**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.

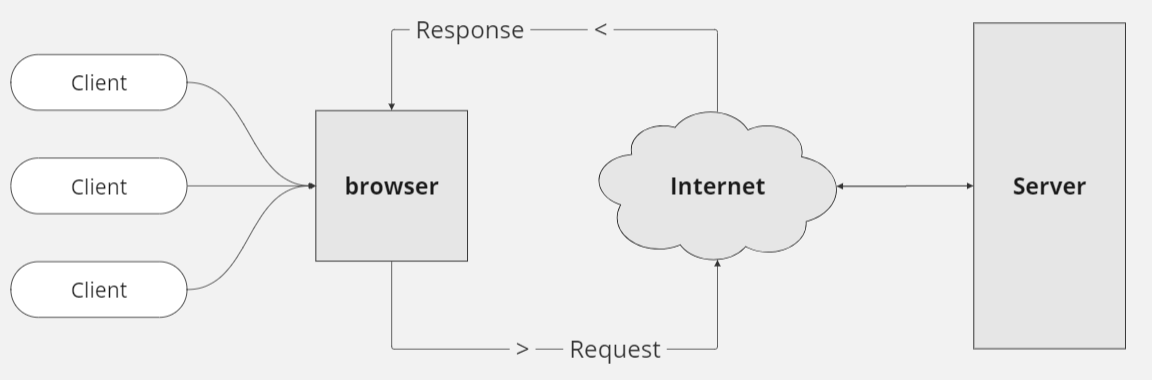
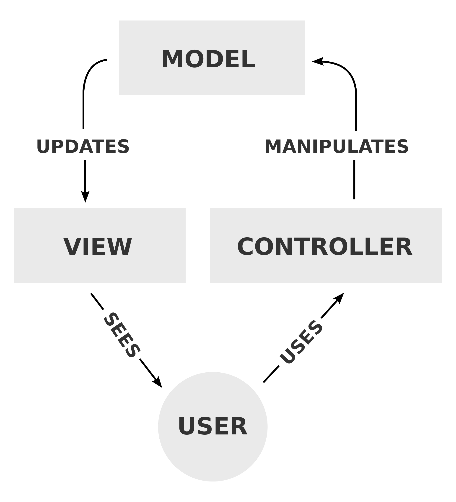


Figure Web Development

**Fundamentals**:

**MVC (Model-View-Controller)**: MVC is a software design pattern commonly used in web development. It separates an application into three interconnected components: Model (data and business logic), View (user interface), and Controller (handles user input and interacts with both Model and View).

Figure MVC



[This Photo](http://stackoverflow.com/questions/29594105/mvc-is-it-model-to-view-or-controller-to-view) by Unknown Author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)

**Dispatcher Servlet**: In the context of Spring MVC, the Dispatcher Servlet is the front controller that receives all incoming HTTP requests and directs them to the appropriate controller for processing. It plays a central role in managing the flow of a Spring web application.

**URL (Uniform Resource Locator)**: A URL is a web address that specifies the location of a resource on the internet. It typically consists of the protocol (e.g., HTTP or HTTPS), domain name, path, and sometimes query parameters.

**Request Mapping**: Request mapping in Spring MVC is the process of associating a specific HTTP request to a particular controller method. You define these mappings using annotations like @RequestMapping in Spring, indicating which method should handle a specific URL.

**HTTP (Hypertext Transfer Protocol)**: HTTP is the protocol used for communication between a client (e.g., a web browser) and a server over the internet. It defines how requests and responses should be formatted and transmitted.

**Ports**: Ports in the context of networking and web development refer to the specific endpoint on a machine where network services listen for incoming connections. In web development, port 80 is commonly used for HTTP, and port 443 is used for HTTPS.

**Localhost**: In networking, "localhost" refers to the current device or machine you are using. When you access a web server running on your own machine, you can use "localhost" as the hostname, which is equivalent to the loopback IP address (127.0.0.1). This allows you to access web applications on your local server.

