**Developer's Guide: SchedEase - Class Timetable Web Application**

---

### Introduction

SchedEase is a web-based application built to help teachers, students, and administrators organize and manage class schedules. This developer's guide provides information on the architecture, technologies used, and instructions on how to set up, extend, and contribute to the project.

---

# Table of Contents



---

## 1. Project Structure

The project is organized as follows:

bash
Copy code

```
/SchedEase
  ├── /src
  │    ├── /main
  │    │    ├── /java
  │    │    │    └── /org.example
  │    │    │         ├── ApplicationView.java      (Main entry
point)
  │    │    │         ├── controllers               (Controllers for
managing routes)
  │    │    │         ├── models                    (Data models)
  │    │    │         ├── services                  (Business logic
and service layer)
  │    │    │         └── utils                     (Utility
classes)
  │    │    ├── /resources
  │    │    │    └── application.properties          (Configuration
file)
  │    │    └── /webapp
  │    │         ├── /assets                       (Static resources
like CSS, JS)
  │    │         └── /views                        (HTML views for
rendering)
  │    └── /test
  │         ├── /java
  │         └── /resources
  └── /pom.xml                                      (Maven project
configuration file)
```

## 2. Technologies Used

SchedEase is built using the following technologies:

- **Java 17+**: Used for the backend development.
- **Maven**: For dependency management and project building.
- **Spring Boot**: To simplify backend development, providing robust support for web applications.
- **Thymeleaf**: A Java templating engine used for rendering dynamic HTML views.
- **Bootstrap**: For responsive front-end layout and design.
- **H2 Database** (or an alternative relational database for persistence): Stores class schedule data.
- **JUnit**: Used for unit testing the application.

## 3. Setting Up the Development Environment

**Prerequisites:**

- **Java 17+**: Install the latest version of Java. Make sure to set `JAVA_HOME` correctly.
- **Maven**: Install Maven to build the project and manage dependencies.
- **Git**: Used for version control and cloning the repository.
- **IDE**: Use IntelliJ IDEA, Eclipse, or Visual Studio Code for development.

**Steps to Set Up:**

**Clone the Repository**
Use Git to clone the repository:
bash
Copy code

```
git clone https://github.com/yourusername/schedease.git
```

1.

**Install Dependencies**
Navigate to the project directory and run Maven to install the dependencies:
bash
Copy code

```
mvn install
```

2.
3. **Open the Project in an IDE**
   Open the project directory in your preferred IDE (e.g., IntelliJ IDEA).
4. **Run the Application**
   From the IDE, run the `ApplicationView.java` class to start the application. The server will start on `http://localhost:8080`.

---

## 4. Key Components of the Application

**Model Layer:**

The model layer represents the data structure for classes and schedules.

- **Class**: Represents the class object, with attributes like name, room, start time, end time, and days.
- **Schedule**: Contains a list of all classes scheduled for the week.

Example:

java
Copy code

```
public class Class {
    private String name;
    private String room;
    private LocalTime startTime;
    private LocalTime endTime;
    private List<DayOfWeek> days;
}
```

**Controller Layer:**

Controllers handle HTTP requests, process input from users, and update the model.

- **ClassController**: Manages class-related actions such as adding, removing, and listing classes.
- **ScheduleController**: Displays the timetable and handles operations like clearing the timetable.

Example:

java
Copy code

```
@Controller
public class ClassController {

    @Autowired
    private ClassService classService;

    @PostMapping("/addClass")
    public String addClass(Class newClass) {
        classService.addClass(newClass);
        return "redirect:/schedule";
    }
}
```

**View Layer:**

The view layer uses **Thymeleaf** to render the HTML pages and displays the data dynamically.

- **ScheduleView**: Displays the class timetable grid.
- **AddClassView**: A form for adding a new class to the schedule.

Example:

html
Copy code

```
<form action="/addClass" method="POST">
    <input type="text" name="name" placeholder="Class Name"
required>
    <input type="text" name="room" placeholder="Classroom" required>
    <input type="time" name="startTime" required>
    <input type="time" name="endTime" required>
    <input type="checkbox" name="days" value="MONDAY"> Monday
    <!-- More checkboxes for other days -->
    <button type="submit">Add Class</button>
</form>
```

---

## 5. Running the Application Locally

To run the application locally, follow these steps:

1. Open your terminal and navigate to the project directory.

Run the following command:
bash
Copy code

```
mvn spring-boot:run
```

2. 
3. Once the application starts, open your browser and go to `http://localhost:8080` to access the application.

---

## 6. Adding New Features

To add a new feature, follow these steps:

1. **Define New Model**: Create a new class or modify the existing ones in the `models` package.
2. **Create or Modify Controller**: Add new endpoints or actions in the `controllers` package to handle requests related to your feature.
3. **Create Views**: Modify or add new Thymeleaf templates in the `views` folder to render new functionality.
4. **Update Database Schema**: If your feature requires new data, modify the database schema (e.g., using migrations) or add new queries in the `services` layer.

---

## 7. Testing

SchedEase uses **JUnit** for unit testing. To add new tests:

1. Create test classes in the `src/test/java` directory.
2. Write unit tests to validate business logic and controller behavior.

Example:

java
Copy code

```
@SpringBootTest
public class ClassServiceTest {

    @Autowired
    private ClassService classService;

    @Test
    public void testAddClass() {
        Class newClass = new Class("Math", "Room 101",
LocalTime.of(9, 0), LocalTime.of(10, 0), List.of(DayOfWeek.MONDAY));
        classService.addClass(newClass);
        assertTrue(classService.getClasses().contains(newClass));
    }
}
```

---

## 8. Common Issues and Troubleshooting

- **Application Not Starting**: Ensure that Java 17+ and Maven are properly installed. Check the `application.properties` file for any misconfigurations.
- **Database Connection Errors**: Verify that the database is running, and check connection settings in the `application.properties` file.

- **Page Not Updating**: Ensure that the web page is re-rendering after changes. Refresh the browser or check for JavaScript errors in the console.

---

## 9. Contributing to the Project

We welcome contributions! To contribute:

1. **Fork the repository**.
2. **Clone your fork** and create a new branch for the feature or bug fix.
3. **Develop your changes** and write tests.
4. **Push your changes** and submit a pull request with a detailed description of the changes.

---

## Conclusion

This developer's guide provides the necessary steps to understand, set up, and contribute to the SchedEase Class Timetable Web Application. Follow these guidelines to extend the functionality, troubleshoot, and help improve the application. Happy coding!