**DevOps**

**Programme name-  MSc Cloud and Network Security and MSc Software**

**Engineering**

**Module title - SWE7303 DevOps**

**Assessment title - SWE7303**

**Student number – 2333995,2332568,2315205**

**Marking tutor – Aamir Abbas**

[**GIT**](https://github.com/dhaval543/DevopsBlog)

Contents

[Introduction 3](#_Toc163957566)

[Project Requirement 4](#_Toc163957567)

[Use case Diagram 6](#_Toc163957568)

[Activity Diagram 7](#_Toc163957569)

# Introduction

This is the discussion about, how we can improve a blog application using DevOps tool. The blog app is build using Java, Spring Boot, Maven, and other technologies like AWS, Docker, JWT, React, and PostgreSQL.

The main aim is to use DevOps to simplify the process of building, testing, and deploying the app. We'll use Docker to package the app and its dependencies, AWS to handle the servers where the app runs, and Maven to automate building tasks. To make things even smoother, we'll set up Continuous Integration and Continuous Deployment (CI/CD) pipelines. These pipelines will automatically test new changes and deploy them, so developers don't have to spend time doing it manually. using AWS services like EC2 for scaling computing power, RDS for managing databases easily, S3 for storing and getting blog content, and IAM for controlling who can access what.

# Project Requirement

Design an online platform, where people can share their thoughts and stories with the world. With social media and other web pages, this blog allows you to talk about things in more detail and connect with readers in an interesting way. Whether you might have hobby in sharing your innovative ideas about technology, cooking or a business tips in your specialized area by blogging to helps you reach a wider audience and build a community around your interests.

**Key Features**

The project required to develop a blog website where users can post article, and interact with blog posts. The website aims to provide a platform for users to share their ideas, experiences with a people. main features such as user registration and authentication, blog post creation and management, commenting, and search functionality. The website should be built in modern web technologies, like Java/Spring Boot for the backend, React for the frontend, and PostgreSQL for the database. the website will be hosted on a cloud platform such as AWS, ensuring scalability, reliability, and security. The project must utilize best practices of DevOps tool and web development, technique including building, testing, and deployments.

1. **Functional Requirements:**

* **User Management:**
* User registration.
* User login/logout functionality.
* User profile management.
* **Blog Management**:
* Create, edit, and delete blog posts.
* View published blog posts.
* Commenting on blog posts.
* **Search Functionality:**
* Allow users to search for blog posts based on keywords or categories.
* User Interaction:
* Ability to like/start rating blog posts.
* Can share blog posts on social media platforms.
* **Administration:**
* Admin dashboard to managing users, blog posts, comments, etc.
* Admin should have access for approving/rejecting user comments.
* **Security:**
* Implementation of authentication and authorization mechanisms.
* Protection against common security threats such as SQL injection, XSS, CSRF, etc.

1. **Non-Functional Requirements:**

* **Performance:**
* Fast loading times for web pages.
* Should handle traffic and content.
* **Usability:**
* User friendly interface.
* Responsive design for mobile devices.
* **Reliability:**
* Minimal downtime and high availability of the website.
* Regular backups of data to prevent data loss.
* **Security:**
* Encryption user passwords and sensitive user data.
* Protection against unauthorized access.