**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.

@RequestMapping(value = "event-view", method = RequestMethod.POST)

private String postEventView(ModelMap model, @Valid Event event) {

return "redirect:event-list";

}

**@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).

@Entity

public class Task {

@Id

@GeneratedValue

private int id;}

**@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.

**Different Components of the Project File:**

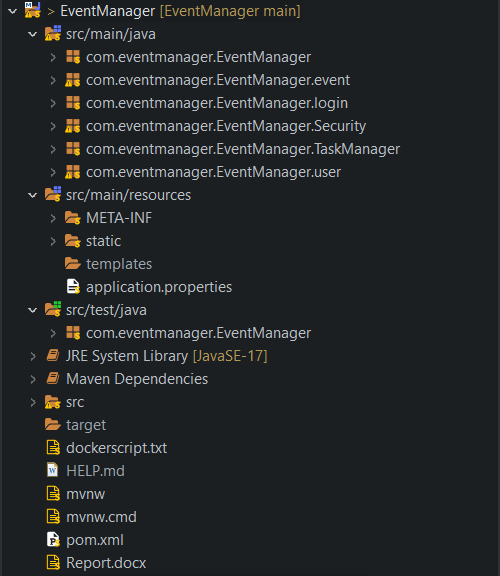
****

Figure Project directory

In a Spring project, various files and directories play different roles.

* **main/java**: This directory typically contains your Java source code, including Spring components, services, controllers, and other classes that make up the core functionality of your application.
* **test/java**: This directory contains your unit tests for the classes in the main/java directory. It helps ensure the quality and correctness of your code.
* **main/resources**: This directory typically includes configuration files, property files, XML configuration, and other non-Java resources that are used by your Spring application.
* **META-INF**: This directory contains metadata information about your application, such as configuration files for Spring (e.g., spring-context.xml), or configuration files for Java EE components, like persistence.xml for JPA.
* **jsp**: This directory often contains JavaServer Pages (JSP) files used for rendering views in web applications built with Spring MVC.
* **common**: The "common" directory is not a standard part of a Spring project. It could be a custom directory used for storing shared resources or code common to multiple parts of your application.
* **static**: This directory typically contains static resources such as CSS, JavaScript, and images that are used in web applications. These resources are served directly by the web server without being processed by the Spring application.
* **application.properties (or application.yml):** This is a configuration file where you specify application-level settings, such as database connection details, server ports, and various properties that the Spring application needs.
* **JRE System Library**: This is a reference to the Java Runtime Environment (JRE) or Java Development Kit (JDK) that your project is configured to use. It specifies the version of Java that your project relies on.
* **Maven Dependencies**: This is a directory that contains all the JAR files and external dependencies specified in your project's pom.xml file. Maven manages these dependencies and automatically downloads them from repositories.
* **pom.xml**: This is the Project Object Model (POM) file for your Maven-based Spring project. It defines project configuration, dependencies, and build settings.

**Dependencies used by Spring Boot:**

In a Maven project, dependencies are external libraries or modules that your project relies on. These dependencies are specified in the *pom.xml* file, and Maven automatically downloads and manages these dependencies from remote repositories. The *pom.xml* file is essentially a manifest of your project's configuration, including its dependencies.

**org.springframework.boot**:   
This dependency is for Spring Boot, which is a framework that simplifies the development of Spring applications. It provides pre-configured settings and a variety of libraries to help you quickly build production-ready Spring applications.

**com.mysql**:   
This is the MySQL database driver, allowing your Java application to connect to a MySQL database. It provides the necessary classes to interact with MySQL databases.

**org.apache.tomcat.embed**:   
In the context of Spring Boot, this dependency includes the embedded Apache Tomcat servlet container. It allows you to run your Spring Boot web application as a self-contained, standalone application with an embedded web server.

**jakarta.servlet.jsp.jstl**:   
This dependency includes the JavaServer Pages Standard Tag Library (JSTL) for Jakarta EE. JSTL provides a set of custom tags and functions to simplify JSP development by reducing Java code within JSP pages.

**org.webjars**:   
WebJars is a project that packages client-side web libraries (like JavaScript and CSS) into JAR files, making it easy to manage and include these resources in your web applications. This dependency is used to include web resources from WebJars.

**org.eclipse.jetty**:   
Jetty is a lightweight and highly scalable Java-based web server and servlet container. You can use it as an alternative to other embedded servlet containers (like Tomcat) for running your web applications in Spring Boot.

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter- validation</artifactId>

</dependency>

**JSP**

JSP (JavaServer Pages) is a web development technology that enables dynamic content generation. It integrates Java code into HTML templates, executes server-side operations, and provides custom tag libraries for common tasks. JSP plays a crucial role in the Model-View-Controller (MVC) architecture, where it seamlessly interacts with Java components. Designers focus on the presentation layer, while developers handle business logic, simplifying maintenance. It's highly extensible and well-supported by IDEs. To run JSP, a servlet container like Apache Tomcat is needed. While JSP remains valuable, alternatives like Spring MVC and Thymeleaf offer modern features and flexibility for dynamic web page creation.

