model, HttpSession session) {

		if(!session.getAttribute("privilege").equals("admin")) {
			System.out.println(session.getId());
			return "/";
		}

		Optional<Meeting> meetings = meetingRepository.findByFlag(0);
		Meeting meeting=meetings.get();
```

```
            meeting.setFlag(1);
          meetingRepository.save(meeting);
          model.addAttribute("message","No Upcoming Meetings Exist");
          return "adminpage";
        }

        @GetMapping("meetings-listm")
        public String getMeetingsList(Model model, HttpSession session,
HttpServletRequest request) {
            //check user privilege
            session=request.getSession();

            if(!session.getAttribute("privilege").equals("member")) {
                  System.out.println(session.getId());
                  return "/";
            }
            Optional<Meeting> meetings=meetingRepository.findByFlag(0);
            model.addAttribute("meetings", meetings.get());
            return "meetings-listm";
        }
}
```

5.3.3 Agenda Controller

```
@Controller
public class AgendaController {

        @Autowired
        private AgendaRepository agendaRepository;

        @Autowired
        private MeetingRepository meetingRepository;

        @GetMapping("/upload-agenda")
        public String showAgendaForm(Model model,HttpSession session) {

            if(!session.getAttribute("privilege").equals("admin")) {
                  System.out.println(session.getId());
                  return "/";
            }
            Optional<Meeting> meetings=meetingRepository.findByFlag(0);
            if(meetings.isPresent()) {
                  Meeting meeting=meetings.get();
                  model.addAttribute("meeting", meeting);
                  model.addAttribute("meetingFlag",true);
            }
            else
                  model.addAttribute("meetingFlag",false);
            return "upload-agenda";
        }

        @PostMapping("upload-agenda")
        public String uploadFile(@RequestParam("doc") MultipartFile file,
                  @RequestParam("title") String
title,@RequestParam("category") String category,
                  @RequestParam("desc") String
description,@RequestParam("sub") String submitted,RedirectAttributes
redirectAttributes,
                  HttpSession session) {
```

```
		if(!session.getAttribute("privilege").equals("admin")) {
			System.out.println(session.getId());
			return "/";
		}
		
		if(file.isEmpty()) {
			redirectAttributes.addAttribute("status", "file_error");
			return "redirect:/upload-agenda";
		}
		try {
			//Get the file and save it to local directory
			byte[] bytes=file.getBytes();
			Path 
path=Paths.get("C:\\Users\\Srinivas\\Desktop\\Ankitha\\MiniProject\\senatem
ain\\src\\main\\resources\\static\\agenda\\"+file.getOriginalFilename());
			Files.write(path, bytes);
			
			Optional<Meeting> 
meetings=meetingRepository.findByFlag(0);
			Meeting meeting=meetings.get();
			
			//Save file details to database
			Agenda agenda=new Agenda();
			agenda.setTitle(title);
			agenda.setCategory(category);
			agenda.setDescription(description);
			agenda.setDoc(file.getOriginalFilename());
			agenda.setSubmitted(submitted);
			agenda.setMeetingId(meeting.getId());
			agenda.setFlag(0);
			meeting.getAgendas().add(agenda);
			agendaRepository.save(agenda);
			redirectAttributes.addFlashAttribute("status", 
"success");
		}catch(IOException e){
			 redirectAttributes.addAttribute("status", "file_error");
		}
		return "redirect:/upload-agenda";
	}
	
	@GetMapping("/show-agenda")
	public String showAgenda(@RequestParam int id,Model model,HttpSession 
session) {
		
		if(!session.getAttribute("privilege").equals("admin")) {
			System.out.println(session.getId());
			return "/";
		}
		
		List<Agenda> 
agendas=agendaRepository.findByMeetingIdAndFlag(id,0);
		if(!agendas.isEmpty()) {
			model.addAttribute("id",id);
			model.addAttribute("agendas",agendas);
			return "show-agenda";
		}else {
			return "redirect:/meetings-list";
		}
	}
	
	@GetMapping("/agendalist")
```

```java
	public String getAgendaList(Model model, HttpSession session) {

		if(!session.getAttribute("privilege").equals("admin")) {
			System.out.println(session.getId());
			return "/";
		}

