**EVENT SCHEDULER AND ASSISTANT**

**A SPRING BOOT WEB APPLICATION**

Submitted in partial fulfillment of the requirements of

**University of Mumbai**

For the Degree of

**Bachelor of Engineering in CSE (AIML / IOT)**

Submitted by

**Mr. Salimmiya Saudagar [Roll no. 62]**

**Mr. Sunny Yadav [Roll no. 77]**

**Mr. Prathamesh Zingade [Roll no. 78]**

Under the guidance of

**PROF. SONALI DESHPANDE**



**DEPARTMENT OF CSE (AIML/IOT)**

**SMT. INDIRA GANDHI COLLEGE OF ENGINEERING**

Ghansoli, Navi Mumbai - 400701

**Academic year: 2023-2024**

**Project Report Approval for S.E.**

This project report entitled

“**Event Scheduler and Assistant – A Spring Boot Web application**”

By

**Mr. Salimmiya Saudagar [Roll no. 60]**

**Mr. Sunny Yadav [Roll no. 77]**

**Mr. Prathamesh Zingade [Roll no. 78]**

are approved for the degree of Bachelor of Engineering in Computer Science and Engineering (Artificial Intelligence and Machine Learning), Semester IV, University of Mumbai.

|  |  |
| --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Examiner 1 | Examiner 2 |

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **Prof. Sonali Deshpande** |
| Internal Guide & Head of the Department |

Date: 30 October, 2023

Place: Ghansoli, Navi Mumbai

## Declaration

We declare that this written submission represents our own ideas in our own words and where others’ ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any act/data/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **Mr. Salimmiya Saudagar**  Roll no. 60 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **Mr. Sunny Yadav**  Roll no. 77 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **Mr. Prathamesh Zingade**  Roll no. 78 |

Date: 30 October, 2023

Place: Ghansoli, Navi Mumbai.

**Vision**

To provide employable CSE – AIML engineers for Society and Industry needs.

**Mission**

**M1:** Empower students with strong basic foundation.

**M2:** Develop technical and non-technical skills for lifelong learning.

**M3:** To promote student’s interest in higher studies, research and entrepreneurship to

Meet global challenges.

**PE01:** Graduates will be prepared for analyzing, designing, developing AIML based

Software with creativity.

**PE02:**  Graduates will be skilled in the use of the modern tools for problem solving and

Analyzing industrial and societal requirements.

**PE03:** Graduates will be exhibited professionalism, inter-personal skills and constant

Learning to develop management qualities.

**PS01:** Ability to analyze, design and develop applications by adopting the dynamic

Nature of Software developments.

**PS02:** Ability to use knowledge in Artificial Intelligence and Machine Learning to solve

Real world problems and identify the research gaps and render solutions with

Innovative ideas.

**ABSTRACT**

**Title:** Event Scheduler and Assistant – a web application

In the realm of event management, the "Event Scheduler and Assistant" is a dynamic web application tailored for the organization and execution of small-scale events. This project aims to provide an intuitive and user-friendly platform that assists event organizers in efficiently planning, scheduling, and coordinating events on a smaller scale.

At the heart of this web application lies its scalable and versatile database system. Users can seamlessly manage their events by creating, updating, and archiving records of past, present, and future occasions. This feature empowers users to keep a comprehensive history of their events, facilitating informed decision-making for subsequent endeavors. By incorporating a database-driven approach, the application offers a comprehensive overview of event-related details, including attendees, schedules, resources, and outcomes, thus enhancing the overall organization and productivity of small-scale events.

The Event Scheduler and Assistant goes beyond basic scheduling functionalities. It offers users the capability to create profiles, allocate tasks, and form dedicated teams for event execution. This collaborative element transforms the application into a one-stop hub for event planning, where team members can effortlessly communicate, collaborate, and track their progress. Task assignment, deadlines, and team coordination are seamlessly integrated, minimizing confusion and enhancing the collective efforts required for successful event management.

This project holds immense potential as a practical learning experience for those interested in web application development and its underlying fundamentals. Developing the Event Scheduler and Assistant involves delving into the realms of database management, user authentication, responsive design, and real-time collaboration.