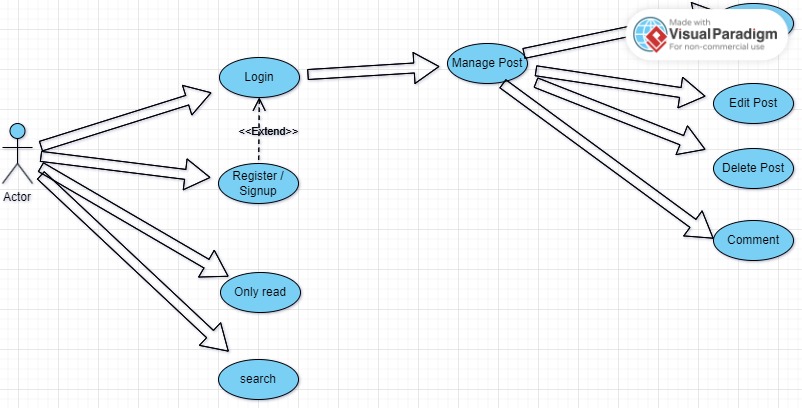
1. **Technology Stack:**

* **Backend**:
* Programming languages and frameworks (e.g., Java/Spring Boot).
* Database management system (e.g., PostgreSQL).
* Web API authentication tool (e.g., JWT for authentication).
* **Frontend:**
* JavaScript libraries/frameworks (e.g., React).
* HTML/CSS for styling.
* **Infrastructure:**
* Cloud platform (e.g., AWS) for hosting.
* Continuous Integration/Continuous Deployment (CI/CD) tools. (Jenkins)

1. **Testing**:

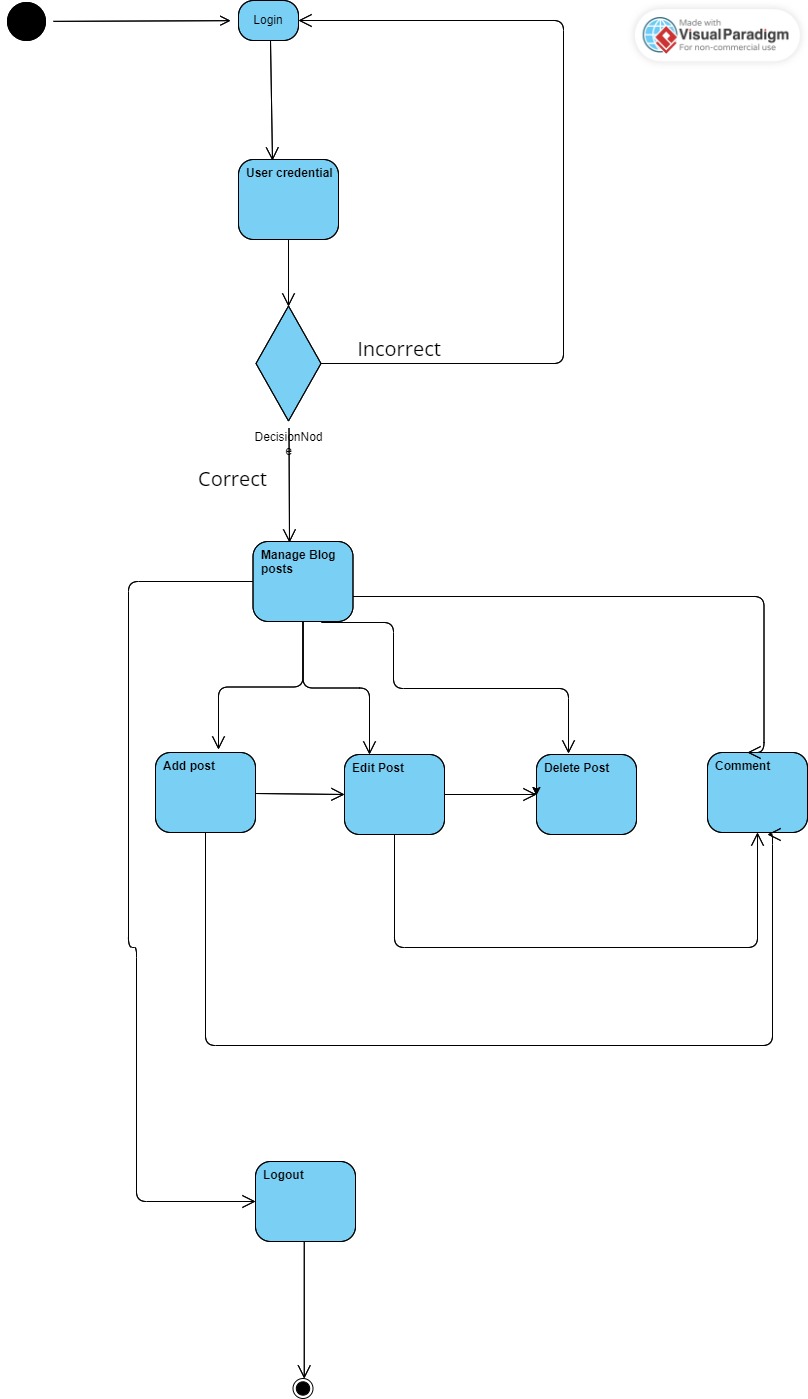
* Unit testing of backend and frontend.
* Integration testing to make sure different parts of the system working fine.
* End-to-end testing to validate the entire user flow.

