

Software Requirement Specification Document for CodeCritic

Sarah Wael, Youmna Hany, Karen George

Mohamed Yasser, Hady Amr

Supervised by: Supervised by: Dr. Salwa Osama, Eng. Nada Ayman

May 29, 2025

Table 1: Document version history

Version	Date	Reason for Change
1.0	5-Apr-2025	SRS First version's specifications are defined.
1.1	27-May-2025	SRS Second Modified version is made.

GitHub: <https://github.com/YoumnaHany/Advanced>

Contents

1	Introduction	4
1.1	Purpose of this document	4
1.2	Scope of this document	4
1.3	Business Context	4
2	Similar Systems	5
2.1	Academic	5
2.2	Business Applications	5
3	System Description	5
3.1	Problem Statement	5
3.2	System Overview	6
3.3	System Scope	6
3.4	System Context	6
3.5	Objectives	6
3.6	User Characteristics	6
4	Functional Requirements	8
4.1	System Functions	8
4.2	Detailed Functional Specification	8
5	Design Constraints	9
5.1	Standards Compliance	9
5.2	Hardware Limitations	9
5.3	Other Constraints	9
6	Non-functional Requirements	9
6.1	Performance Requirements	9
6.2	Scalability Requirements	9
6.3	Security Requirements	10
6.4	Usability Requirements	10
6.5	Availability Requirements	10
6.6	Maintainability Requirements	11
6.7	Compliance Requirements	11
7	Data Design	11
8	Preliminary Object-Oriented Domain Analysis	11
8.1	Initial Class Diagram	11
8.2	Core System Components	12
8.2.1	User Management System	12
8.2.2	Content Management	12
8.2.3	Administrative Controls	13
8.2.4	Developer Capabilities	13

8.2.5	Request Management System	13
8.2.6	Notification Framework	13
8.2.7	Communication Features	13
8.2.8	Status Tracking	13
9	Operational Scenarios	14
9.1	Scenario 1: Developer Signs Up	14
9.2	Scenario 2: Developer Logs In	14
9.3	Scenario 3: Viewing the Code Issue Feed	14
9.4	Scenario 4: Commenting on a Post	15
9.5	Scenario 5: Requesting a Code Modification	15
10	Project Plan	16
11	Appendices	17
11.1	Definitions, Acronyms, Abbreviations	17
11.2	Supportive Documents	18

Abstract

CodeCritic is a web-based code review system designed to streamline the peer review process in software development. It enables developers to collaboratively review code, leave feedback, and ensure quality before merging changes into the main codebase. The system features a secure login page, a dashboard displaying the status of code reviews, and a dedicated pull request page that lists items awaiting review. Developers can navigate to a review page that shows code diffs and allows inline commenting, approvals, or change requests. Once feedback has been addressed, the merge page helps confirm that a pull request is ready for integration by displaying unresolved issues and approvals. CodeCritic enhances software quality by promoting team collaboration, enforcing coding standards, and reducing the likelihood of bugs in production. It is a lightweight, focused tool suitable for teams seeking an efficient and user-friendly solution for managing code reviews.

1 Introduction

1.1 Purpose of this document

The purpose of this document is to outline the functional and non-functional requirements for the CodeCritic Code Review System. It serves as a comprehensive guide for developers, testers, and stakeholders, detailing the features and behaviors expected from the system. The document ensures all team members have a shared understanding of the system goals and scope.

1.2 Scope of this document

CodeCritic is a web-based application that streamlines the code review process for development teams. The system enables users to log in, view dashboards, manage pull requests, conduct reviews, and merge changes effectively. It ensures code quality through structured feedback, promotes collaboration, and minimizes bugs before deployment. This document focuses on describing the core functionalities and technical boundaries of the system.

1.3 Business Context

In collaborative software development, maintaining code quality while enabling rapid feature deployment is crucial. CodeCritic addresses this need by providing a centralized platform for managing code reviews. It allows team members to view open, closed, and pending reviews through an interactive dashboard. Developers can inspect pull requests, leave detailed comments on file changes, approve or request modifications, and finally merge the code if it meets all standards. This process encourages accountability, adherence to best practices, and reduces the likelihood of bugs reaching production. By automating and organizing code review activities, CodeCritic boosts team efficiency and code reliability.

