**Project Plan**

***Emergency Web App***

*Netherlands Citizens*

|  |
| --- |
| **Date : 8 September 2023** |
| **Version : 1** |
| **State : Ongoing** |
| **Author : Jennifer Oentoro** |

#### Version history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Changes** | **State** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Distribution**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Receivers** |
|  |  |  |
|  |  |  |

Contents

[1. Project assignment 4](#_Toc42673512)

[1.1 Context 4](#_Toc42673513)

[1.2 Goal of the project 4](#_Toc42673514)

[1.3 Scope and preconditions 4](#_Toc42673515)

[1.4 Strategy 4](#_Toc42673516)

[1.5 Research questions 4](#_Toc42673517)

[1.6 End products 4](#_Toc42673518)

[2. Project Organisation 6](#_Toc42673519)

[2.1 Stakeholders and team members 6](#_Toc42673520)

[2.2 Communication 6](#_Toc42673521)

[3. Activities and time plan 7](#_Toc42673522)

[3.1 Phases of the project 7](#_Toc42673523)

[3.2 Time plan and milestones 7](#_Toc42673524)

[4. Testing strategy and configuration management 8](#_Toc42673525)

[4.1 Testing strategy 8](#_Toc42673526)

[4.2 Test environment and required resources 8](#_Toc42673527)

[4.3 Configuration management 8](#_Toc42673528)

[5. Finances and Risk 9](#_Toc42673529)

[5.1 Project budget 9](#_Toc42673530)

[5.2 Risk and mitigation 9](#_Toc42673531)

# Project assignment

## Context

In the year 2022, the Netherlands witnessed a notable increase in crime rates, with the Dutch police force recording over 797,700 incidents. This marked a substantial increase when compared to the previous year, raising concerns about public safety and the need for effective emergency response mechanisms. In light of this growing challenge, there has been an increasing demand for innovative solutions to aid law enforcement agencies and emergency responders in efficiently addressing and managing emergencies. Recognizing the critical role that technology can play in these efforts, the development of an emergency web application has emerged as a promising initiative. Such an application could serve as a vital tool to bridge the gap between citizens in distress and the appropriate authorities, facilitating faster response times, accurate location tracking, and seamless communication during critical situations. The aim is to provide a user-friendly platform that empowers individuals to swiftly report emergencies, while also assisting emergency services in swiftly mobilizing resources and coordinating their response efforts. By harnessing the power of digital technology, this emergency web app seeks to enhance public safety and contribute to the overall security and well-being of the Dutch population in an ever-evolving landscape of security challenges.

## Goal of the project

The goal of this project is to address rising crime rates in the Netherlands, notably in 2022, by developing an advanced emergency web application. This initiative is driven by the need to improve public safety and enhance emergency response mechanisms. The desired outcome is a user-friendly platform enabling citizens to report emergencies swiftly and accurately while facilitating seamless communication with authorities for faster response and precise location tracking. The project's advantages include enhanced public safety, efficient emergency response, bridging communication gaps, and contributing to overall security and well-being. It adds value to the company or organization by meeting the pressing need for improved emergency systems and leverages ICT to offer real-time communication, location services, multimedia reporting, data analytics, and instant alerts.

## Scope and preconditions

|  |  |
| --- | --- |
| **Inside scope:** | **Outside scope:** |
| 1. Project Plan |  |
| 1. Design Document Version 1 |  |
| 1. Design Document Version 2 |  |
| 1. Design Document Version 3 |  |
| 1. Final Design Document |  |
| 1. Security Reports |  |
| 1. UI Design |  |
| 1. Database Schema |  |
| 1. Source Code |  |
| 1. Continous Delivery Code |  |
| 1. Testing Documentation |  |

