GrizzTime

DCAA Time Tracking

Daniel Salar

Anthony Robin

Gurnirunjun Shergill

Melissa Palmer

Misael Hernandez

Table of Contents

[**Project Vision**](#_4e1dqqz5fmwu) **3**

[Background](#_ey43g84b33on) 3

[Socio-economic Impact, Business Objectives, and Gap Analysis](#_ey43g84b33on) 4

[Security and Ethical Concerns](#_ey43g84b33on) 4

[Glossary of Key Terms](#_ey43g84b33on) 4

[**Project Execution and Planning**](#_lfo850b5h8ce) **5**

[Team Information](#_4l6ynev6y4pj) 5

[Tools and Technology](#_4l6ynev6y4pj) 5

[Project Plan](#_4l6ynev6y4pj) 5

[Best Standards and Practices](#_4l6ynev6y4pj) 5

[**System Requirement Analysis**](#_ydliwp5gbkdm) **6**

[Functional Requirements](#_wfieu4q0s5k) 6

[Non-functional Requirements](#_wfieu4q0s5k) 6

[On-Screen Appearance of landing and other pages requirements](#_wfieu4q0s5k) 6

[Wireframe Designs](#_wfieu4q0s5k) 6

[**Functional Requirements Specification**](#_hrxd3fpms8q5) **6**

[Stakeholders](#_2jljob2ivuba) 6

[Actors and Goals](#_2jljob2ivuba) 6

[User stories, Scenarios and Use Cases](#_2jljob2ivuba) 7

[System Sequence/Activity Diagrams](#_2jljob2ivuba) 8

[**User Interface Specification**](#_kzf1sa4ig98j) **8**

[Preliminary Design](#_8rdvimnrkai3) 8

[User Effort Estimation](#_8rdvimnrkai3) 8

[**Static Design**](#_6trgyuzihmgw) **8**

[Class Model](#_t4xhuz22vpmh) 8

[System Operation Contracts](#_t4xhuz22vpmh) 8

[Mathematical Model](#_t4xhuz22vpmh) 9

[Entity Relation](#_t4xhuz22vpmh) 9

[**Dynamic Design**](#_4k7792mhlqbt) **9**

[Sequence Diagrams](#_rvcv8rc5tqwf) 9

[Interface Specification](#_rvcv8rc5tqwf) 9

[State Diagrams](#_rvcv8rc5tqwf) 10

[**System Architecture and System Design**](#_f288qo40bvoy) **10**

[Subsystems/Component/Design Pattern Identification](#_i1v6tdbblyoa) 10

[Mapping Subsystems to Hardware (Deployment Diagram)](#_i1v6tdbblyoa) 10

[Persistent Data Storage](#_i1v6tdbblyoa) 10

[Network Protocol](#_i1v6tdbblyoa) 10

[Global Control Flow](#_i1v6tdbblyoa) 10

[Hardware Requirement](#_i1v6tdbblyoa) 10

[**Algorithms and Data Structures**](#_dd7utfuwbsrx) **10**

[Algorithms](#_bk0x9bpbtqoz) 10

[Data Structures](#_bk0x9bpbtqoz) 10

[**User Interface Design and Implementation**](#_9wds70wc26z9) **11**

[User Interface Design](#_uazj0walb2ra) 11

[User Interface Implementation](#_uazj0walb2ra) 11

[**Testing**](#_i6kdivqu2xwx) **11**

[Unit Test Architecture and Strategy/Framework](#_lfi0tx6jzg7f) 11

[Unit Test Definition, Test Data Selection](#_lfi0tx6jzg7f) 11

[System Test Specification](#_lfi0tx6jzg7f) 11

[Test Report per Sprint](#_lfi0tx6jzg7f) 11

[**Project Management**](#_g1uiwvvcsfo0) **11**

[Project Plan](#_lv8u2nmjeng7) 11

[Risk Management](#_lv8u2nmjeng7) 11

# 

# 

# Project Vision

## Background

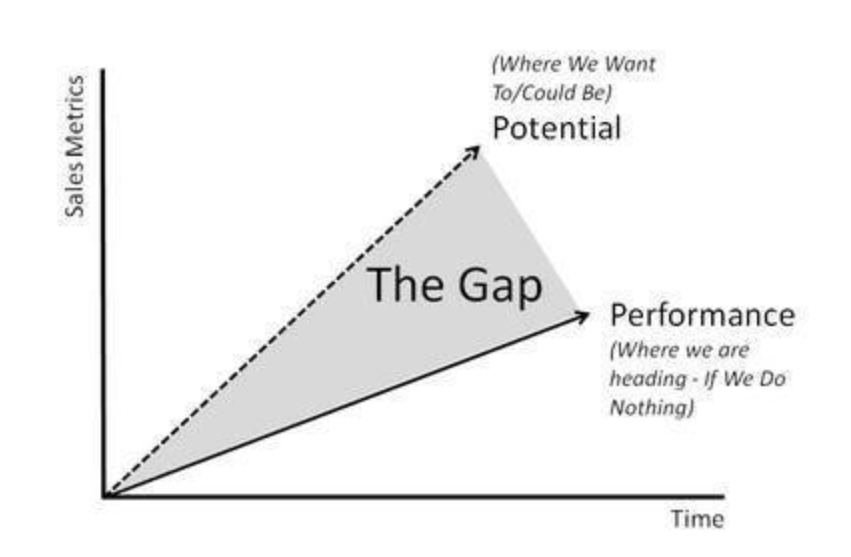
The goal of this project is to create a DCAA time tracking web application that will track the amount of time a contractor or project manager worked on a project. It will also calculate the amount of pay the worker will receive. It will be able to track expenses for a project being worked on and it will be able to generate an invoice for a project.