**List of Abbreviations**

1. API: Application Programming Interface
2. JVM: Java Virtual Machine
3. JDK: Java Development Kit
4. JSP: Java Server Pages
5. JSTL: Java Server Pages Standard Tag Library
6. UI: User Interface
7. SQL: Structured Query Language
8. HTML: Hypertext Markup Language
9. CSS: Cascading Style Sheets
10. HTTP: Hypertext Transfer Protocol

**List of Figures**

1. Constraints Layout
2. Media Flowchart
3. Splash Screen
4. Home Page
5. Description Activity
6. Pdf viewer
7. Media Player

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr. no.** | **Title** | | | **Page no.** |
| 1 | Certificate | | | 1 |
| 2 | Project Report Approval | | | 2 |
| 3 | Declaration | | | 3 |
| 4 | Vision & Mission | | | 4 |
| 5 | Abstract | | | 5 |
| 6 | Lists   1. Abbreviations 2. Figures 3. Tables | | | 6 |
| 7 | Introduction | | | 8 |
| i. | Problem Statement | |
| ii. | Approach | |
| iii. | Objectives & Scope | |
| iv. | Features | |
| 8 | Theory | | | 11 |
| i. | Web Application | |
| ii. | Java & OOPs | |
| iii. | Spring, Spring Boot | |
| 10 | Methodology & Process | | | 14 |
| 11 | Walkthrough   1. System Design & User Interface | | | 19 |
| 12 | Analysis | | | 21 |
| 1 | | Plagiarism of the report |
| 2 | | System Requirements |
| 3 | | Scalability |
| 4 | | Future Scope |
| 13 | Conclusion | | | 22 |
| 14 | Bibliography | | | 23 |
| 15 | Acknowledgement | | | 24 |

**INTRODUCTION**

In the fast-paced world of today's work life, the task of organizing events can quickly become overwhelming and stressful for many individuals. This complexity escalates when an event involves numerous elements, such as tasks, volunteers, and teams. To simplify this process and provide a comprehensive solution, we have developed a web application. This application is built on the Java Framework, utilizing Spring and Spring Boot, in conjunction with MySQL, HTML & CSS, JavaScript, and a selection of libraries.

The web application offers users the ability to log in with two distinct roles: User and Admin. Administrators have the privilege of creating events, defining tasks, and assigning them to team members who are also managed by the admin. Users, on the other hand, can access event details, collaborate on tasks, and monitor their progress.

After an event has been created, it appears in an event list. Admins can select a specific event from the list to access a dedicated event page. This page allows administrators to add, assign, or delete tasks, as well as update event particulars such as its name, venue, start and end dates, and status (upcoming, ongoing, or completed). Admins can also delete entire events if needed.

The application further facilitates the process by allowing users to choose from a range of skill sets, including graphic design, management, content curation, cinematography, decoration, discipline, food preparation, promotion, and media handling. Admins can then assign tasks based on these skill sets. Additionally, tasks can have deadlines, and specific skill sets may be required to complete them.

**Problem Statement:**

While organizing any events it is very crucial to prepare very well in advance. It becomes critical that the tasks and work load is properly assigned to appropriate volunteers. The work should align with the interests of the volunteer and there is a need to keep track of everything in a very simple and collaborative way.

**Approach:**

To address and resolve the issue mentioned above, we can make a web application which can assist in the organization and management of these events in a very simplistic manner. The web application will provide options to create events, list them, update and delete them. Once the event is created the next stage is to add tasks to it while mentioning the specific domain, they fall under for e.g., Management, Content Curation, Media etc. Once these is done, the further stage involves assigning these tasks to appropriate volunteers which is added to the event and carefully considering their skillset which is mentioned in their profile. Once that is done the user will be able to see the tasks in their page and the admin can see and update these tasks.

**Objectives:**

* To provide a web application to organize and manage events
* To provide a collaborative environment.
* To provide a simple user interface to keep track of every volunteer’s role
* To lessen the friction and miscommunication.

**Scope:**

* The web application can almost be applied to any kind of events.
* The application is simple and users from age group of 14 and above can use.
* The web application can be used by clubs, colleges, communities etc.

**Features:**

There are many features to this application. Few of them are mentioned briefly below. There are many features that has yet to be added and many are yet to be sophisticated. With our knowledge we are able to bring:

* A User Login web application made using spring boot.
* A User-friendly UI to organize and manage event.
* A web application that keeps track in assigning tasks.
* A Web application that helps in aligning the interests of the volunteer with the work load of any event.