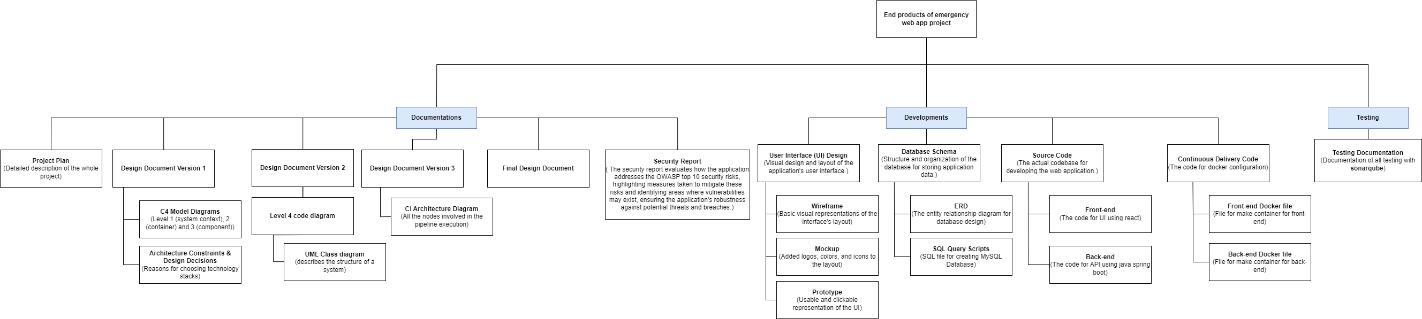
## Strategy

An Agile approach using the Scrum framework has been chosen for this project, as it is ideally suited to its needs. Scrum's high flexibility enables accommodation of changes in requirements, making it invaluable in scenarios where user needs, security concerns, and regulatory demands may evolve over time. This adaptability allows for seamless adjustment of the project's direction as needed. Additionally, this approach operates on regular sprint cycles, facilitating incremental delivery of the application within short timeframes. This enables us to gather feedback from stakeholders and customers promptly, which is a top priority, while maintaining transparency through open communication. Moreover, this framework excels in risk management, helping to identify and address potential issues effectively.

## Research questions and methodology

1. Research Questions
   1. What are the specific needs and preferences of users when it comes to reporting emergencies and seeking assistance through a web application?
   2. What are the technical requirements and constraints for ensuring the security and reliability of the emergency web application?
2. Methodology
   1. Technical Feasibility Study: Assess the technical feasibility of integrating real-time location tracking, multimedia reporting, and communication features, while ensuring data security and privacy compliance.
   2. Agile Development: Implement an Agile development methodology, specifically Scrum, to iteratively build and refine the application based on user feedback and evolving requirements.
   3. Data Analysis: Analyze user feedback, system performance data, and incident reports to make data-driven improvements and adjustments to the application.

## End products



# Project organisation

## Stakeholders and team members

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Abbreviation** | **Role and functions** | **Availability** |
| Jennifer Oentoro | JO | Project maker | Everyday |
| Maja | M | Technical Teacher | Wednesdays, Fridays |
| Bart | B | Technical Teacher | Mondays, Wednesdays |

## Communication

In-Person Meetings at Fontys University R10:

* Goals: To discuss project progress, clarify doubts, and align on tasks.
* Location: Fontys University R10.
* Timing and Frequency: Mondays, Wednesdays, and Fridays.

# Activities and time plan

## Phases of the project

Sprint 1

* + - Project Plan
    - Kanban Board
    - 3 Layers of RESTful API
    - CI/CD environment

Sprint 2

* + - Design Document
    - Cors configuration
    - Frontend Setup

Sprint 3

* + - Design Document
    - Backend and database setup
    - SonarQube

Sprint 4

* + - Login service and login interface
    - Role based a uthorization
    - SonarQube integration

Sprint 5

* + - Final document design
    - Security report
    - Websocket features
    - Minimum viable products features implemented
    - Continuous integration and SonarQube

Sprint 6

* + - Final UX feedback report
    - Final individual track product with minimum viable product features implemented
    - Continuous Integration and Sonarqube
    - Continuous Delivery8

## Time plan and milestones

|  |  |  |  |
| --- | --- | --- | --- |
| **Phasing** | **Effort** | **Start date** | **Finish date** |
| 1. Sprint 1 |  | Sep 4 | Sep 22 |
| 1. Sprint 2 |  | Sep 22 | Oct 13 |
| 1. Sprint 3 |  | Oct 13 | Nov 10 |
| 1. Sprint 4 |  | Nov 10 | Dec 1 |
| 1. Sprint 5 |  | Dec 1 | Dec 25 |
| 1. Sprint 6 |  | Dec 25 | Jan 19 |

# Testing strategy and configuration management

## Testing strategy

In order to ensure the reliability, functionality, and security of this project, I will implement a comprehensive testing strategy that encompasses multiple levels of testing. Each level serves a specific purpose and contributes to the overall quality of our software. The key elements of our testing strategy are as follows:

1. Testing Levels
   1. Unit Testing: These tests focus on individual units or components of this project’s code. Unit tests ensure that each part of the code functions as expected in isolation. Our goal is to achieve at least 80% code coverage for unit tests. I will automate unit testing using testing frameworks and libraries.
   2. Integration Testing: Integration tests verify that different components of this application work together seamlessly. I will automate integration tests to ensure efficient and consistent testing of these interactions.
   3. System Testing: System tests assess the overall functionality and behavior of the entire system. I aim to thoroughly test all user scenarios and edge cases. Some system tests will be automated, while others will be manually executed to simulate real user interactions.
   4. Acceptance Testing: These tests validate that the software meets the specified requirements and user expectations. User acceptance testing (UAT) will be performed manually by stakeholders, including product owners and end-users, to ensure alignment with their needs.
2. Quality Testing Tools

I will utilize SonarQube as a quality testing tool to perform code quality analysis. SonarQube will help identify code smells, bugs, and security vulnerabilities, allowing us to maintain high code quality standards. The specific goals with SonarQube include:

* 1. Achieving a code quality rating of at least "A" as per SonarQube's rating system.
  2. Identifying and resolving code issues and vulnerabilities promptly.
  3. Continuously monitoring code quality throughout the development process.

