Software Requirements specifications

Project ROME

v0.3-04.23.2023

Rasool Abbas, Onel Jimenez, Michael Mcfarlin, Ethan Wharton

Table of Contents

[Revision History 2](#_Toc129124801)

[1 Introduction 3](#_Toc129124802)

[1.1 Overview 3](#_Toc129124803)

[1.2 Goals and Objectives 3](#_Toc129124804)

[1.3 Scope 3](#_Toc129124805)

[1.4 Definitions 4](#_Toc129124806)

[1.5 Document Conventions 4](#_Toc129124807)

[1.6 Assumptions 4](#_Toc129124808)

[2 General Design Constraints 5](#_Toc129124809)

[2.1 Product Environment 5](#_Toc129124810)

[2.2 User Characteristics 5](#_Toc129124811)

[2.3 Mandated Constraints 5](#_Toc129124812)

[2.4 Potential System Evolution 5](#_Toc129124813)

[3 Nonfunctional Requirements 6](#_Toc129124814)

[3.1 Usability Requirements 6](#_Toc129124815)

[3.2 Operational Requirements 6](#_Toc129124816)

[3.3 Performance Requirements 6](#_Toc129124817)

[3.4 Security Requirements 6](#_Toc129124818)

[3.5 Safety Requirements 6](#_Toc129124819)

[3.6 Legal Requirements 6](#_Toc129124820)

[3.7 Other Quality Attributes 6](#_Toc129124821)

[3.8 Documentation and Training 6](#_Toc129124822)

[3.9 External Interface 6](#_Toc129124823)

[3.9.1 User Interface 6](#_Toc129124824)

[3.9.2 Software Interface 6](#_Toc129124825)

[4 System Features 7](#_Toc129124826)

[4.1 Feature: <name> 7](#_Toc129124827)

[4.1.1 Description 7](#_Toc129124828)

[4.1.2 Use Case 7](#_Toc129124829)

[4.1.3 Additional Requirements 7](#_Toc129124830)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Revising Author | Description |
| v0.3-04.23.2023 | 04/23/2023 | Onel Jimenez | -Begin Segment Two – General design constraints segment |
| v0.2-03.07.2023 | 03/07/2023 | Onel Jimenez | Begin Segment One – introduction and definition segment |
| v0.1-02.27.2023 | 02/27/2023 | Onel Jimenez | Initialize document – begin formatting and process. |

# 1 Introduction

## Overview

This document describes the requirements for Project ROME, the fundraising web application system. The purpose of this document is to adequately describe the requirements and standards upheld for the project, allowing the client to evaluate the project’s satisfactory of their needs. It also allows the client an easy to understand, summarized document of;

* how the Project works,
* what is required to host the Project,
* what maintenance of the Project may be required,
* how the Project can be improved upon.

This document does not contain the ongoing process of the project. Topics such as designation of tasks, development time, time budgeting, development phases, deliverables (and their definitions), testing data, testing methods, and other specifics of developing the Project are not included here.

Project ROME is a multi-layer web application designed to host fundraisers for individuals and organizations to easily gather funds to achieve a goal. The Project takes an emphasis in short-term motives, especially humanitarian efforts.

## Goals and Objectives

The goals of Project ROME are as follows;

1. Provide a simplistic way for users to build funds for a designated effort.
2. Allow ease of use for all users, including those requiring accessible features.
3. Ensuring secure practices, particularly with identity access management.
4. Ensuring anonymity when users do not want to publicize their donations.
5. An optimized balance of implementation ease and scalability for improvements.

## Scope

Project ROME is based on the primary requirements presented by Commerce Bank for a generic fundraising application. The PowerPoint that includes these requirements can be found [here](https://app.box.com/s/9r7lm3bhxendzaef6rql7pdbgf900nt8). Implementation features were discussed and ultimately voted upon by the initial development team; those features are highlighted in Segment 4 of this document.

The Project is not designed to sell products or design for for-profit intentions. The Project is also not a final product – it does not incorporate a way for users to pay with real money, only mimic a transaction. Should practical implementation occur, this document will be revised to incorporate and document the system.

The Project is also not designed to have complex moderation over fundraising. Deletion of a fundraiser and comments of a fundraiser are included in the Project, but no further moderation features are anticipated.

## Definitions

## Document Conventions

Note the following conventions that can be found in this document;

**PENDING** – The following change is being discussed by the development team, and may not be final.

**TBD** – There is not enough information at the revision version to be absolute. The content may change significantly as the Project continues development.

**NEEDS REVIEW** – During revision, the section has been marked for editing and adjustment. This may be due to incongruence with implementation or consideration of the content.

## Assumptions

It is assumed that;

* The system hosting the Project has Docker containerization capabilities.
* The network is properly configured;
  + The hosting system sits behind a network with a static IP address, with no more than a single network address translation. (IPv6 support **TBD**)
  + The system and gateways above it in the network can expose the HTTPS default port. (443)
  + The system does not unnecessarily expose API ports (API ports **TBD**)
* The product requires secure communication protocols (HTTP is not enabled).
* The system is hosted on a secure system with minimal access.
  + Although containerized, it is important to ensure the device is not unnecessarily exposed to external networks/systems.
* The system that hosts the API has the necessary pre-requisite packages to run the API node.

# General Design Constraints

## Product Environment

The Project has been designed to run modularly. In development, the project requires either PostgresSQL or Docker (to build PostgresSQL within a container) and npm. The business logic module of the project is built in C# using the .NET Core framework. In a production environment, each component would be run in separate containers for proper isolation.

## User Characteristics

The intended users of this system are as follows;

* The average user, who would register and donate to fundraisers or create fundraisers themselves.
  + The user would have the option to include comments with their donations.
  + The user would be able to upload images to fundraisers they make.
  + The implication of this user is that they are not knowledgeable of the application before using it – they would be able to explore the website with relative ease as the application is simplistic.
* The front-end administrator of the page can access the application and have “global” administrative privileges to all fundraisers
  + The front-end administrator is not implied to have a technical background – any user could be an administrator to the application.
  + This role is designed more for moderative purposes – if there were malicious fundraisers or comments, the front-end administrator could remove them.

## Mandated Constraints

The following constraints have been enforced for the Project by the Team (based upon the constraints of the client):

* Security is a forefront requirement of the Project. Storage of passwords will meet strict compliance of PCI DSS standards.
* Front-end is built in React.
* Business logic is built in C#.
* PostgresSQL is our intended database management system. Other database systems are unsupported.
* At least 10% of the code within the Project will be covered by test code.
* The Project shall have implemented;
  + A login page
  + A dashboard that presents all available fundraisers
  + A page that displays details of an individual fundraiser
  + A donation form that allows users to submit donations
  + A page for user-related settings, such as profile settings
* The Project will be implemented in a 3-layer system.
* The Project will be version controlled in GitHub.
* The Project will implement a session system, allowing for users to continuously stay logged in for ~18 hours.

## Potential System Evolution

The Project is designed in a modular fashion to make it easier to diagnose and improve upon the original design. Should changes be made, it is imperative to keep these changes from making the modules dependent on one another. The changes made to one module should not impact another module.

# Nonfunctional Requirements

## Usability Requirements

## Operational Requirements

## Performance Requirements

## Security Requirements

## Safety Requirements

## Legal Requirements

## Other Quality Attributes

## Documentation and Training

## External Interface

### 3.9.1 User Interface

### 3.9.2 Software Interface

# 4 System Features

## 4.1 Feature: <name>

### 4.1.1 Description

### 4.1.2 Use Case

### 4.1.3 Additional Requirements