## Socio-economic Impact, Business Objectives, and Gap Analysis

This application will have an impact on how contractors and project managers calculate how much time they worked on a project according to DCAA standards.

As a business, this application will allow contractors to work on many projects, project managers will be able to manage many projects, a contractor will fill out many timesheets, a project manager will approve many timesheets, etc. The goal for a business is to create an application that will display the workers and which projects are being worked on. In addition, the business will be able to see how much is being paid to employees and what is being spent on certain projects.

The goal of our team is to accomplish these tasks within three months. So, if we are consistent in accomplishing everything for each sprint, then the goal is on track; however, in the case that the project falls behind, not everything may be accomplished.



## Security and Ethical Concerns

For a timesheet web-application, this must be secure. To avoid any hacking or any attack on the system, an anti-virus could be placed; however, since the database is placed on a cloud made by microsoft, there shouldn’t be a big concern for security because the sensitive data of each user is stored in the cloud that is protected by Microsoft.

## Glossary of Key Terms

* **Business:** A business will accept a contract and they will have their own code to the GrizzTime website.
* **Contract:** A contract is offered to a business for a certain amount of hours.
* **Project Manager:** The one who manages a project.
* **Contractor:** The one who works on the project.
* **Business Admin:** The one who administers the GrizzTime website.

# Project Execution and Planning

## Team Information

Our team consists of five people: Gurnirunjun Shergill, Melissa Palmer, Misael Hernandez, Anthony Robin, and Daniel Salar.

## Tools and Technology

This project will be created using an ASP.NET framework with Microsoft Azure as a source for the database. We will be using Microsoft Visual Studio with the ASP.NET framework. Google Drive will be used to share any documentation. Discord will be used as a way of communicating with the group.

## Project Plan

The project, GrizzTime, has been divided into six sprints:

* **Sprint 1**: Create a Login and user Authentication system
* **Sprint 2**: Be able to create a business and have workers add that business. In addition, new projects can be created.
* **Sprint 3:** A business admin will be able to manage accounts. Contractors will be assigned to different projects. In addition, the pay rate of a contractor can be modified.
* **Sprint 4:** Contractors and project managers will be able to create and submit time sheets. Expense reports will be generated as well.
* **Sprint 5:** Project managers will be able to create invoices for projects and contractors.
* **Sprint 6:** Business admins will be able to view past invoices and contractors will be able to view past timesheets.

## Best Standards and Practices

In the development of this website, it will be best to use GitHub for version control. Each developer will work in their own branch and then merge with the master branch when a task is complete. As time progresses, documentation will be done on the project.

