**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**SECOND SEMESTER 2024-25**

DSECS ZG628T DISSERTATION (DSE)

**Mid Semester Project Report on**

CI/CD Pipeline with DevSecOps Integration for a Microservices-Based Application

**Submitted By**

Yash Hoshing

BITS ID: 2021MT70136

**Name of supervisor : Nomitha Angadi Name of Examiner: Praveenkumar Gopagoni**

**Designation:** Associate Principal – Software Engineering **Designation:** Professor

**INDEX:**

**1. Project Overview ......................................................**

**2. Project Goal and Objectives ...........................................**

**3. UI Overview ...........................................................**

**3.1 User Login Page ...................................................**

**3.2 Add Task Page .....................................................**

**4. Challenges Encountered ................................................**

**5. Project Timeline and Status ...........................................**

**5.1 Timeline Table ....................................................**

**5.2 Status Summary ....................................................**

**6. Planned CI/CD Integration .............................................**

**7. Testing and Security ..................................................**

**1.Project Overview**

This project is centered on the combined design, development, and deployment of an advanced Task Management Application and the creation of a robust Continuous Integration/Continuous Deployment (CI/CD) pipeline. Both components play equally critical roles in the automation, reliability, scalability, and productivity delivered through the final solution. The dual focus ensures that not only is a functional application created, but it is also built, tested, secured, and deployed through industry-standard pipelines.

**2.Project Goal and Objectives**

**Goal :**

Design, develop, and deploy a robust Task Management Application that streamlines organizing, tracking, and completing tasks for individuals and teams, ensuring productivity and collaboration.

**Objectives :**

* Implement secure user authentication and authorization systems.
* Enable creation, updating, deletion, and retrieval of tasks with features like status, progress tracking, priority, dependencies, and labels.
* Provide real-time dashboards and reporting features for monitoring progress and productivity.
* Lay the groundwork for integrating CI/CD pipelines to automate testing, building, and deployment in future stages.

**System Requirements**

**Hardware Requirements :**

* Processor: Quad-core 2.4GHz or greater.
* RAM: Minimum 8GB (16GB recommended for production).
* Storage: At least 20GB of available disk space.

**Software Requirements :**

* **Backend:** Python 3.9+, FastAPI, SQLAlchemy, PostgreSQL.
* **Frontend:** React.js
* **Containerization:** Docker (for future deployment).
* **Optional (Future plans):** monitoring tools (Grafana, Prometheus).

**Progress to Date :**

* **Requirements Analysis:** Defined use cases and main features around secure task operations and user roles.
* **System Design:** Designed the application architecture and database schemas to support advanced task features (status, dependencies, priorities, progress, labels).
* **Backend Implementation:** Developed API endpoints for:
  + Secure user registration/authentication.
  + CRUD operations for tasks, including support for dependencies, priorities, and labels.
  + Dashboard endpoints supplying real-time data.

**API Endpoints Used in the Project :**

* **User Service Endpoints:**These endpoints allow users to register a new account, authenticate (log in) and obtain a JWT token, and retrieve their own profile information.

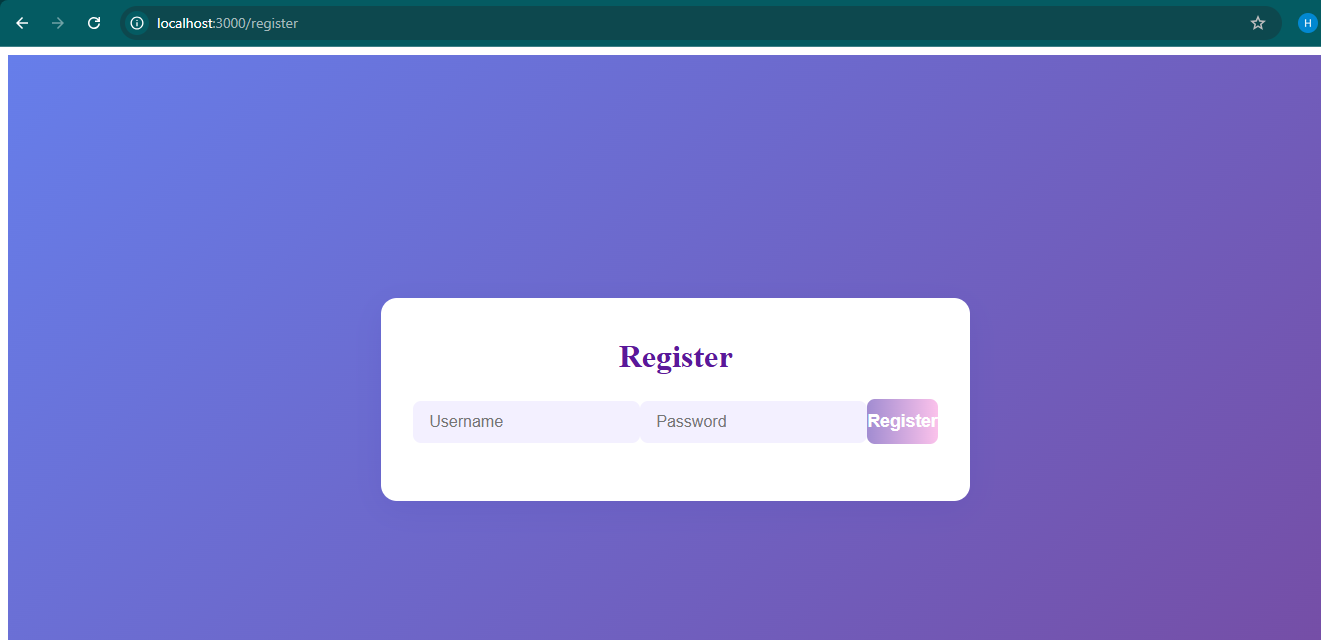
|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| /register | POST | Register a new user account with required credentials. |
| /login | POST | Authenticate user and return a JWT token for secure access. |
| /users/me | GET | Retrieve profile information of the currently authenticated user. |