2 Similar Systems

2.1 Academic

1. **Gerrit Code Review (Academic Research Use):**

What it is: Gerrit is a real-world open-source code review tool widely used by developers and also studied in academic research to understand review patterns, reviewer assignments, and quality outcomes.[2]

2. **Review Board:**

What it is: Review Board is a web-based code review tool developed at VMware and has been used in academic environments for experiments and studies.[3]

3. **GitHub:**

What it is: A similar system in the academic context is GitHub, a widely used platform for code version control and collaborative review. GitHub's code review system integrates features such as pull requests, file diffs, commenting, and merge approvals, all of which align closely with your "CodeCritic" project.[1]

2.2 Business Applications

1. **GitLab:**

Description: GitLab is another web-based DevOps platform that provides tools for source code management (SCM), CI/CD, and DevOps collaboration. GitLab allows teams to collaborate on code, perform code reviews, and handle merge requests, similar to GitHub.[4]

2. **Azure DevOps:**

Description: Azure DevOps is a suite of development tools offered by Microsoft that provides a comprehensive set of features for version control, project tracking, build automation, and code reviews. It supports Git-based repositories and offers pull requests as a core feature for managing code review workflows.[5]

3 System Description

3.1 Problem Statement

In collaborative software development, managing and maintaining high code quality across team members can be challenging. Without a streamlined review system, code may be merged without proper oversight, leading to bugs, inconsistent standards, and reduced maintainability. Existing tools like GitHub and Gerrit provide code review features, but they may not fully cater to academic or lightweight project needs. CodeCritic aims to solve this by offering a simple, focused code review platform that allows users to create, view, and manage pull requests with detailed code diffs and feedback tools. This ensures that all code changes are peer-reviewed before integration, maintaining software quality and promoting team collaboration.

3.2 System Overview

CodeCritic is a web-based code review tool designed to help development teams manage pull requests and maintain high code standards. The system includes user authentication, a dashboard for tracking review status, and dedicated pages for viewing pull requests, leaving feedback, and merging code. Reviewers can view file differences, comment on specific lines, and either approve or request changes. The merge interface confirms readiness by displaying unresolved issues and approvals. CodeCritic provides an intuitive interface and streamlines the review process for better collaboration.

3.3 System Scope

CodeCritic focuses on the core functionality of code review in a collaborative environment. The system allows users to log in, view a dashboard of reviews, inspect pull requests, give feedback, and merge approved changes. It does not handle full version control or repository hosting but integrates with existing repositories. The scope includes UI components for reviewing, commenting, and approval tracking. Future expansions may include integration with Git, notifications, and analytics.

3.4 System Context

CodeCritic operates as a standalone web application or integrated within an existing development environment. It interacts with code repositories to fetch pull request data and supports team collaboration through a shared review workflow. Users access the system via a login page, navigate through dashboards, and interact with various modules to review and approve code. It complements existing development tools by emphasizing review management.

3.5 Objectives

- Enable developers to log in and securely access the system.
- Display categorized pull requests and reviews (open, closed, pending).
- Provide a detailed pull request view with diff and comment functionality.
- Allow reviewers to leave feedback and approve or request changes.
- Confirm readiness for merge by showing approvals and unresolved issues.
- Improve code quality and collaboration across the development team.

3.6 User Characteristics

The users of CodeCritic are primarily software developers and team leads involved in code collaboration. They have basic knowledge of version control systems and understand code review practices. Users are expected to navigate web applications comfortably and interpret code differences. They seek a simple, focused tool to manage code reviews efficiently without the overhead of complex platforms. Reviewers and contributors will interact with different system pages based

on their roles—some reviewing pull requests, others creating them. The platform is designed to be user-friendly, responsive, and informative, catering to both novice and experienced developers in academic or professional settings.

4 Functional Requirements

4.1 System Functions

The system provides the following core functionalities to facilitate an effective code review process:

1. **User Authentication**

Users must log in to access the system. Role-based access control ensures developers, reviewers, and administrators have appropriate permissions.

2. **Code Submission for Review**

Developers can submit code changes for review, allowing other users to comment, provide feedback, and request changes.

3. **Commenting and Feedback**

Reviewers can leave comments on specific lines or sections of code, facilitating communication and improvement suggestions. The system supports threaded conversations for better collaboration.

4. **Approval and Merging of Code**

Once the review is complete and code changes are approved, the system allows for the merging of code into the main codebase. This ensures the reviewed code is integrated after approval.

4.2 Detailed Functional Specification

Below are the key functional features the system must support:

1. **User Authentication and Authorization**

Users must be able to register, log in, and be assigned specific roles such as developer, reviewer, or admin. The system must enforce role-based access control to ensure proper permissions.