		Optional<Meeting> meetings=meetingRepository.findByFlag(0);
		Set<Agenda> filteredAgendas=new HashSet<>();
		if(meetings.isPresent()) {
			System.out.println(meetings.get().toString());
			Set<Agenda> agendas=meetings.get().getAgendas();
		for(Agenda agenda:agendas) {
			if(agenda.getFlag()!=1)
				filteredAgendas.add(agenda);
		}
		}
		model.addAttribute("meetings", meetings);
		model.addAttribute("agendas", filteredAgendas);
		System.out.println(filteredAgendas.size());
		return "agendalist";
	}

	@GetMapping("/delete/{id}")
	public String deleteAgenda(@PathVariable int id, HttpSession session)
{

		System.out.println("Delete Agenda");
		if(!session.getAttribute("privilege").equals("admin")) {
			System.out.println(session.getId());
			return "/";
		}

		Optional<Agenda> agendas=agendaRepository.findById(id);
		if(agendas.isPresent()) {
			Agenda agenda=agendas.get();
			/*
			List<MeetingAgenda>
meetingAgendas=meetingAgendaRepository.findByAgendaId(agenda.getId());
			for(MeetingAgenda meetingAgenda:meetingAgendas) {
				meetingAgendaRepository.delete(meetingAgenda);
			}*/
			agenda.setFlag(1);
			agendaRepository.save(agenda);
		}
		return "redirect:/agendalist";
	}

	@GetMapping("/show-agendam")
	public String showAgendaM(@RequestParam int id,Model
model,HttpSession session) {

		if(!session.getAttribute("privilege").equals("member")) {
			System.out.println(session.getId());
			return "/";
		}

		List<Agenda> agendas=agendaRepository.findByMeetingId(id);
		if(!agendas.isEmpty()) {
			model.addAttribute("id",id);
			model.addAttribute("agendas",agendas);
```

```
				return "show-agendam";
			}else {
				return "redirect:/meetings-list";
			}
		}
}
```

5.3.4 Email Controller

```
@Controller
public class EmailController {

	@Autowired
	private EmailService emailService;

	@Autowired
	private MemberService memberService;

	@GetMapping("/invitemember")
	public String inviteMember(@RequestParam Integer id, HttpSession
session, Model model) throws MessagingException{

		if(!session.getAttribute("privilege").equals("admin")) {
			System.out.println(session.getId());
			return "/";
		}

		//get the member by id
		Member member=memberService.getMemberById(id);

		//send the email
		String recipientEmail=member.getEmail();
		String subject="Invitation to the senate Meeting";
		String message = "Dear " + member.getName() + ",<br><br>"
				+ "We would like to invite you to join the team. It
is a pleasure to have you in the Senate.<br><br>"
				+ "Your login details:<br>"
				+ "Email: " + member.getEmail() + "<br>"
				+ "Password: " + member.getPassword() + "<br><br>"
				+ "Best regards,<br>"
				+ "The Team";
		emailService.sendEmail(recipientEmail, subject, message);

		model.addAttribute("message",member.getName());

		return "redirect:/memberlist";
	}
}
```

5.3.5 Login Controller

```
@Controller
public class LoginController {

	@Autowired
	private LoginService loginService;

	@PostMapping("/login")
	public String login(@RequestParam("email") String email,
@RequestParam("pass") String password, HttpServletRequest request) {
		return loginService.login(email, password, request);
	}

}
```

5.3.6 Main Controller

```
@Controller
public class MainController {
	@RequestMapping("/")
	public String index() {
		System.out.println("Index Page");
		return "index";
	}

	@GetMapping("/logout")
	public String logout(HttpServletRequest request) {
		System.out.println("Logout");
		HttpSession session=request.getSession();
		System.out.println(session.getId());
		session.invalidate();
		return "redirect:/";
	}