**MySQL**

Figure MySQL



[This Photo](https://blog.toright.com/posts/1214/mysql-%E6%96%B0%E5%A2%9E%E4%BD%BF%E7%94%A8%E8%80%85%E8%88%87%E6%AC%8A%E9%99%90%E8%A8%AD%E5%AE%9A-%E7%AD%86%E8%A8%98.html) by Unknown Author is licensed under [CC BY-ND](https://creativecommons.org/licenses/by-nd/3.0/)

MySQL is a popular open-source relational database management system (RDBMS) known for its speed, reliability, and scalability. Developed by Oracle Corporation, it is widely used in web applications to store and manage data. MySQL supports SQL (Structured Query Language), making it accessible for developers to interact with databases. It is particularly suited for web development due to its ability to handle large datasets, and it's commonly used with server-side technologies like PHP, Java, and Python. MySQL's ACID compliance ensures data consistency and reliability, while its support for InnoDB storage engine offers transaction support. It also integrates seamlessly with various web frameworks and platforms, making it a preferred choice for web developers.

**MySQL in Web Development**

In web development, MySQL is a preferred choice for storing, retrieving, and managing data. It plays a crucial role in web applications, content management systems, e-commerce platforms, and more. It integrates seamlessly with server-side technologies and frameworks, allowing developers to create dynamic, data-driven websites. MySQL's speed and scalability are particularly valuable for web applications with high traffic and complex data requirements. Its support for transactions ensures data integrity, making it a reliable choice for applications that involve financial transactions or user data. MySQL's compatibility with various programming languages and web frameworks makes it an essential tool for web developers.

**JDBC**

JDBC (Java Database Connectivity) is a Java-based API that enables Java applications to interact with databases. It provides a standardized way to connect, query, and manipulate data in relational databases. JDBC drivers, specific to each database, facilitate communication between the Java application and the database server. Developers use JDBC to establish connections, execute SQL queries, retrieve results, and manage transactions programmatically. This API is an integral part of Java-based web development, often used in conjunction with frameworks like Spring.

JDBC offers flexibility, allowing developers to work with various databases without significant code changes. It's particularly valuable for web applications where data persistence and retrieval are essential, ensuring that data-driven web applications can interact seamlessly with relational databases. By providing a bridge between Java and databases, JDBC simplifies the process of developing web applications that rely on database storage.

**Spring Data JPA**

Spring Data JPA is a part of the broader Spring Data project that simplifies data access in Java applications. It's built on top of the Java Persistence API (JPA) and provides a higher-level, more abstracted approach to working with databases. Spring Data JPA simplifies data access by reducing the amount of boilerplate code required to interact with databases.

In web development, Spring Data JPA is highly advantageous. It allows developers to define database operations using Java interfaces and custom query methods, reducing the need for writing complex SQL queries. Spring Data JPA is particularly useful when used within the Spring framework, as it seamlessly integrates with other Spring components, making it easy to build data-driven web applications.