2. **Code Submission and Review Initiation**

Developers should be able to submit code changes for review, enabling reviewers to view and leave feedback on the submitted code.

3. **Inline Commenting and Discussion**

Reviewers should be able to add comments to specific lines or sections of the code. The system must support threaded discussions for organized feedback and responses.

4. **Approval and Merge Functionality**

Once the code is reviewed and feedback has been addressed, the system must allow reviewers to approve the changes. After approval, the system must facilitate merging the code into the main codebase.

5 Design Constraints

5.1 Standards Compliance

The system must comply with industry-standard protocols for security, data privacy, and accessibility. It must adhere to best practices for user authentication, authorization, and secure communication (e.g., HTTPS, OAuth). Additionally, the system should follow accessibility standards, ensuring it is usable for people with disabilities (e.g., WCAG 2.1).[6]

5.2 Hardware Limitations

The system must be designed to run efficiently on standard hardware commonly used in development environments. It should not require specialized hardware or high-performance servers. The system should be optimized for scalability, capable of handling multiple simultaneous users without causing significant delays or performance degradation.

5.3 Other Constraints

The system must be compatible with commonly used web browsers (e.g., Chrome, Firefox, Safari) and support responsive design for use across different devices, including desktops, tablets, and mobile phones. It must also be easy to integrate with existing development tools and workflows, ensuring minimal disruption to the users' current practices.

6 Non-functional Requirements

6.1 Performance Requirements

The system must ensure efficient performance, even with a large number of simultaneous users and complex code reviews. The following performance standards must be met:

- Code diff generation, including large files, should be processed in under 5 seconds.
- Comments, approvals, and feedback interactions should have a response time of under 2 seconds.
- The system must handle at least 500 concurrent users without noticeable performance degradation.[9]

6.2 Scalability Requirements

The system must be able to scale horizontally to accommodate increasing users, repositories, and code submissions. It should be able to:

- Handle a growing number of users and codebases without significant degradation in performance.

- Support the addition of new functionalities or modules in the future without significant re-work of the existing system.
- Manage increasing volumes of code review data and history.[9]

6.3 Security Requirements

The system must adhere to modern security practices and protocols to protect user data and ensure secure code submissions. Key security requirements include:[8]

- All sensitive data (e.g., user credentials) must be encrypted using AES-256 encryption.
- User authentication must use industry-standard protocols such as OAuth 2.0 or JWT.
- The system should employ role-based access control to enforce permissions and ensure that users can only access features relevant to their role.
- All communication between the client and server must be encrypted using SSL/TLS.

6.4 Usability Requirements

The system must be user-friendly and designed to ensure ease of use for both developers and reviewers. Key usability requirements include:

- The system should have an intuitive and clear user interface (UI), with minimal learning curve for users.
- The code review interface should be responsive, accessible, and compatible with all modern browsers.
- The system should support responsive design to ensure it works efficiently across desktops, tablets, and mobile devices.

6.5 Availability Requirements

The system must be available for use 24/7, with minimal downtime for maintenance. The availability requirements include:

- The system must achieve 99.9% uptime.
- Scheduled downtime for maintenance should be communicated to users with at least 24 hours notice.
- The system should be fault-tolerant, with the ability to recover from system failures without losing data.

6.6 Maintainability Requirements

The system must be designed for easy maintenance and support. Key maintainability requirements include:

- The codebase should follow standard coding practices and be well-documented to facilitate updates and bug fixes.
- The system should support easy updates to both software and infrastructure without requiring significant downtime.
- It should be easy to diagnose and resolve issues through logging and error reporting.

6.7 Compliance Requirements

The system must comply with relevant legal, regulatory, and industry standards, including:

- Compliance with data protection laws, such as the General Data Protection Regulation (GDPR), ensuring proper handling and storage of personal data.
- Adherence to accessibility standards (e.g., WCAG 2.1) to ensure the system is usable by individuals with disabilities.[7]
- The system must comply with all applicable software licensing regulations.