}
```

5.4 Service Classes

5.4.1 Email Service

```
@Service
public class EmailService {
	@Autowired
	private JavaMailSender mailSender;

	public void sendEmail(String recipientEmail,String subject,String 
message) throws MessagingException{
		MimeMessage mimeMessage=mailSender.createMimeMessage();
		MimeMessageHelper helper=new 
MimeMessageHelper(mimeMessage,"utf-8");
		helper.setTo(recipientEmail);
		helper.setSubject(subject);
		helper.setText(message,true);
		mailSender.send(mimeMessage);
	}
}
```

The above code is used to send invites to the members of the senate.

5.4.2 Login Service

```
public class LoginService {
	public String login(String email,String password, HttpServletRequest 
request) {
		Member member=memberRepository.findByEmail(email);
		if(member!=null) {
			if(password.equals(member.getPassword())) {
				HttpSession session=request.getSession();
				session.setAttribute("privilege","member");
				session.setAttribute("memberId", member.getId());
				return "redirect:/memberpage";
```

```
                        }
                }
                else {
                        Admin admin=adminRepository.findByEmail(email);
                        if(admin!=null) {
                                if(password.equals(admin.getPassword())) {
                                        HttpSession session=request.getSession();
                                        session.setAttribute("privilege", "admin");
                                        return "redirect:/adminpage";
                                }
                                else {

                                        request.setAttribute("error","Invalid email
or password");
                                        return "redirect:/";
                                }
                        }
                }
                request.setAttribute("error","Invalid email or password");
                return "redirect:/";
        }
}
```

The above piece of code is used to direct admins and members to their respective admin and member pages respectively.

5.4.3 Meeting Service

```
@Service
public class MeetingService {

        @Autowired
        private MeetingRepository meetingRepository;

        public List<Meeting> getAllMeetings(){
                return meetingRepository.findAll();
        }