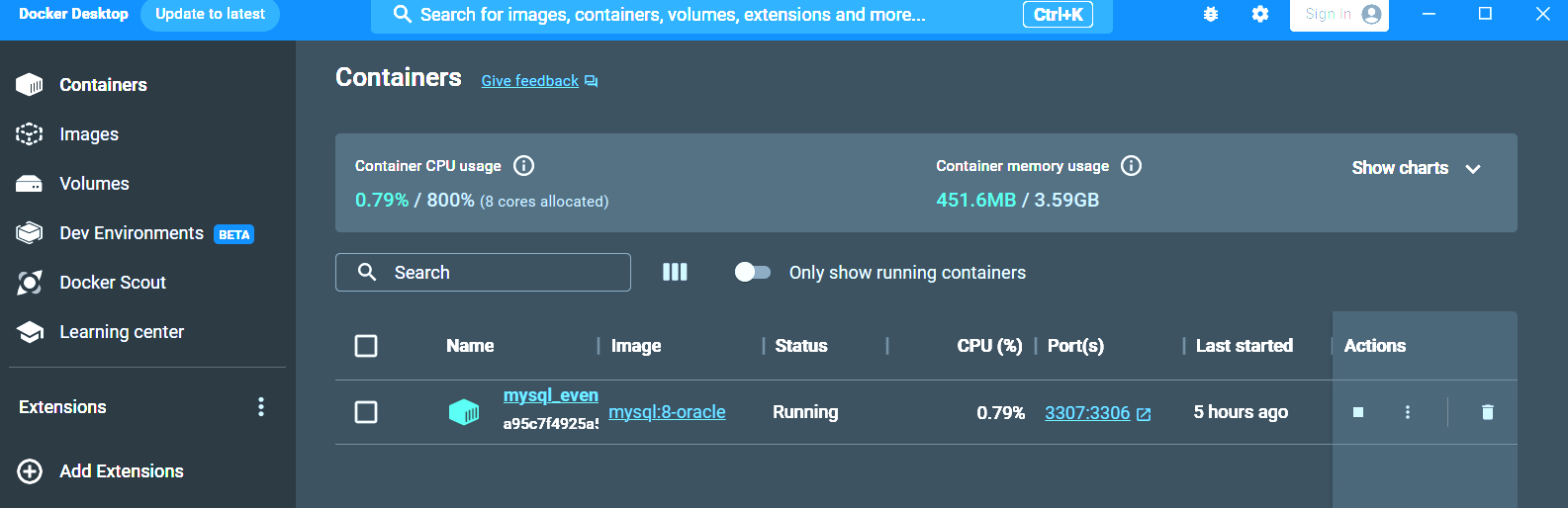
**Docker**

Docker leverages container technology, creating lightweight, self-contained environments called containers. These containers encapsulate an application and all its dependencies, including libraries and runtime components. One of the defining features of Docker is its ability to ensure that applications run consistently across different environments, from developers' laptops to test environments and production servers. This consistency is a response to the age-old challenge of "it works on my machine."

**Containers**

Containers are a form of virtualization that allows you to package an application, along with its libraries, dependencies, and runtime environment, into a single unit called a container. These containers run consistently across different platforms, ensuring that what you develop in your local environment behaves the same way in production. Containers are isolated from one another and share the host operating system's kernel, making them more efficient and lightweight compared to traditional virtual machines.

**Use Case: Dockerized MySQL for a Spring Boot Web Development Project**

In our software development project, we adopted Docker as a key component to streamline our development and deployment workflow. One of the pivotal use cases involved utilizing a Dockerized MySQL image and connecting it to our Spring Boot web development application. This approach offered several advantages, enhancing the development and testing phases while ensuring consistency and reproducibility in our project. Incorporating Dockerized MySQL into our Spring Boot web development project significantly improved our development workflow. It minimized configuration discrepancies, ensured application and database compatibility, facilitated version control, and provided efficient testing environments. By leveraging Docker in conjunction with Spring Boot, we achieved a more reliable, maintainable, and scalable web application development process. This use case stands as a testament to the practicality and advantages of containerization in modern software development.

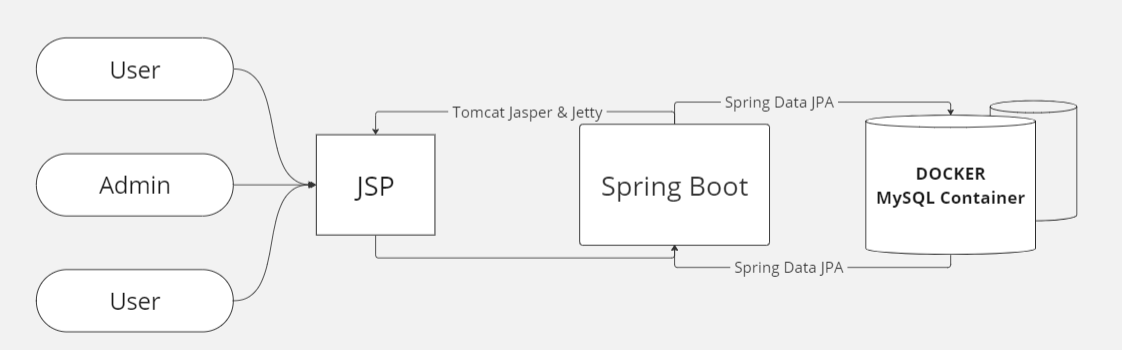
**METHODOLOGY and process**

The development of this web application project involved a comprehensive approach that integrated various components and technologies to create a seamless user experience. In this methodology, we will outline the key aspects of the development process, focusing on the use of Spring Boot for the backend, JSP and JSTL tags for the front-end, and Spring Data JPA for database management.