1. Automation

Automation is a crucial aspect of our testing strategy to ensure efficiency, repeatability, and consistency. I will automate unit testing, integration testing, and certain system tests. This automation will be integrated into the CI/CD pipeline to provide immediate feedback to developers.

1. Manual Testing

Manual testing will be conducted for certain system tests and user acceptance testing (UAT). Manual tests simulate real user interactions and are essential for validating the user experience and ensuring alignment with user requirements.

1. Testing Goals
   1. Achieve a minimum of 80% code coverage for unit tests.
   2. Identify and address critical issues detected during integration and system testing within 24 hours.

## Test environment and required resources

1. Test Environment Structure

I will adopt a DTAP (Development, Testing, Acceptance, Production) environment model to manage the progression of code changes through various stages of development and testing before deployment to the production environment.

* 1. Development (D) Environment: This environment is dedicated to the development team for coding, experimentation, and initial testing. It serves as a sandbox for developers.
  2. Testing (T) Environment: Code changes from the development environment are promoted to the testing environment for comprehensive testing. This includes unit testing, integration testing, and system testing.
  3. Acceptance (A) Environment: Once code changes pass all tests and quality checks in the testing environment, they are promoted to the acceptance environment. Here, stakeholders validate that the changes align with their requirements.
  4. Production (P) Environment: The production environment is the live environment where the application is accessed by end-users. Only thoroughly tested and accepted code changes are deployed here.

1. CI/CD Integration

I will implement a robust Continuous Integration/Continuous Deployment (CI/CD) pipeline to automate code quality checks and deployment processes. This pipeline will ensure that code changes progress seamlessly through the test environments. Key components of the CI/CD pipeline include:

* 1. Continuous Integration (CI): Developers commit code changes to the version control system. CI servers trigger automated builds and initial testing in the development environment.
  2. Continuous Testing (CT): Automated testing, including unit tests, integration tests, and code quality checks using SonarQube, takes place in the testing environment.
  3. Continuous Deployment (CD): If all tests are successful, code changes are promoted to the acceptance environment for further testing and user acceptance testing (UAT).
  4. Continuous Delivery (CD): Upon acceptance, code changes are automatically deployed to the production environment (Docker).

1. SonarQube Integration

SonarQube will be a critical component of this testing environment. It will provide comprehensive code quality analysis, identifying issues such as code smells, bugs, and security vulnerabilities. SonarQube's integration into our CI/CD pipeline ensures that code quality is continuously monitored and improved throughout the development process.

1. Resource Requirements
   1. Specific Tooling: I will employ CI/CD tools using GitLab CI/CD, containerization tools using Docker, and SonarQube for code quality analysis.

## Configuration management

1. Tooling: For version management, GitLab will be the central tool for source code version control, issue tracking, continuous integration (CI), and deployment. GitLab offers a comprehensive set of features for managing the software development lifecycle.
2. Branching Strategy:
   1. Master Branch: The master branch will represent the production-ready code. Code in this branch should always be in a deployable state.
   2. Development Branch: A development branch will be used for ongoing development work. Feature branches will be created from this branch for implementing new features or fixes.
   3. Feature Branches: Developers will create feature branches for working on specific features or bug fixes. These branches will be merged back into the development branch upon completion.
3. Promotion, Release, and Baseline Strategy
   1. Promotion: Promotion of code will follow a continuous integration/continuous delivery (CI/CD) pipeline in GitLab. Code changes will go through automated testing, and upon successful completion, they will be promoted to the development branch.
   2. Release: Major releases will be coordinated through release branches. These releases will be tagged in GitLab, and the code in the release branch will be deployed to production. Minor and patch releases will be managed similarly.
   3. Baseline Strategy: The baseline will be maintained in the master branch, representing the stable and production-ready version of the application. Any critical bug fixes or updates will be merged into this branch.
4. Change Requests and Problem Reports:
   1. Change Requests: Change requests will be managed through GitLab's issue tracking system. When a change request is received, it will be documented as an issue, including details of the requested change, its impact, and priority.
   2. Problem Reports: Problem reports, such as bug reports or issues identified in the application, will also be logged in GitLab's issue tracking system. They will include information about the problem, steps to reproduce, and severity.
   3. Handling Process: Both change requests and problem reports will follow a standardized process:
      1. Issues will be triaged and assigned to the project maker
      2. Project maker will create feature or bug-fix branches for issue resolution.
      3. Upon completion, code changes will be reviewed, tested, and merged.
      4. Change request status and problem resolution progress will be tracked within GitLab.

# Finances and risk

## Project budget

The budget for this project is not applicable.

## Risk and mitigation

|  |  |  |
| --- | --- | --- |
| **Risk** | **Prevention activities** | **Mitigation activities** |
| 1. Eye Strain and Discomfort | Use glasses with built-in blue light filters | Regular eye exams |
| 1. Loss of Motivation | Set clear and achievable goals for the project, maintain a well-organized and motivating workspace | Seek support and encouragement from peers or technical teachers |