        public void addMeeting(Meeting meeting) {
                Meeting existingMeeting
=meetingRepository.findById(meeting.getId());
                if(existingMeeting!=null) {
                        throw new RuntimeException("Upcoming Meeting already
exists");
                }
                meeting.setFlag(0);
                meetingRepository.save(meeting);
        }
}
```

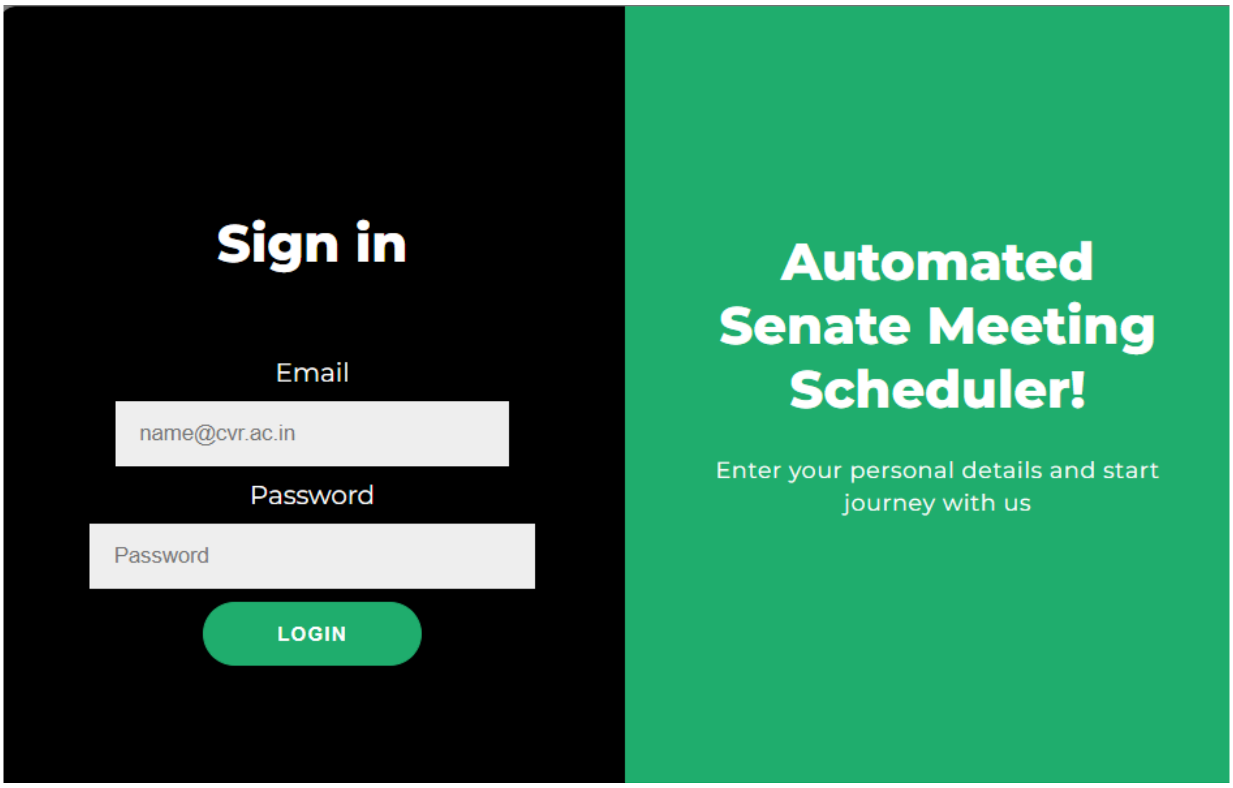
The above piece of code is used to add meetings

5.4.4 Member Service

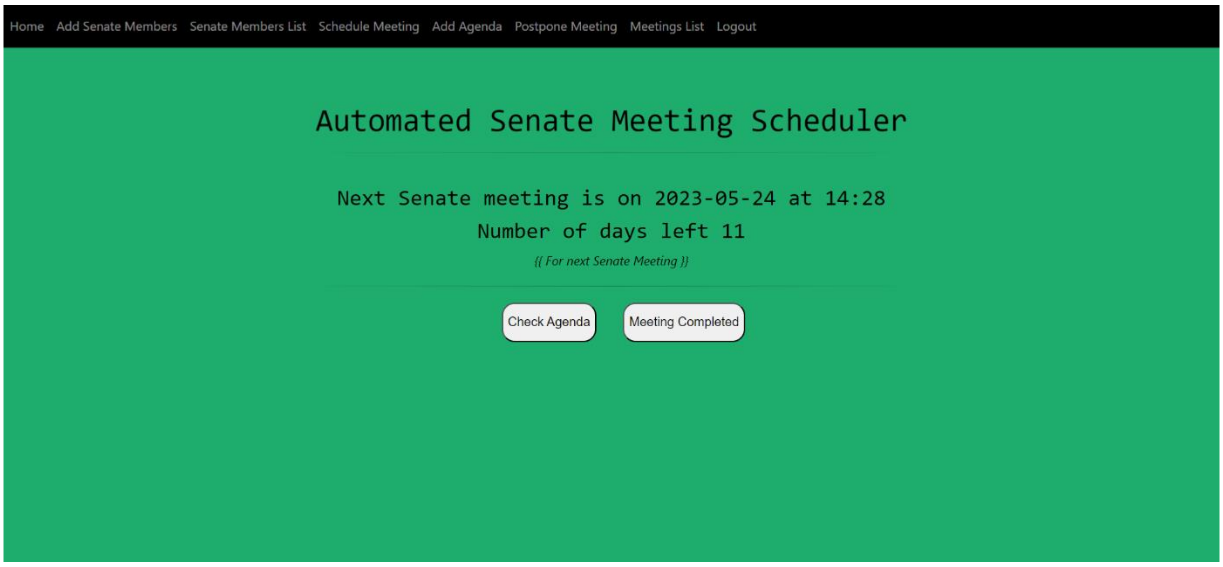
```
@Service
public class MemberService {
        @Autowired
        private MemberRepository memberRepository;

