Advanced Web Design Project

Development of a Full-Stack **Project Management** Application

Универзитет „Св. Кирил и Методиј“ во Скопје
ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО

Marko Nedelkovski 213280

Introduction

The Task and Project Management application is a robust, full-stack solution designed to help users organize and track their work efficiently. Whether the user is an individual managing personal tasks or part of a team working on a complex project, this application provides the tools to manage multiple projects, assign tasks, and visualize progress using charts. Built with scalability and ease of use in mind, the application offers a seamless experience through a well-structured backend and a dynamic frontend interface.

At its core, the application revolves around three main entities: **Users**, **Projects**, and **Tasks**. Each user can create multiple projects, and each project can have a list of associated tasks. Tasks, in turn, are categorized based on their current state, such as "TODO," "Ongoing," or "Completed," enabling users to track their workflow with clarity. This hierarchical structure ensures that the user's progress is both measurable and manageable.

The backend is powered by **Java Spring Boot**, a well-known framework that facilitates the development of RESTful web services. It is responsible for managing the application's core functionalities, including user authentication, project management, and task tracking. Spring Boot's modularity and scalability make it an ideal choice for building large-scale applications, ensuring that the business logic is clearly separated from the data layer and the presentation layer.

For data persistence, **PostgreSQL** serves as the relational database management system (RDBMS). PostgreSQL was chosen due to its reliability, scalability, and ability to handle complex queries. It manages the storage and retrieval of the application's entities, providing a solid foundation for data-driven features like task categorization, project ownership, and user-specific data handling.

On the frontend, **Angular** is the framework of choice, providing a powerful and flexible architecture that supports the dynamic user interface. Angular's component-based approach allows for clean separation of concerns, enabling the creation of reusable components such as project and task lists, login forms, and charts. Moreover, Angular's two-way data binding ensures that the UI remains synchronized with the backend, providing real-time updates as users interact with the system.

To enhance the look and feel of the application, **Tailwind CSS** was used for styling. Tailwind's utility-first approach allows for rapid development of responsive and aesthetically pleasing interfaces without the need for custom CSS. Every element, from buttons to navigation menus, is styled with minimal effort while maintaining a consistent and modern design language. Furthermore, **Angular Material** was integrated to add polished UI elements like buttons and navigation menus, enhancing the overall user experience.

A key feature of the application is its use of charts to visualize data. Leveraging **ng2-charts** (a wrapper for Chart.js), the frontend provides users with meaningful insights into their progress.

For example, the pie charts display the distribution of tasks by state, while the bar charts show the number of projects across different categories. These visualizations help users quickly grasp the overall status of their work, enabling them to make informed decisions about their next steps.

The application also includes comprehensive user authorization. Users can register for the system, log in, and log out. User-specific data, such as their projects and tasks, is loaded dynamically after login, ensuring that each user only has access to their own information.

In the sections below, we will explore the service layer and controller layer in more detail, providing insight into how business logic is implemented and how data flows between the frontend and backend. This modular, service-oriented architecture ensures that each layer of the application is cleanly separated and maintainable.

1. Project Overview

The Task and Project Management application allows users to register, create projects, assign tasks, and track their progress. The projects can be categorized, and tasks can be marked with different states such as "TODO", "Ongoing", and "Completed." Charts are integrated into the application to provide users with visual data insights about their projects and tasks.

Some key features include:

- **User Authentication**: Users can register, log in, and log out.
- **Project Management**: Users can create, edit, delete, and view projects.
- **Task Management**: Users can assign tasks to projects, track task progress, and visualize task states.
- **Charts**: Both pie charts and bar charts are used to visualize the number of tasks and projects by category or state.

2. Technologies Used

Backend:

- **Java Spring Boot**: A popular Java framework that simplifies the development of RESTful services and backends. It was used to build the service and controller layers, handle user authentication, and manage the project/task data.
- **PostgreSQL**: The database used for this project. PostgreSQL is an open-source relational database that ensures data integrity and supports complex queries.

Frontend:

- **Angular**: A TypeScript-based web framework used for building the frontend. Angular was chosen for its powerful data binding, component-based architecture, and integration with Material UI.
- **Tailwind CSS**: A utility-first CSS framework that allows rapid development of responsive and customizable designs. Tailwind was used to style components across the frontend.
- **Material UI (Angular Material)**: Material UI components were used for some UI elements like buttons and navigation, giving the app a clean, modern look.