**THEORY**

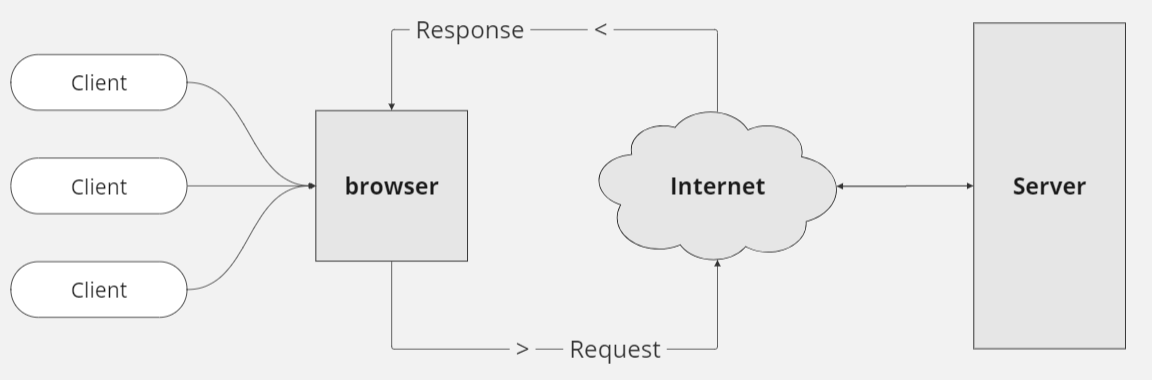
The web application with all its feature is possible due to 4 major technology which are Spring Boot, MySQL, HTML & CSS and Libraries like JSTL and many more. The backend part is made using Spring Boot while the UI is handled by the JSP pages. There are many tools that has been put to test while making this web application, the major two are GitHub and Docker. Let’s understand all these terminologies one by one starting with the What exactly is Web Application?

**Web-Application**  
Web applications are software programs that run on web servers and are accessed by users through web browsers over the internet. They are a type of application software specifically designed to be used over the World Wide Web. Web applications have become increasingly popular due to their accessibility and cross-platform compatibility.

**Web-Servers**  
A web server is a software application or hardware device that stores, processes, and serves web content to clients over the internet. It responds to client requests by providing web pages, files, or data.

**Web-Client**  
A web client, often referred to as a user agent, is a software application, typically a web browser, that requests and displays web content obtained from web servers. Common web clients include web browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, and Safari. These browsers are used to navigate the web, view websites, and interact with web applications.

The interaction between web servers and clients is based on the Hypertext Transfer Protocol (HTTP) or its secure version, HTTPS. Clients send HTTP requests to servers, which respond with HTTP responses, typically containing the requested web content. This communication allows users to access and interact with web resources on the internet.



**JAVA**  
Java is a popular, versatile, and high-level programming language known for its portability, robustness, and extensive libraries. It's widely used in various domains, including web development. Here are some key features that make Java a good choice for web applications:

* **Runs Everywhere:** Code written in Java can run on different devices and operating systems, ensuring consistent behavior.
* **Catches Errors Early:** It helps catch errors during development, making your web apps more reliable.
* **Offers Tools:** Java has libraries and frameworks to speed up development.
* **Handles Multiple Users:** Java's support for concurrency ensures your web app can handle many users at once.
* **Prioritizes Security:** It has built-in security features for authentication and data protection.
* **Scales Easily:** Java architecture supports scalable web apps for lots of users and transactions.
* **Is Reliable:** Java manages memory well and handles exceptions gracefully.
* **Has a Strong Community:** There's a large and active developer community for support.
* **Is Enterprise-Ready:** It's trusted for building big, complex web apps in various industries.
* **Stays Compatible:** Older Java apps can run on new Java versions with minimal changes.

**Object Oriented Programming in JAVA**Object-Oriented Programming (OOP) is a programming paradigm that is widely used in Java. OOP is centered around the concept of objects, which are instances of classes. Here are the key principles and concepts of OOP in Java that have been majorly used in the project “Event Scheduler and Assistant”.

**Classes and Objects:** In Java, everything is an object, and these objects are instances of classes. Classes are templates that define the structure and behavior of objects. Objects are created from classes and represent real-world entities with attributes (data) and methods (functions).

**Encapsulation:** Encapsulation is the concept of bundling data (attributes) and methods (functions) that operate on that data into a single unit called a class. It restricts access to some of an object's components and prevents the accidental modification of data.