        public Member getMemberById(int id) {
                return memberRepository.findById(id);
        }
}
```

5.5 Testing


Sign in
Email
name@cvr.ac.in
Password
Password
LOGIN
Automated Senate Meeting Scheduler!
Enter your personal details and start journey with us

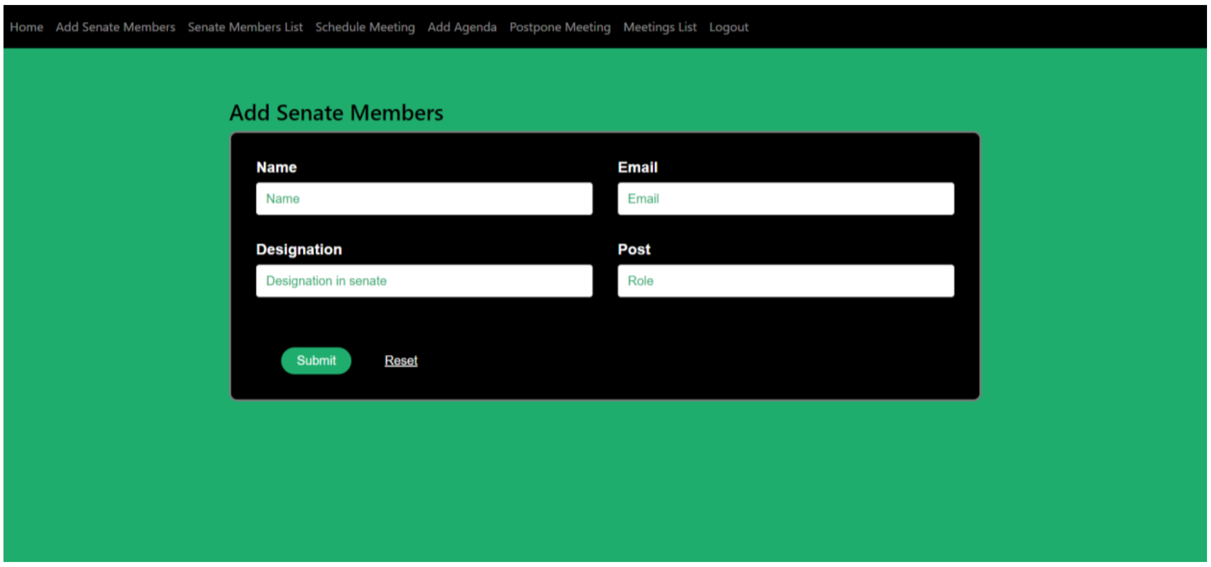
5.5.1 Login Page



Automated Senate Meeting Scheduler

Next Senate meeting is on 2023-05-24 at 14:28
Number of days left 11
{{ For next Senate Meeting }}

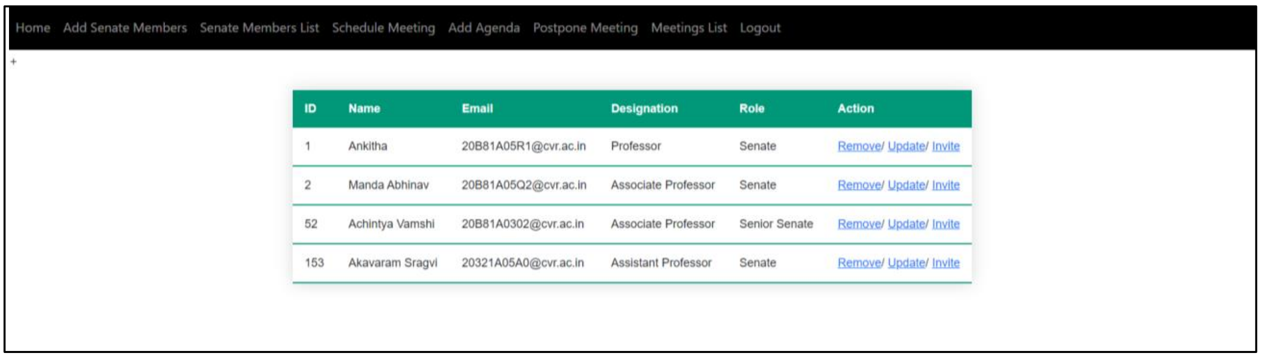
5.5.2 Admin Home Page


Home Add Senate Members Senate Members List Schedule Meeting Add Agenda Postpone Meeting Meetings List Logout

Add Senate Members

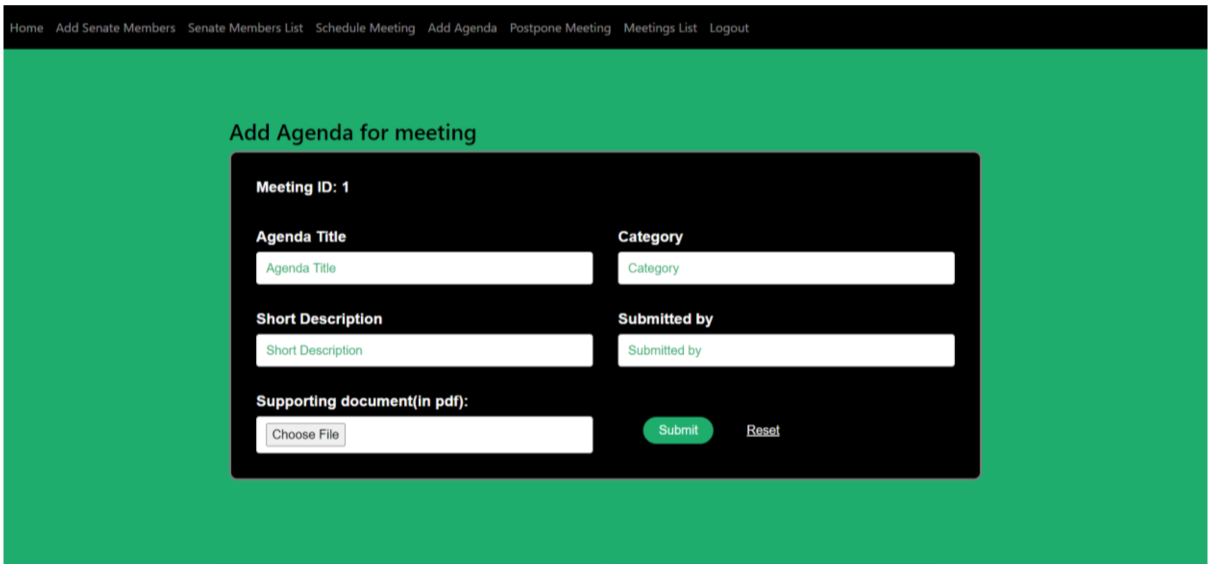
Name: Name	Email: Email
Designation: Designation in senate	Post: Role

5.5.3 Add Senate Member Page


Home Add Senate Members Senate Members List Schedule Meeting Add Agenda Postpone Meeting Meetings List Logout

ID	Name	Email	Designation	Role	Action
1	Ankitha	20B81A05R1@cvr.ac.in	Professor	Senate	Remove/ Update/ Invite
2	Manda Abhinav	20B81A05Q2@cvr.ac.in	Associate Professor	Senate	Remove/ Update/ Invite
52	Achintya Vamshi	20B81A0302@cvr.ac.in	Associate Professor	Senior Senate	Remove/ Update/ Invite