3. Backend Design

The backend of the application is structured around a service-oriented architecture. It is composed of several key layers:

- **Entity Layer**: This layer contains the data models representing users, projects, and tasks.
- **Repository Layer**: This layer interfaces directly with the database, allowing CRUD operations.
- **Service Layer**: Contains the business logic for handling projects, tasks, and users.
- **Controller Layer**: Responsible for handling incoming HTTP requests and providing appropriate responses.

Data Models

The project uses three main entities:

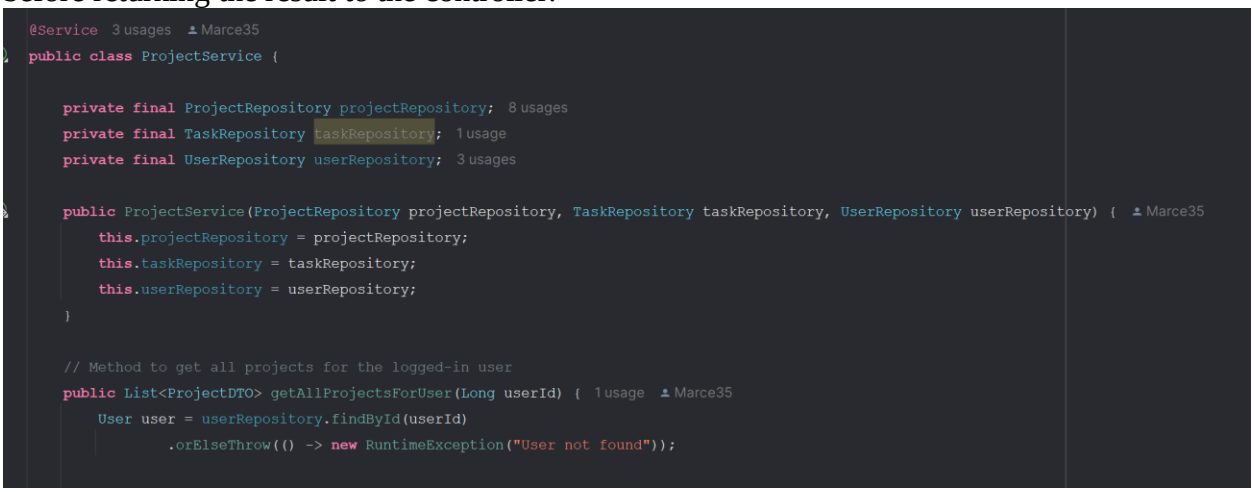
- **User**: Represents an application user. Each user can own multiple projects.
- **Project**: Represents a project created by the user. Each project has a name, description, category, and a list of tasks.
- **Task**: Represents a task within a project. Each task has a name, description, due date, and task state (TODO, Ongoing, Completed).

```java
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String username;
    private String password;
    private String email;

    // Getters and setters omitted for brevity
}
```

Service Layer

The service layer contains the business logic for the application. Each entity has a corresponding service class that handles operations like creating, updating, retrieving, and deleting entities. For instance, the `ProjectService` handles all project-related business logic, while the `TaskService` manages tasks.

The services interact with repositories to fetch or modify data, applying necessary business logic before returning the result to the controller.



Controller Layer

The controller layer handles incoming HTTP requests and sends responses to the frontend. Each entity (User, Project, Task) has its own controller class, which defines routes for CRUD operations. The controller layer uses DTOs (Data Transfer Objects) to pass data between the frontend and backend.

```java
@RestController  Marce35
@CrossOrigin(origins = "http://localhost:4200")
@RequestMapping("api/projects")
public class ProjectController {

    private final ProjectService projectService;  6 usages

    public ProjectController(ProjectService projectService) {  Marce35
        this.projectService = projectService;
    }

    // Endpoint to get all projects for the logged-in user
    @GetMapping("/user/{userId}")  Marce35
    public ResponseEntity<List<ProjectDTO>> getAllProjectsForUser(@PathVariable Long userId) {
        List<ProjectDTO> projects = projectService.getAllProjectsForUser(userId);
        return ResponseEntity.ok(projects);
    }

    // Endpoint to get a single project by ID along with tasks
    @GetMapping("/{projectId}")  Marce35
    public ResponseEntity<ProjectDTO> getProjectById(@PathVariable Long projectId) {
        ProjectDTO projectDTO = projectService.getProjectById(projectId);
        return ResponseEntity.ok(projectDTO);
    }
```

4. Frontend Design

The frontend was built using **Angular**. The application follows a component-based architecture, where different pieces of functionality (like project management, task management, and user authentication) are broken down into reusable components. Angular's data binding feature ensures that data retrieved from the backend is dynamically updated in the UI.