7 Data Design

The platform uses a relational database to manage users, code issue posts, interactions, and modification requests. Key entities include Users, Posts, Comments, Reactions, and ModificationRequests. Each user has a unique ID and can create posts describing code issues. Posts contain fields such as title, description, code snippet, language, and timestamp. Other developers can interact with posts by commenting or reacting (e.g., helpful, nno changes, changes needed). Each comment and reaction is linked to both the user and the post. Developers can also request to modify the posted code, generating entries in the ModificationRequests table, which includes request status, linked user and post IDs, and optional proposed changes. The system maintains relationships such as one-to-many between users and posts, posts and comments, and posts and modification requests. Data consistency is ensured using primary and foreign keys, while indexing on post titles, usernames, and timestamps allows efficient retrieval. The design supports community-driven peer review and can scale to include features like pull request previews or review scoring in the future.

8 Preliminary Object-Oriented Domain Analysis

8.1 Initial Class Diagram

The UML class diagram illustrates a comprehensive code-sharing and collaboration platform with user management, content posting, and moderation features. The system is structured around several key components:

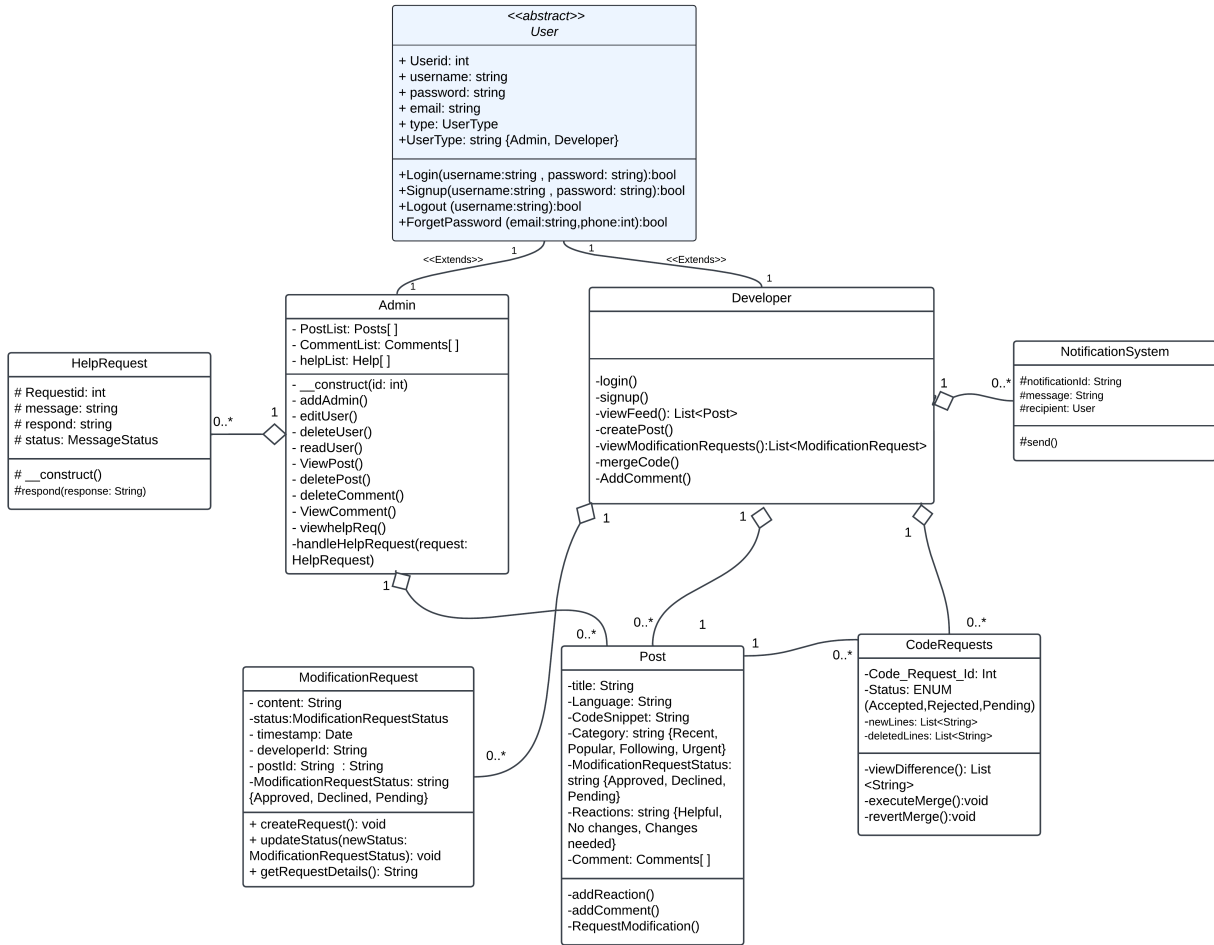


Figure 1: Initial Class Diagram of the CodeCritic System

8.2 Core System Components

8.2.1 User Management System

The platform implements a robust user hierarchy with an abstract `User` class that serves as the foundation. Users are categorized by `UserType` (`Admin` or `Developer`), each with specific capabilities. The `User` class contains identification attributes (`$id`, `$username`, `$password`, `$email`) and authentication methods (`+Login()`, `+Signup()`, `+Logout()`, `+ForgetPassword()`).