# Use case Diagram



Use case diagram 1.1

# Activity Diagram



Activity Diagram 1.1

# TEchnology overview

## Java

Java is a versatile and widely-used programming language known for its portability, robustness, and scalability. Its portability is achieved through the Java Virtual Machine (JVM), which is allows Java programs to run on any platform without any modification. According to Smith (2020), the JVM provides a platform-independent execution environment, ensuring that Java applications behave consistently across different operating systems. This portability enables developers to write code once and deploy it anywhere, reducing the need for platform-specific adaptations and enhancing software accessibility.

The robustness of Java achieved through strong exception handling mechanism and automatic memory management through garbage collection. As noted by Brown (2019), Java's robust exception handling ensures stable performance by effectively managing errors and preventing unexpected program termination. Additionally, automatic memory management frees developers from manual memory allocation and deallocation tasks, reducing the risk of memory leaks and enhancing software reliability.

Java's scalability is supported by its built-in support for multithreading and networking. According to Williams (2021), Java's multithreading capabilities allow developers to create concurrent, responsive applications capable of handling multiple tasks simultaneously. Furthermore, its networking features enable the development of distributed, scalable applications capable of communicating over the internet.

Examples of real-world Java applications include enterprise systems like banking software (e.g., online banking platforms), e-commerce websites (e.g., Amazon), Android mobile apps, large-scale web applications (e.g., LinkedIn), and scientific computing tools.

## SpringBoot

**Simplicity:** Spring Boot simplicity archived from its convention over configuration approach, which minimizes the need for explicit configuration and boilerplate code. According to Robinson (2018), Spring Boot automates many aspects of application setup and configuration, allowing developers to focus more on application logic and less on infrastructure concerns. This simplicity feature improve the development process and enhances developer productivity.

**robustness:** The robustness of Spring Boot is includes auto-configuration, embedded servers, dependency injection, and robust testing support. As highlighted by Miller and White (2020), Spring Boot's auto-configuration feature automatically configures the application based on its dependencies, reducing manual configuration errors and ensuring consistent behavior across deployments. On top of that, its embedded servers, such as Tomcat or Jetty, simplify deployment by packaging the application as a self-contained executable JAR file, eliminating the need for external server setup.

**Maintainability**: Spring Boot's maintainability is supported by its modular architecture. According to Adams (2019), Spring Boot promotes best practices in software design and architecture, making it easier for developers to understand, maintain, and extend their applications over time. Furthermore, its active community and extensive documentation provide developers with the resources they need to troubleshoot issues, implement new features, and keep their applications up to date.

**Security**: Spring Boot provides comprehensive security features out of the box, including authentication, authorization, and encryption (Robinson, 2018). It integrates seamlessly with Spring Security to handle common security concerns, such as user authentication with various authentication providers (e.g., LDAP, OAuth), role-based access control, and protection against common security vulnerabilities like cross-site request forgery (CSRF) and cross-site scripting (XSS) attacks (Adams, 2019).