**Inheritance:** Inheritance is a mechanism that allows a new class (a subclass or derived class) to inherit properties and behaviors from an existing class (a superclass or base class). In Java, the extends keyword is used to create a subclass.

public interface EventRepository extends JpaRepository<Event, Integer> {

public Event findByName(String name);

}

**Interfaces:** Interfaces define a contract that classes must adhere to. A class can implement one or more interfaces, ensuring that it provides specific methods defined in those interfaces.

**Polymorphism:** Polymorphism allows objects of different classes to be treated as objects of a common superclass. This promotes flexibility and reusability. In Java, polymorphism is achieved through method overriding and interfaces.

**Abstraction:** Abstraction is the process of simplifying complex reality by modeling classes based on essential attributes and behaviors. It allows you to focus on the relevant aspects of an object and ignore the irrelevant details.

**Methods:** Methods are functions defined within classes that operate on the data of those classes. Methods are an essential part of objects, and they define the behavior of the objects.

public String getDescription() {

return description;

}

**Constructors:** Constructors are special methods used for initializing objects when they are created. In Java, a constructor has the same name as the class and is called when an object is instantiated.

public class Event {

//default constructer

public Event() {}

}

**Access Modifiers:** Access modifiers (e.g., public, private, protected) control the visibility and accessibility of classes, methods, and fields. They help enforce encapsulation and access control.

**Java Spring Framework:**

The Java Spring Framework, commonly referred to as Spring, is an open-source application framework for building enterprise-level Java applications. It provides comprehensive infrastructure support for developing Java-based applications, particularly in the context of web development. Spring simplifies and accelerates the development process by offering a wide range of functionalities and features.

**Why it is:**

The Spring Framework was created to address several challenges faced by Java developers, including:

* Simplifying Java EE Development: Spring simplifies and streamlines Java Enterprise Edition (Java EE) development by providing a lightweight and non-intrusive alternative to the complex and sometimes cumbersome Java EE specifications.
* Modular Design: Spring adopts a modular design that allows developers to pick and choose the specific components they need for their projects. This modular approach makes it highly adaptable to various application scenarios.
* IoC Container: The Inversion of Control (IoC) container in Spring enables better management of application components and their dependencies. This helps improve maintainability and testability.
* AOP Support: Spring includes support for Aspect-Oriented Programming (AOP), making it easier to separate cross-cutting concerns, such as logging and security, from the main business logic.
* Simplifies JDBC and JEE: Spring simplifies the use of JDBC for database access and offers abstractions for many Java EE components, reducing the complexity of working with these technologies.

**Release Date:**

The Spring Framework was first released in March 2004.

**Use of Spring in Web Application**

Spring is a popular and versatile framework for building web applications in Java. It provides a comprehensive suite of features and modules for web development, making it suitable for a wide range of web applications, from simple websites to complex, enterprise-level systems.

Here are some key aspects of using Spring in web applications:

* **Spring Web Module**:  
  Spring's web module, often referred to as Spring Web, provides essential components and features for building web applications. It includes the Spring MVC framework, which is a crucial part of Spring's web support.
* **Spring MVC (Model-View-Controller):**

Spring MVC is a framework for building web applications in a structured and organized way. It follows the MVC architectural pattern, separating the application into three key components: the Model (business logic and data), the View (presentation layer), and the Controller (request handling and routing).

* **IoC and Dependency Injection:**

In Spring, Inversion of Control (IoC) and dependency injection are core concepts. These facilitate the management of components, their configurations, and their relationships. It allows for easier testing and better modularity of web application code.

* **Aspect-Oriented Programming (AOP):**

Spring supports AOP, which allows you to apply cross-cutting concerns, such as logging, security, and transactions, across different parts of your application. This is particularly useful in web applications where you may have common concerns that need to be applied uniformly.

* **Integration with Other Technologies:**

Spring provides integration with various technologies often used in web development, such as Hibernate, JPA, JMS, and more. This helps you manage database access, messaging, and other aspects of web applications seamlessly.

* **Security:**

Spring Security is a module that helps secure web applications. It provides features for authentication, authorization, and protection against common security threats, making it essential for securing web applications.

* **RESTful Web Services:**

Spring provides support for building RESTful web services through Spring Web MVC or Spring WebFlux. You can easily expose REST APIs, handle requests, and format responses using Spring's REST features.

* **View Technologies:**

Spring supports various view technologies, including JSP, Thymeleaf, FreeMarker, and others. You can choose the one that best suits your needs and preferences.

* **Testing Support:**

Spring's testing framework allows for unit testing and integration testing of web applications, which is crucial for maintaining application quality.