8.2.2 Content Management

Content is primarily managed through the `Post` class, which stores programming content including title, language, and code snippets. Posts can receive user reactions and comments, and can be subject to modification requests.

8.2.3 Administrative Controls

The Admin class extends the base User functionality with specialized methods for user management (-addAdmin(), -editUser(), -deleteUser(), -readUser()) and content moderation (-addPost(), -deletePost(), -deleteComment(), -viewComment()). Admins can also handle help requests from users.

8.2.4 Developer Capabilities

Developers have distinct functionality focused on content creation, viewing feeds of posts, and managing modification requests. They can view their own posts as well as modification requests made against their content.

8.2.5 Request Management System

The system supports several types of requests:

- **HelpRequest:** Allows users to submit assistance inquiries
- **ModificationRequest:** Enables collaborative code improvement through a formal request process
- **PullRequests:** Manages code modification suggestions with accept/reject capabilities
- **MergeRequests:** Facilitates code integration with version control features including diff views

8.2.6 Notification Framework

A NotificationSystem keeps users informed about platform activities, with messages directed to specific recipients.

8.2.7 Communication Features

The platform supports user interaction through:

- **Comments:** Allowing discussion on posts
- **Reactions:** Providing quick feedback options (Helpful, No changes, Changes needed)

8.2.8 Status Tracking

Various enumeration types track the state of objects within the system:

- **ModificationRequestStatus** (Approved, Declined, Pending)
- **Category** (Recent, Popular, Following, Urgent)

The system demonstrates a clear separation of concerns with appropriate relationships between classes, including inheritance, composition, and association, supporting a collaborative development environment where users can share, discuss, and improve code.

9 Operational Scenarios

9.1 Scenario 1: Developer Signs Up

Actor: Developer

Precondition: User is not registered on the platform.

Main Flow:

1. Developer opens the signup page.
2. Developer enters username, First name, Last name, Date of birth, Phone number, Gender, email, and password.
3. Developer clicks the “Sign Up” button.
4. System validates the input data.
5. If valid, system stores the new user and redirects to the code feed.

Postcondition: Developer account is created and user is logged in.

9.2 Scenario 2: Developer Logs In

Actor: Developer

tecondition: Developer has an existing account.

Main Flow:

1. Developer navigates to the login page.
2. Developer enters their registered email and password.
3. Developer clicks the “Login” button.
4. System verifies the credentials.
5. Upon success, developer is redirected to the post feed.

Postcondition: Developer is logged into their account.

9.3 Scenario 3: Viewing the Code Issue Feed

Actor: Developer

Precondition: Developer is logged in.

Main Flow:

1. System loads the post feed after login.
2. Developer scrolls through posts submitted by others based on post category "Recent, Following, , Popular or Urgent" or code language "python, javascript,...etc".

3. Each post includes title, code snippet, language, and interactions.
4. Developer can click on a post to view full details.

Postcondition: Developer is able to browse and access code issues from the feed.

9.4 Scenario 4: Commenting on a Post

Actor: Developer

Precondition: Developer is logged in and viewing a post.

Main Flow:

1. Developer clicks the “Comment” button under a post.
2. Developer types their comment in the input field.
3. Developer clicks “Submit” arrow.
4. System stores the comment and updates the post view.

Postcondition: The comment is displayed under the post.

9.5 Scenario 5: Requesting a Code Modification

Actor: Developer

Precondition: Developer is logged in and viewing a post.