153	Akavaram Sragvi	20321A05A0@cvr.ac.in	Assistant Professor	Senate	Remove/ Update/ Invite

5.5.4 Senate Member List Page

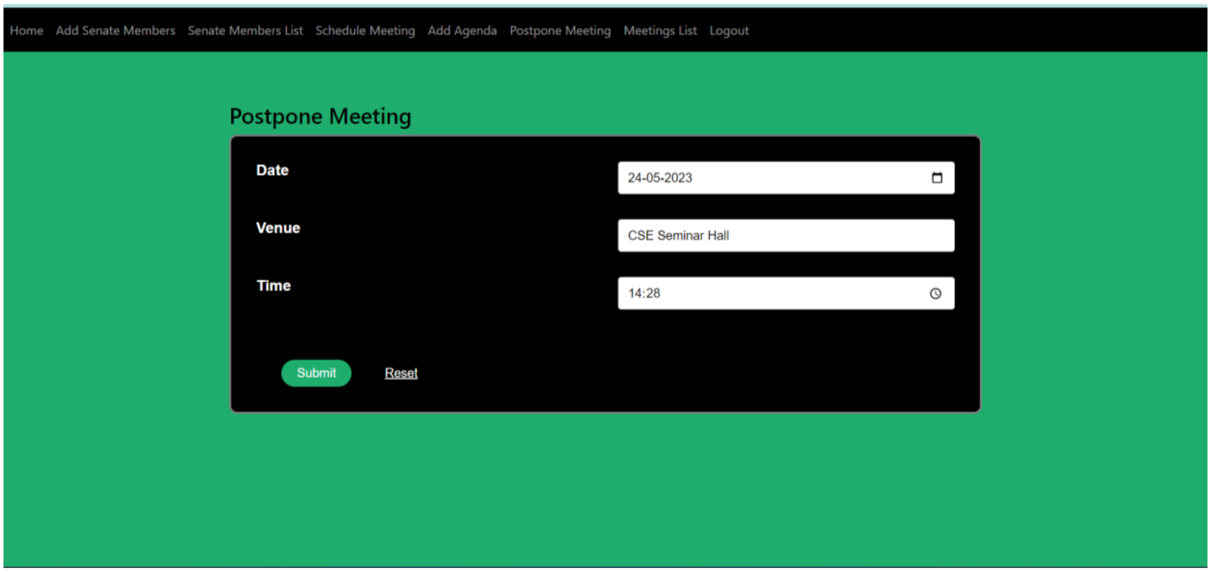

Home Add Senate Members Senate Members List Schedule Meeting Add Agenda Postpone Meeting Meetings List Logout

Add Agenda for meeting

Meeting ID: 1

Agenda Title: Agenda Title	Category: Category
Short Description: Short Description	Submitted by: Submitted by
Supporting document(in pdf): Choose File	

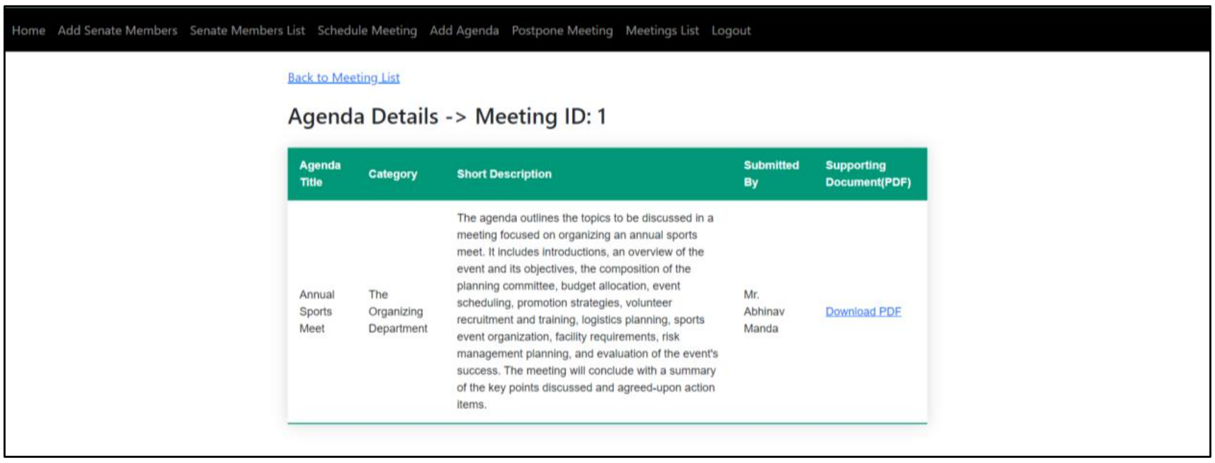
5.5.5 Add Agenda Page


Home Add Senate Members Senate Members List Schedule Meeting Add Agenda Postpone Meeting Meetings List Logout
Submit
Reset

Postpone Meeting

Date	24-05-2023
Venue	CSE Seminar Hall
Time	14:28

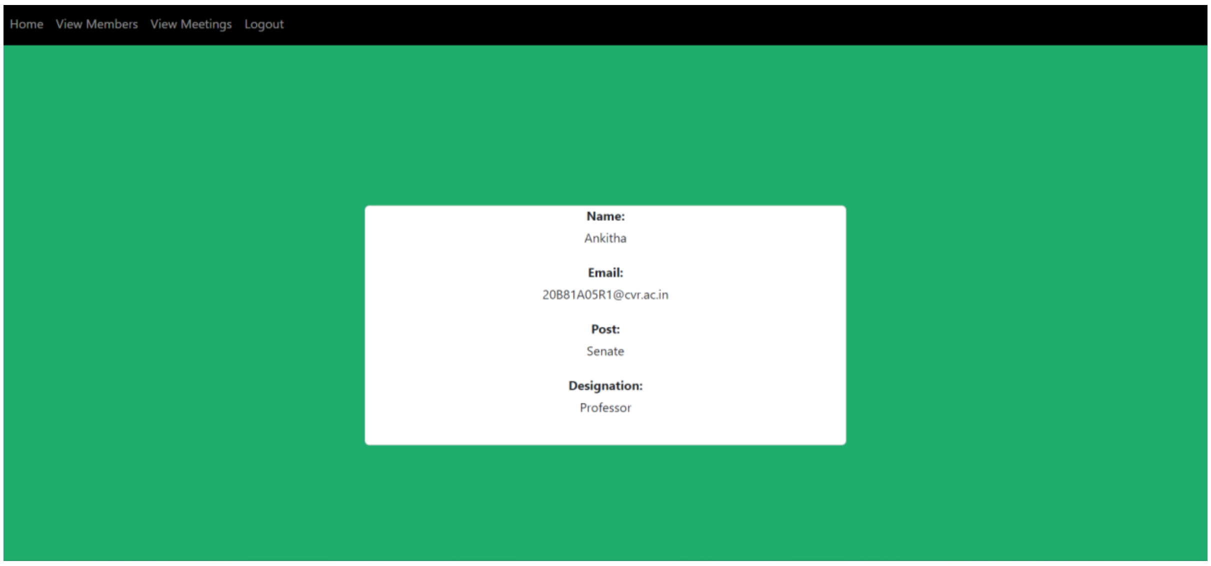
5.5.6 Postpone Meeting Page


Home Add Senate Members Senate Members List Schedule Meeting Add Agenda Postpone Meeting Meetings List Logout
Back to Meeting List

Agenda Details -> Meeting ID: 1

Agenda Title	Category	Short Description	Submitted By	Supporting Document(PDF)
Annual Sports Meet	The Organizing Department	The agenda outlines the topics to be discussed in a meeting focused on organizing an annual sports meet. It includes introductions, an overview of the event and its objectives, the