**Choice of Technology Stack**

The selection of technology stack is a critical decision that underpins the entire development process. In this project, we opted to use Spring Boot as the framework for building the web application due to its efficiency, robustness, and the availability of various libraries and tools. We also chose to employ JSP (JavaServer Pages) and JSTL (JavaServer Pages Standard Tag Library) for the front-end, while leveraging Tomcat as the application server. Additionally, we integrated Date Picker, a form library, and Bootstrap to enhance the user interface.

Figure Tech Stack

 **Front-End Development**

The front-end component of the web application is responsible for providing a user-friendly interface. We adopted a set of technologies and practices to achieve this goal:

**JSP and JSTL Tags**: JSP and JSTL tags were employed to create dynamic web pages. JSP allows us to embed Java code within HTML, making it easy to generate dynamic content. JSTL tags facilitate logic and data manipulation within the pages.

**Bootstrap**: Bootstrap, a popular front-end framework, was integrated to enhance the design and layout of the application. It provides responsive design elements, CSS classes, and components that ensure a visually appealing and consistent user experience across different devices.

**Date Picker**: Date Picker was incorporated to simplify the selection of dates, enhancing user interaction in scenarios where date input is required.

**Back-End Development**

The back-end component of the application, developed using Spring Boot and JAVA, is responsible for managing the core functionalities. We adhered to Object-Oriented Programming (OOP) principles to ensure code organization and maintainability:

**Controller and Request Mapping**: Controllers were used to handle incoming requests and map them to appropriate methods. Request mapping played a vital role in routing GET and POST requests to the respective controller functions.

**Spring Data JPA**: Spring Data JPA streamlined the interaction with the database. It provided high-level abstractions for data access, reducing the need for manual SQL queries and enhancing data integrity.

**Docker**: Docker containerized the MySQL database, ensuring consistency and portability across development and deployment environments.

**SQL**: SQL was employed to design and manage the database schema, offering a structured approach to data storage and retrieval.

**Process:**

**Admin Authentication:**

* The application starts with an authentication page for the admin.
* Admin enters their hardcoded credentials (username and password) and clicks the "Login" button.
* Upon successful login, the admin is directed to the admin dashboard.

**Admin Dashboard:**

The admin dashboard is dedicated to the admin and provides various functionalities:

* Create Events: The admin can create new events by providing event details and clicking the "Create Event" button.
* Create Tasks: Within each event, the admin can create tasks, assign them to users via a dropdown menu, and set task details.
* View Current Tasks: The admin can see a list of all tasks and their current status (e.g., pending or completed).
* Navigation Bar: A navigation bar at the top allows the admin to access different sections:
* "All Events": Shows a list of all events.
* "All Tasks": Displays all tasks.
* "Profile": Navigates to the admin's profile page.

**Create Event:**

* Admin clicks on the "Create Event" option in the dashboard.
* A form is presented where the admin enters event details, such as name, date, description, and any other relevant information.
* After entering the details, the admin clicks the "Create Event" button.
* The event is validated, and if the validation is successful, the event is stored in the MySQL database.

**Create Tasks:**

* Within an event, the admin selects the "Create Task" option.
* A form appears for creating tasks, including task name, description, and an option to assign the task to a user via a dropdown menu.
* After filling in the task details, the admin clicks the "Create Task" button.
* The task is validated, and if successful, it is associated with the event and assigned to the selected user. The task is stored in the database.

**User Registration and Login:**

* Users who are not admins go through a separate registration and login process.
* Users must sign up by providing their details, including a username, password, and any other required information.
* After successful registration, users can log in with their credentials.

**User Dashboard:**

Once logged in, users are directed to their dashboard.

The user dashboard displays:

* Assigned Tasks: A list of tasks assigned to the user within various events.
* Events List: A list of events the user is part of.
* Navigation Bar: Users can access different sections from the navigation bar, including marking tasks as complete.

**Task Management for Users:**

* Users can view their assigned tasks within events.
* Users can mark tasks as completed when they finish them.
* Completed tasks are updated in the database, and the status is changed accordingly.