* **Spring Boot:**

Spring Boot, an extension of the Spring framework, simplifies the development of web applications even further. It provides auto-configuration, embedded web servers, and various starter templates, making it easy to create stand-alone, production-ready web applications.

* **Dependency Management:**

Spring's built-in container for managing dependencies simplifies the integration of various libraries and frameworks used in web development.

* **Internationalization and Localization:**

Spring supports internationalization and localization, making it easier to build web applications that cater to a global audience.

* **Externalized Configuration:**

Spring allows you to externalize configuration, which is especially important in web applications that may need different configurations for various environments (e.g., development, testing, production).

Spring's flexibility and modularity make it suitable for web applications of all sizes, and its strong community and ecosystem provide extensive resources and support for developers building web applications in Java.

**Spring Boot**

Spring Boot was created to simplify the process of developing production-ready, stand-alone, and production-grade Spring-based applications. It builds on top of the Spring Framework, providing a set of conventions and tools that make it easier to set up, configure, and deploy Spring applications.

Figure Spring Boot



[This Photo](https://alexkosarev.name/category/javaee/spring/) by Unknown Author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)

The Spring Framework, a long-standing favorite for Java-based enterprise applications, offered a robust and modular platform. However, it posed challenges for newcomers due to the complexity of setup and the need for extensive configuration and boilerplate code even for basic applications.

The demand for more agile and lightweight development in the era of microservices spurred the need for simplified Spring application development. Developers sought an easier way to begin with Spring, focusing on business logic rather than grappling with infrastructure intricacies.

This demand led to the birth of Spring Boot in 2014, when Pivotal Software (now part of VMware) officially introduced it. Spring Boot aimed to provide a hassle-free approach to Spring application development. It introduced a "no-xml" or "no-configuration" methodology, offering sensible defaults and automatic configurations. This made it a breeze for developers to kickstart projects with minimal effort and gradually customize applications according to their needs. Spring Boot's key features included auto-configuration, embedded servers, standalone application packaging, built-in production-ready support, and starter templates for various common use cases, simplifying the setup of features like security and data access.

**Annotations**

Annotations are a powerful feature in Java that allow developers to add metadata or configuration information to classes, methods, fields, or other program elements. In the context of Java frameworks and libraries, annotations are extensively used to simplify configuration, improve code readability, and enable various functionalities. Annotations are prefixed with the "@" symbol and are processed by the Java compiler, runtime, or frameworks.

Now, let's discuss some common annotations and their use cases in the context of Spring:

**@Controller:**

Use Case: This annotation is used in Spring MVC to mark a class as a controller. It handles incoming HTTP requests and maps them to methods within the class.

@Controller

@SessionAttributes("username")

public class EventController {}

**@RequestMapping:**

Use Case: It is used in Spring MVC to map HTTP requests to specific controller methods. Developers can define URL patterns, HTTP methods, and other request parameters.

**@Valid:**

Use Case: Often used in Spring, it validates the input data or model objects in controller methods. It ensures that the data adheres to predefined rules, such as data types, required fields, and other constraints.

**@Entity:**

Use Case: In Java Persistence API (JPA), this annotation marks a class as a persistent entity. It is used to map Java objects to database tables and columns.

**@Id:**

Use Case: Also, part of JPA, it marks a field as the primary key of an entity. It is used to uniquely identify records in the database.

**@GeneratedValue:**

Use Case: Often used in combination with @Id in JPA, it indicates that the primary key value should be automatically generated by the database (e.g., auto-incremented).

**@RestController:**

Use Case: In Spring, this annotation combines @Controller and @ResponseBody. It simplifies the creation of RESTful web services, as it automatically converts the response to JSON or XML.

**@Configuration:**

Use Case: This annotation is used in Spring to define a configuration class that provides bean definitions. It's particularly useful for configuring various aspects of the Spring application.

**@Bean:**

Use Case: In Spring, it is used within a configuration class to define a method that produces a bean managed by the Spring container.

**@Repository:**

Use Case: Commonly used in Spring for data access classes, it indicates that the class should be treated as a repository, allowing for automatic bean creation and exception translation.

**@Service:**

Use Case: In the Spring framework, it marks a class as a service or business logic component. It's often used in combination with @Autowired to inject dependencies.

These are just a few examples of the many annotations available in Java and Java frameworks. Annotations play a crucial role in simplifying configuration and enhancing the functionality of Java applications, making development more efficient and maintainable.