# System Requirement Analysis

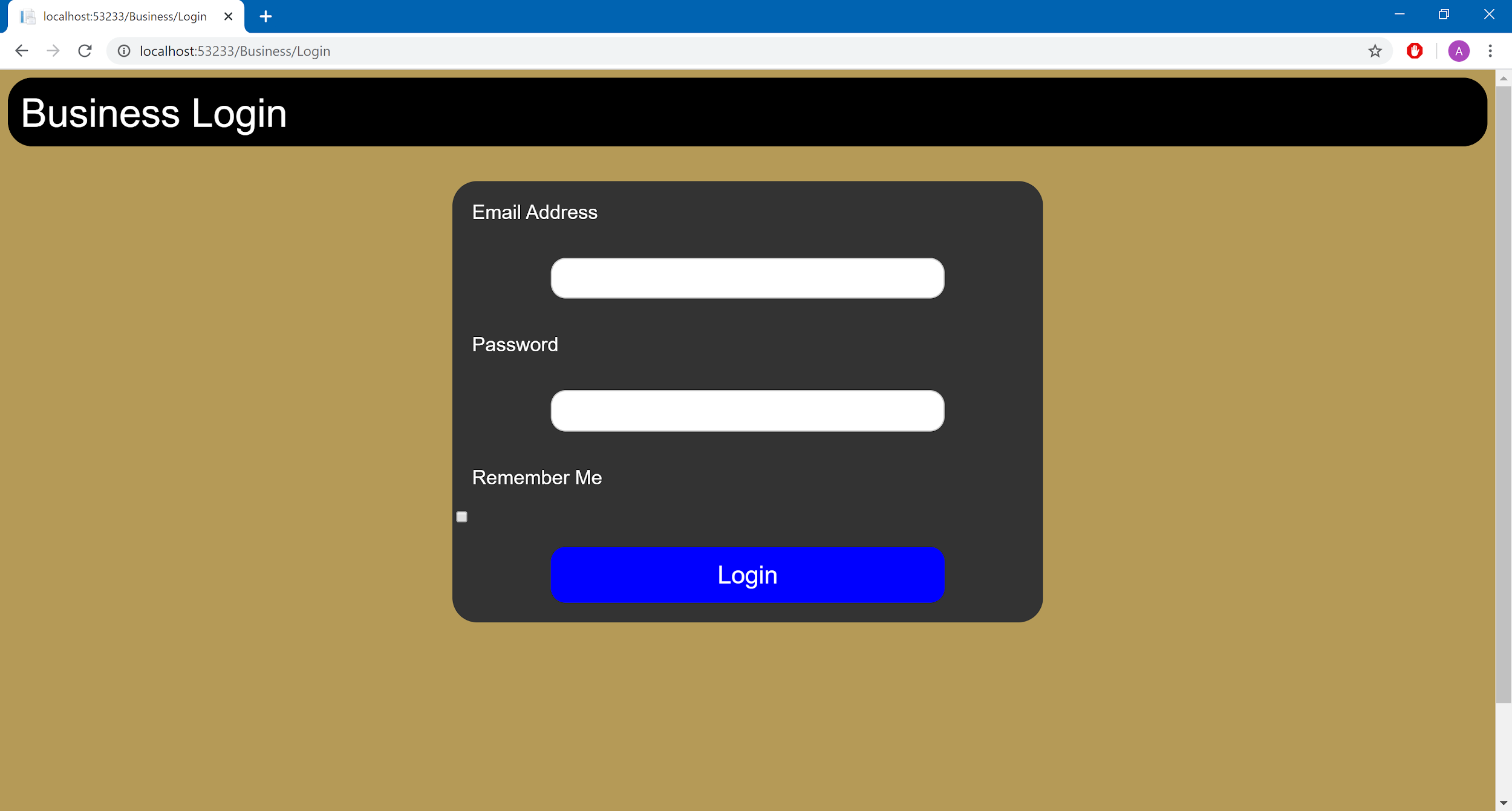
## Functional Requirements

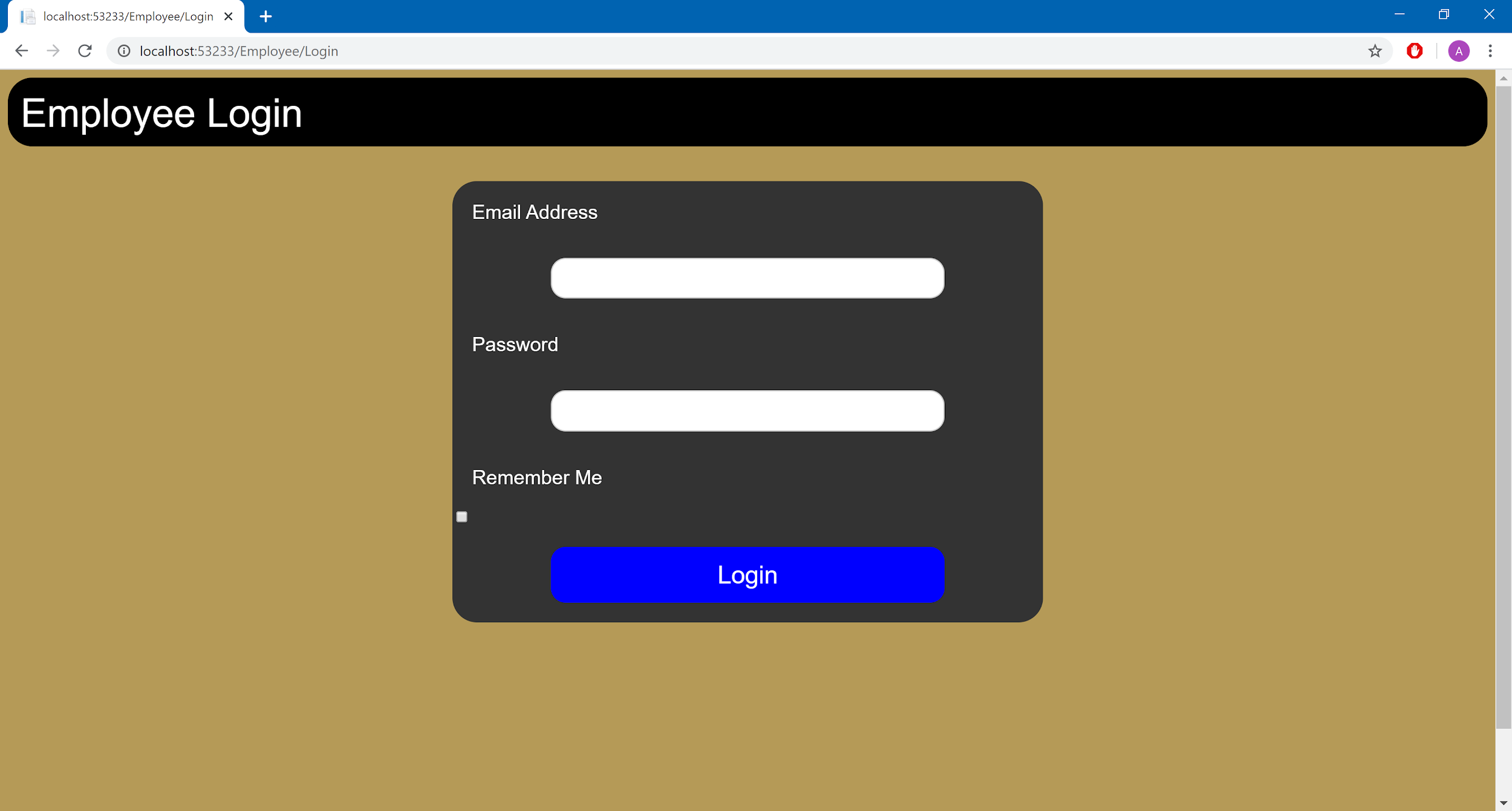
* A user should be able to create an account and connect to a certain business.
* Authentication is implemented by the business. So, the business admin will send out an email to the employee and they will complete the registration from there.
* Each user will have a dashboard that includes their current projects and timesheets.
* The employee should be able to fill out a weekly timesheet and submit it for approval. The project manager should be able to approve this.
* The employee should be able to fill out an expense report for the project and be able to submit it for approval. The project manager should be able to approve this.

## Non-functional Requirements

* If a user forgets their password, a user should be able to reset it.
* The user’s dashboard will be easy to navigate. So, getting details on timesheets or current projects should be simple.