Angular Components

- **ProjectListComponent**: Displays all the projects a user has created. Each project is represented as a card.
- **TaskListComponent**: Displays all tasks related to a specific project.
- **ChartComponent**: Displays the data visualizations (charts) for tasks and projects.

- **LoginComponent & RegisterComponent**: Handles user authentication.

5. Key Functionalities

User Management

The user management module handles user registration, login, and logout. The login system tracks the current user's session and displays their information in the sidebar.

Project Management

The project management module allows users to create, edit, view, and delete projects. Each project is associated with a user, and users can see a list of all their projects in the **ProjectListComponent**.

Task Management

The task management module allows users to add tasks to a project, set due dates, and update the task status (TODO, Ongoing, Completed). The tasks are displayed in a list and are visualized through charts.

Charts and Data Visualization

Data visualization is handled using **ng2-charts** (a wrapper for Chart.js). Pie and bar charts are used to display the number of projects by category and the number of tasks by state (TODO, Ongoing, Completed).

Conclusion

In conclusion, the Task and Project Management application demonstrates the power and flexibility of modern web technologies when combined into a cohesive full-stack solution. The integration of **Spring Boot** on the backend, **Angular** on the frontend, **PostgreSQL** as the database, and **Tailwind CSS** for styling shows how a well-chosen tech stack can result in a highly functional, responsive, and scalable system.

The backend, designed using **Spring Boot**, is the backbone of the application. It handles the core business logic and data manipulation through its service and repository layers. By leveraging Spring Boot’s powerful features like dependency injection, data validation, and RESTful APIs, the application’s backend can easily handle CRUD operations for users, projects, and tasks. The clear separation of concerns ensures that future enhancements or bug fixes can be implemented with minimal disruption to other parts of the system.

The database layer, built using **PostgreSQL**, ensures that data is stored efficiently and securely. PostgreSQL's robust query capabilities, combined with the use of JPA (Java Persistence API) in Spring Boot, allow for complex data relationships (such as the one-to-many relationship between users and projects, and projects and tasks) to be managed seamlessly. The choice of PostgreSQL further ensures that the application can scale effectively as the user base and data volume grow.

On the frontend, **Angular** provides a dynamic, single-page application (SPA) experience, where users can interact with the system without experiencing page reloads. This modern approach improves user experience by making the interface more fluid and responsive. Angular's component-based architecture allows the application to be built in small, reusable pieces. For instance, components like the project list, task list, and user authentication are modular, making it easy to develop, test, and maintain them independently.

Tailwind CSS and **Angular Material** combine to offer a sleek, professional design that is both functional and aesthetically pleasing. Tailwind's utility classes ensure that the layout is fully responsive, adapting smoothly to different screen sizes and devices. This is especially important for applications like this one, where users may need to access the system on both desktop and mobile devices. Meanwhile, Angular Material's components, like buttons and cards, provide a polished, professional finish to the UI, contributing to the overall usability of the system.

One of the standout features of the application is the data visualization using **ng2-charts**. The ability to quickly see the status of tasks and projects through charts provides users with a deeper understanding of their workload and progress. The pie and bar charts, which show the distribution of tasks by state (TODO, Ongoing, Completed) and the number of projects across different categories, are an invaluable tool for project managers and team members alike. This visual representation allows for quick decision-making, as users can easily identify bottlenecks or areas that require more attention.

Additionally, the user management functionality ensures that users have a personalized experience. By securely managing user sessions, the system ensures that each user only has access to their own data. This is crucial for a multi-user application, as it guarantees data privacy and security while also offering a tailored experience.

Looking ahead, the architecture of this application provides a solid foundation for future growth. The modular design allows for the easy addition of new features, such as more advanced project reporting, collaborative task management, or integration with external tools like email notifications or calendars. The choice of technologies ensures that the application is not only scalable but also maintainable in the long term.

In summary, the Task and Project Management application is a comprehensive, well-designed system that solves real-world problems in task and project management. It leverages the strengths of modern web technologies to deliver a solution that is both powerful and user-friendly. By providing features like project and task tracking, user authentication, and data visualization, the application empowers users to manage their work effectively and efficiently. Whether used by individuals or teams, this application is a valuable tool for staying organized, tracking progress, and achieving goals.