PAK-AUSTRIA FACHHOCHSCHULE:
INSTITUTE OF APPLIED SCIENCES AND TECHNOLOGY

Team Members:

Safi Ullah ***(B22F0549SE031)***

Sharyar Naveed ***(B22F0782SE014)***

Muhammad Moin ***(B22F1629SE148)***

Course:

Software Construction and Development
(COMP – 370)

Program:

Software Engineering – 22 – RED

Instructor:

Dr. Nabeel Ahmed

Submitted Date:

29 – January – 2024

Title:

(SRS) Document

Software Requirement Specification (SRS)

Title:

GitHub Integration for Agile Team Release Management

Introduction

Agile development teams need an efficient and automated way to track software releases, manage tasks, and monitor CI/CD pipelines. This project is integrated with GitHub APIs to fetch release data, track tasks, and monitor CI/CD pipeline statuses through a centralized dashboard.

1. Functional Requirements

1.1. GitHub Repository Monitoring

FR-01 The system must monitor a specified GitHub repository for changes.

- **Input:** Repository URL, monitoring interval.
- **Process:** Poll GitHub API at scheduled intervals.
- **Output:** Detect new commits, branches, pull requests.

FR-02 The system must fetch the latest changes when a repository update is detected.

- **Input:** New commit detected in the GitHub repository.
- **Process:** Fetch updated code using GitHub API.
- **Output:** Store changes locally for further processing.

1.2. CI/CD Pipeline Integration

FR-03 The system must trigger a CI/CD pipeline when changes are detected.

- **Input:** Updated repository files.
- **Process:** Call GitHub Actions API to start a build process.
- **Output:** Start build and deployment pipeline.

FR-04 The system must track and log CI/CD pipeline progress.

- **Input:** Pipeline status (running, failed, completed).
- **Process:** Continuously fetch build logs from GitHub Actions API.
- **Output:** Store build status and logs into the database.

FR-05 The system must notify users when a build is completed or fails.

- **Input:** Build status (Success/Failure).
- **Process:** Send notifications using Slack API, email, and in-app alerts.
- **Output:** Notify users about deployment success or failure.

1.3. Task and Release Management

FR-06 The system must track software releases linked to GitHub tags.

- **Input:** New release tag detected in GitHub repository.
- **Process:** Store release name, description, and date.
- **Output:** Display release history in the dashboard.

FR-07 The system must allow users to assign tasks to each release.

- **Input:** Task name assigned team member, status.
- **Process:** Link tasks to GitHub issues or pull requests.
- **Output:** Task progress tracking in the dashboard.

FR-08 The system must support task status updates.

- **Input:** Task completion percentage, status change.
- **Process:** Update database and notify users.
- **Output:** Reflect real-time task updates on the dashboard.

1.4. User Authentication and Authorization

FR-09 The system must allow users to log in using GitHub OAuth authentication.

- **Input:** GitHub credentials.
- **Process:** Authenticate via GitHub OAuth API.
- **Output:** Grant user access.

FR-10 The system must restrict access based on user roles.

- **Input:** User role (Admin, Developer, Viewer).
- **Process:** Assign access permissions based on role.
- **Output:** Control system actions based on user role.

1.5. Dashboard and Notifications

FR-11 The system must display a real-time dashboard for monitoring releases and deployments.

- **Input:** Data from GitHub repository, CI/CD pipelines.
- **Process:** Fetch and display real-time information.
- **Output:** Interactive dashboard for users.

FR-12 The system must send notifications for important events.

- **Input:** Task completion, build failure, release update.
- **Process:** Trigger notifications using Slack, email, and in-app alerts.
- **Output:** Notify users in real-time.

---

2. Non-Functional Requirements

2.1. Performance

NFR-01 The system must handle multiple GitHub repositories simultaneously.

- **Justification:** Teams may have multiple projects using CI/CD.

NFR-02 The system must provide real-time updates with minimal latency.

- **Justification:** Users need immediate feedback on builds and releases.

2.2. Security

NFR-03 The system must ensure secure authentication using GitHub OAuth.

- **Justification:** Prevent unauthorized access.

NFR-04 The system must encrypt sensitive data in the database.

- **Justification:** Protect user credentials and API keys.

2.3. Usability

NFR-05 The system should have a user-friendly UI with minimal training required.

- **Justification:** Users should easily navigate dashboards and reports.

NFR-06 The system must support both desktop and mobile browsers.

- **Justification:** Allow teams to monitor releases on any device.

2.4. Scalability

NFR-07 The system must be scalable to support large development teams.

- **Justification:** Multiple teams should be able to use the system simultaneously.

NFR-08 The system may be deployable on cloud platforms.

- **Justification:** May Support deployment on AWS, Azure, or private cloud if needed.

---

3. Technology Stack

Component	**Technology**
Backend	Java (Spring Boot)
Frontend	React.js
Database	PostgreSQL / MongoDB
Authentication	GitHub OAuth
CI/CD Integration	GitHub Actions
Notifications	Slack API, SMTP, In-App Alerts

---

4. Expected Outcome

- **Automated deployment system** integrated with GitHub.
- **Centralized dashboard** for monitoring software releases.
- **Real-time tracking** of CI/CD pipeline and task status.
- **Secure authentication** and **role-based access** control.
- **Scalable architecture** supporting multiple teams.

---

5. Future Enhancements

- Support for **Jenkins, GitLab CI/CD** integration.
- Integration with **cloud platforms (AWS, Azure, GCP)** for deployments.
- Advanced analytics for **deployment success/failure rates**.

---

6. Note:

The requirements outlined in the above **Software Requirement Specification (SRS) document** are **not strictly locked**. They may evolve over time due to **changes in project scope, technological change, or unforeseen challenges** during development. Any modifications or enhancements will be documented and reviewed as part of the **continuous improvement process** to ensure the system meets our expectations and needs effectively.