composition of the planning committee, budget allocation, event scheduling, promotion strategies, volunteer recruitment and training, logistics planning, sports event organization, facility requirements, risk management planning, and evaluation of the event's success. The meeting will conclude with a summary of the key points discussed and agreed-upon action items.	Mr. Abhinav Manda	Download PDF

5.5.7 Agenda List Page



Name:
Ankitha

Email:
20B81A05R1@cvr.ac.in

Post:
Senate

Designation:
Professor

5.5.8 Member Home Page

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

In conclusion, the Senate Management System developed using Spring Boot is a valuable tool for managing various tasks and activities within a Senate. The system's user-friendly interface allows Senators to easily create, review, track committee activities, manage members, send invites and more. The project has been successful in achieving its objectives of streamlining Senate management processes and improving efficiency.

The system has demonstrated great potential for improving the way Senate business is conducted, and can serve as a model for similar systems in the future.

6.2 Future Scope

The future scope of the project includes adding more features such as integration with external systems, additional data visualization tools, and more robust security measures to ensure the confidentiality of sensitive information. Moreover, there is also potential for further expansion of the system beyond the Senate, such as adapting it for use in other legislative bodies or organizations with similar needs for efficient management of tasks and activities

A mobile application can be developed to allow Senators to work on the go, and multi-language support can be implemented to improve the user experience for Senators from different regions and countries. To ensure the confidentiality of sensitive information, robust security measures, such as two-factor authentication and data encryption, can be implemented. Finally, the system can be adapted for use in other legislative bodies or organizations with similar needs for efficient management of tasks and activities, opening up new markets and potential revenue streams.

CHAPTER 7

REFERENCES

1. Bootstrap. https://getbootstrap.com/

2. Html. https://developer.mozilla.org/en-US/docs/Web/HTML

3. Introduction to Spring Framework. (n.d.-b). https://docs.spring.io/spring-framework/docs/3.0.x/reference/overview.html