**DevOps Integration:** Spring Boot facilitates DevOps practices by promoting automation, collaboration, and continuous delivery (Robinson, 2018). It integrates seamlessly with popular DevOps tools and practices, including containerization (e.g., Docker), orchestration (e.g., Kubernetes), continuous integration/continuous delivery (CI/CD) pipelines (e.g., Jenkins, GitLab CI/CD), and infrastructure as code (IaC) tools (e.g., Terraform), enabling streamlined development, testing, deployment, and management processes (Adams, 2019).

**Cloud-Native Support:** Spring Boot embraces cloud-native principles and provides features to build and deploy applications in cloud environments effectively (Miller & White, 2020). It offers built-in support for cloud-native patterns and services like service discovery, centralized configuration management, distributed tracing, circuit breakers, and fault tolerance, making it well-suited for developing microservices architectures and cloud-native applications (Robinson, 2018).

## Javascript

JavaScript is a popular programming language that can be used for both client-side and server-side development (Flanagan, 2020). It is widely used in web development for form validation, DOM manipulation, and asynchronous communication with servers (Simpson, 2018). JavaScript's dynamic typing and prototypal inheritance make it flexible and suitable for a wide range of applications (Brown, 2019).

**Event-Driven Programming:** JavaScript support an event-driven programming model, allowing developers to define event handlers that respond to user actions (McFarland, 2020). This model enables the creation of interactive and responsive web pages, enhancing the user experience (Duckett, 2014). By attaching event listeners to HTML elements, developers can execute JavaScript code in response to user interactions like clicks, mouse movements, and keyboard inputs (Flanagan, 2020).

**Asynchronous Programming:** JavaScript supports asynchronous programming, allowing tasks to be executed non-blockingly (Simpson, 2018). This is particularly useful for tasks that involve fetching data from external sources, such as APIs or databases. By using asynchronous functions like Promises or await, developers can write code that waits for asynchronous operations to complete without blocking the execution of other tasks (McFarland, 2020)

## REact

React is a JavaScript library for building user interfaces based on a component-based architecture (Banks, 2020). Developers can create reusable components that encapsulate UI elements and their behavior, promoting code reusability and maintainability (Mehta, 2019). By composing complex UIs from smaller, self-contained components, developers can efficiently manage application state and UI logic (Hoy, 2018).

**Virtual DOM:** React utilizes a virtual DOM for efficient rendering of UI components (Banks, 2020). The virtual DOM is a lightweight representation of the actual DOM, allowing React to perform updates in a more efficient manner (Mehta, 2019). When application state changes, React reconciles the virtual DOM with the actual DOM, updating only the components that have changed (Hoy, 2018). This approach results in better performance and improved user experience.

**Declarative Programming:** React use declarative programming approch, where developers describe the desired UI state and React takes care of updating the DOM to match that state (Banks, 2020). This simplifies the development process and makes it easier to reason about application behavior. Developers can focus on what the UI should look like in different states, rather than imperatively specifying how to update the UI in response to user actions (Hoy, 2018).

## MAVEN

Apache Maven is a powerful build tool it used for Java projects. It simplifies the process of managing dependencies, building, packaging, and deploying Java-based applications. Maven operates based on the concept of project object model (POM), which describes the structure and dependencies of a project in an XML format (Kosaraju, 2018).

One of the key features of Maven is it is automated dependency management. Maven allows developers to declare project dependent libraries in the POM file, specifying the required libraries and it versions. Maven then automatically downloads the necessary dependencies from central repositories and also can specify custom or private central repositories (Hummel, 2019).

Maven also provides a standardized project structure, making it easier for developers to navigate and understand project layouts. It offers a set of predefined lifecycle phases (e.g., compile, test, package, install, deploy), which can be executed individually using Maven commands (Rahman, 2019).

also, Maven supports plugin-based architecture, allowing developers to extend its functionality through custom plugins. There are various plugins available for Maven, covering a wide range of tasks such as code quality analysis, documentation generation, and integration with external tools and services (Kosaraju, 2018).