* **Task Service Endpoints:**These endpoints let authenticated users create, read, update, and delete tasks—supporting features such as filtering by priority , reporting, and dashboard analytics. They provide all functionality required for users to manage their workflow and monitor their progress.

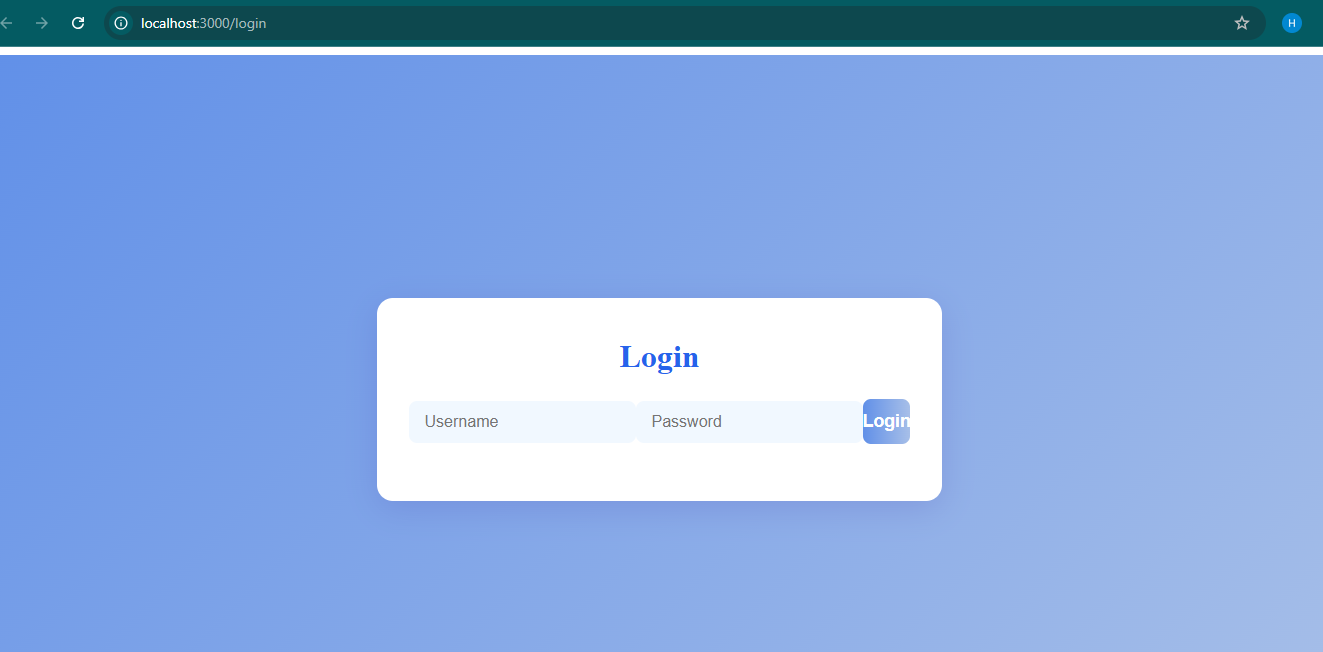
|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| /tasks | POST | Create a new task for the authenticated user. |
| /tasks | GET | Retrieve the list of tasks for the authenticated user. |
| /tasks/{task\_id} | GET | Get detailed information on a specific task owned by the authenticated user. |
| /tasks/{task\_id} | PUT | Update an existing task's data, including status, priority, dependencies, and labels. |
| /tasks/{task\_id} | DELETE | Delete a specific task. |
| /dashboard/summary | GET | Get real-time analytics including counts of completed, in-progress, blocked, and overdue tasks, plus averages. |