Main Flow:

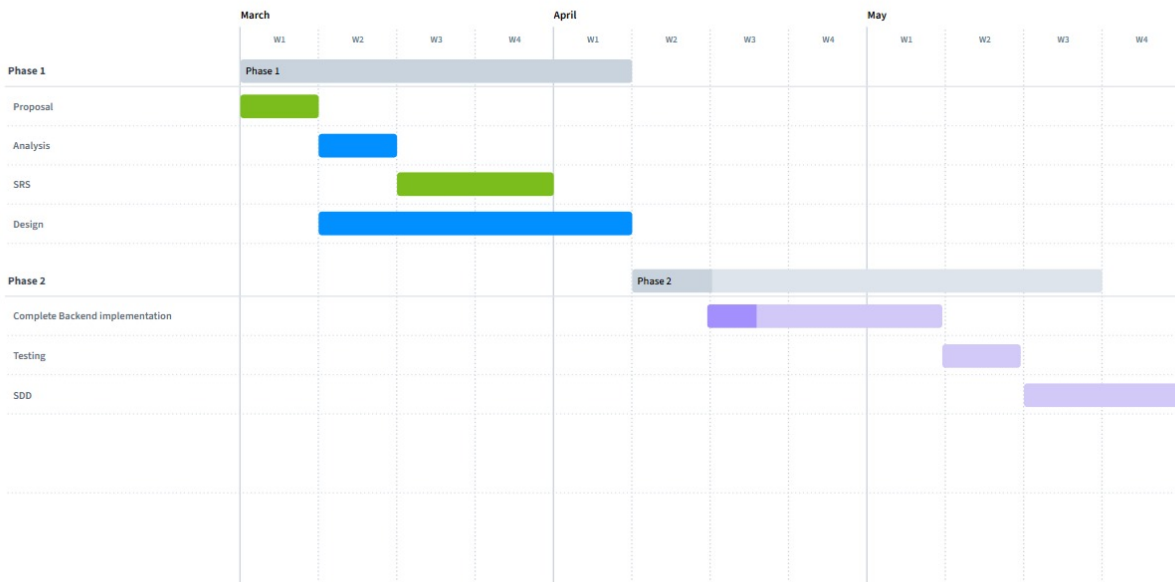
1. Developer clicks “Request Modification” under a code post.
2. Developer enters a description or attaches a file with changes.
3. Developer submits the request.
4. System saves the modification request with a “Pending” status.

Postcondition: Request is stored and viewable by the post owner and system.

10 Project Plan

Detailed plan from Proposal to SDD.

CodeCritic Timeline



11 Appendices

11.1 Definitions, Acronyms, Abbreviations

Term	Definition
SRS	Software Requirements Specification – A document that describes the functional and non-functional requirements of a software system.
CodeCritic	The name of the code review system being developed; it tracks and manages pull requests.
Pull Request (PR)	A feature in version control systems where developers submit changes for review before merging into the main code-base.
Code Review	The process of systematically examining source code to find and fix mistakes before merging it.
Diff	A representation of changes between two versions of a file, showing what was added or removed.
Merge	The process of combining code changes from different branches into a single branch.
Threaded Discussion	A conversation structure where replies are nested under the original comment for organized discussions.
Role-based Access Control (RBAC)	A method of restricting system access based on the roles of individual users.
OAuth 2.0 / JWT	Industry standards for secure user authentication and authorization.
SSL/TLS	Protocols used to secure data transmitted over a network.
WCAG 2.1	Web Content Accessibility Guidelines – rules to ensure web content is accessible to all users, including those with disabilities.
AES-256 Encryption	A secure encryption standard used to protect sensitive data.
GDPR	General Data Protection Regulation – a European law for data protection and privacy.
CI/CD	Continuous Integration and Continuous Deployment – practices for automating software building, testing, and deployment.
Repository	A storage location for software code, usually using version control systems like Git.
Frontend / Backend	Frontend refers to the user interface of the app; backend refers to server-side logic and data processing.
Scalability	The ability of a system to handle increased workload without impacting performance.
Threaded Comments	Comments organized so replies are clearly linked to the original message.
Fault-tolerant	A system's ability to keep working even when part of it fails.

11.2 Supportive Documents

Note that you should use a minimum of ten references (80% Academic)

- [1] <https://docs.github.com/en/get-started/quickstart/github-flow>
- [2] <https://www.gerritcodereview.com/>
- [3] <https://www.reviewboard.org/>
- [4] https://docs.gitlab.com/ee/user/project/merge_requests/
- [5] <https://learn.microsoft.com/en-us/azure/devops/repos/git/pull-requests>
- [6] <https://owasp.org/www-project-top-ten/>
- [7] <https://www.w3.org/TR/WCAG21/>
- [8] <https://eur-lex.europa.eu/eli/reg/2016/679/oj>
- [9] <https://learn.microsoft.com/en-us/azure/architecture/best-practices/scalability>