**WALKTHROUGH**

**ANALYSIS**

**Duration of the project:** 3 Months

**Technology** **used**: Spring Boot, HTML & CSS, JSP, MySQL, Docker

**Third Year Level**

**Specification:** The web application can easily run on any browser.

**JAVA version:** Java 17

**Spring Boot:** 3.1.4

**Spring:** 6

**MySQL:** 8-oracle

**XML version**: "1.0"

**XML Encoding:** "utf-8"

**Docker:** version 4.24.2

**Future Scope for Application Enhancement:**

In order to continually improve the user experience and provide a high-quality service, we can explore several feature enhancements. While we may need to acquire new skill sets and adapt to emerging technologies, these additions will help us better meet user needs. Over time, we can learn and implement the following features:

* **Team Formation**:

Introduce a feature for users to create and join teams or groups within the application.

* **Notifications and Reminders:**

Enhance the notification system to provide real-time updates on task assignments, event updates, and team-related notifications. Implement reminder functionality to alert users of upcoming tasks and events, helping them stay organized and meet deadlines.

* **Analysis and Reporting:**

Incorporate analytics and reporting capabilities to provide insights into task and event progress.

* **Task and Event Scheduling:**

Develop a scheduling feature that allows users to set due dates and times for tasks and events. Implement a calendar view for users to manage their schedules and prioritize tasks.

**BIBLIOGRAPHY**

**YouTube:**

YouTube, a prominent video-sharing platform, has been an invaluable resource in our journey of learning and enhancing our skills in Android Development, Kotlin Programming, API integration, and various other aspects. The platform hosts a multitude of tutorials, lectures, and informative videos that have significantly enriched our knowledge base.

Access to these resources has empowered us to grasp new concepts and techniques vital for the successful realization of our project. YouTube's engaging and informative tutorials have played a pivotal role in deepening our understanding of the subject matter. A notable advantage of YouTube is its accessibility, providing a wealth of educational materials at no cost. This inclusivity ensures that students from diverse backgrounds can enhance their skills and knowledge.

**Udemy:**

Udemy, a renowned online learning platform, has offered us a treasure trove of courses that have greatly enriched our Android development journey. This platform allowed us to explore advanced topics beyond our standard curriculum, and we are indebted to the quality content it provides.

Udemy's role in our project's success has been substantial, equipping us with the essential skills and knowledge required to design, deploy, and operate our application. The invaluable learning resources offered by Udemy have been instrumental in facilitating our development journey.

**GitHub:**

GitHub, a leading platform for version control and collaboration, has provided an extensive repository of resources to enhance our programming skills, spanning Android Development, Kotlin, API integration, and various libraries. The platform grants access to a diverse array of open-source projects, code samples, and tutorials, serving as indispensable resources for learning and expanding our knowledge.

Our use of GitHub extended beyond learning; it enabled efficient collaboration and code sharing among team members, contributing significantly to our project's success. This platform streamlined our development process, granting seamless access to vital resources and boosting our team's productivity. The comprehensive resources available on GitHub have played a pivotal role in augmenting our programming proficiency and meaningfully contributing to our project's triumph.

**ChatGPT and Bard in Software Development:**

*ChatGPT* and *Bard* are two AI-driven solutions that have emerged as indispensable assets in the realm of software development. These tools complement each other, offering a potent combination for tackling errors, optimizing code, enhancing logic, and streamlining debugging in the development process.

**ACKNOWLEDGEMENT**

*The development of this web application using Spring Boot, Docker, MySQL, Java, and the overarching concept of workload management for events has been an inspiring and rewarding experience for our team. We extend our heartfelt appreciation to the following parties who have contributed to the successful realization of this project.*

*Our gratitude extends to our professors and mentors, who guided and encouraged us to explore innovative solutions beyond the scope of our curriculum. Their unwavering support and expertise have been paramount in shaping our project into a well-rounded and efficient tool for event workload management.*

*We also acknowledge the invaluable contribution of the broader programming and open-source community. The wealth of knowledge, documentation, and support provided by this community has been indispensable in resolving challenges and expanding our understanding of the technologies used in this project.*

*Furthermore, our appreciation goes to the creators of online tutorials, documentation, and resources that cover a wide array of topics related to Spring Boot, Docker, MySQL, and Java. These resources have served as beacons of knowledge, guiding us through the intricacies of web application development.*

*Lastly, we express our gratitude to our peers and colleagues who have been an endless source of encouragement, collaboration, and knowledge sharing. Their support has bolstered our team's spirit and fortified our ability to overcome hurdles during the development lifecycle.*

|  |  |  |
| --- | --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| **Salimmiya Saudagar** | **Sunny Yadav** | **Prathamesh Zingade** |
| TE-AIML, 62 | TE-AIML, 77 | TE-AIML, 78 |