**3.UI Overview:**

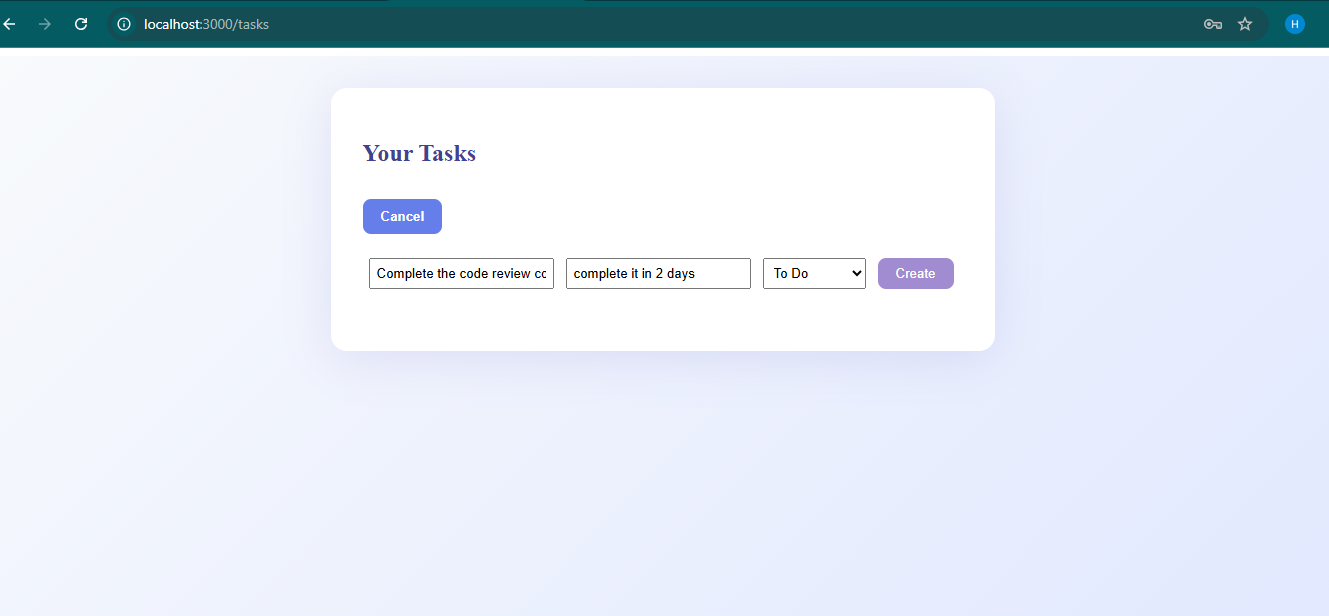
* **Register User**

****

* **User login Page**

****

* **Add Task Page**



**4.Challenges Encountered**

* Managing and validating complex task dependencies, ensuring data integrity.
* Designing extensible schemas to accommodate future collaborative and reporting features.
* Planning for future CI/CD integration while maintaining clean separation of concerns.

**5.Project Timeline and Status**

|  |  |  |  |
| --- | --- | --- | --- |
| **Week(s)** | **Task/Activity** | **Deliverables** | **Status** |
| 1-2 | Requirement analysis, literature review, and technology selection | Project plan, reviewed literature | Completed |
| 3-4 | Design application architecture and microservices | Architecture diagrams, service specs | Completed |
| 5-6 | Develop individual microservices | Source code for microservices | Completed |
| 7 | Containerize microservices with Docker | Dockerfiles, container images | In Progress |
| 8 | Set up CI/CD pipeline for build and unit testing | Pipeline scripts, test reports | In Progress |
| 9 | Integrate static code analysis and dependency scanning | Security scan reports | Pending |
| 10 | Implement integration and dynamic security testing | Integration test and DAST reports | Pending |
| 11 | Configure and deploy to Kubernetes cluster | Deployment YAMLs, running services | Pending |
| 12 | Set up monitoring and logging | Monitoring dashboards, logs | Pending |
| 13 | Conduct system-level testing and performance evaluation | Test results, performance metrics | Pending |
| 14 | Documentation of architecture, pipeline, and security analysis | Draft documentation | Pending |
| 15 | Final review, addressing feedback from supervisor/examiner | Revised documentation, final tweaks | Pending |
| 16 | Submission of final report and project demonstration | Final report, presentation | Pending |

**Future Plans and Immediate Next Steps**

* **Frontend Development:** Build a user-friendly interface for interacting with the backend APIs, supporting dashboards .
* **Documentation Expansion:** Complete end-user and deployment manuals.

**6.Planned CI/CD Integration**

* **CI/CD Pipeline Implementation:**
  + Automate code building, testing, and packaging using tools such as Jenkins.
  + Integrate static (SonarQube) and dynamic (OWASP ZAP) security scanning into the pipeline.
  + Enable automated Docker image creation and publishing for backend services.
  + Develop and test deployment scripts for Kubernetes.
  + Configure continuous deployment with automatic rollback, monitoring, and reporting.

**7.Testing and Security:**

* + Expand unit and integration test coverage.
  + Harden the system against vulnerabilities through automated scans.
* **Deployment:**
  + Roll out production deployments using the new pipeline.
  + Establish centralized logging and monitoring.