* The timesheet will be easy to input data into.
* This website will have excellent performance and any software bugs will be eliminated.

## On-Screen Appearance of landing and other pages requirements





## Wireframe Designs

[insert here - initial design of website]

# Functional Requirements Specification

## Stakeholders

The stakeholders of this project will be the business, the project managers, and the people working for the project manager.

## Actors and Goals

* **Business Admin**: Will interact with the business and get accounts set up.
* **Project Manager**: Manages the project that an employee is working on. They will fill out timesheets and approve the timesheets of the employees working for them. They should also be able to assign employees to the project they are managing.
* **Employee:** They will fill out the timesheets and they can view their projects.

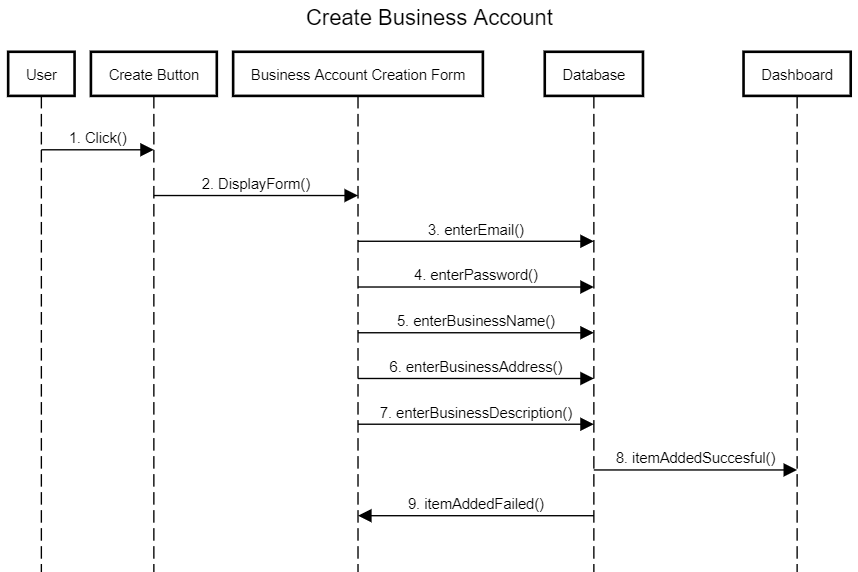
## User stories, Scenarios and Use Cases

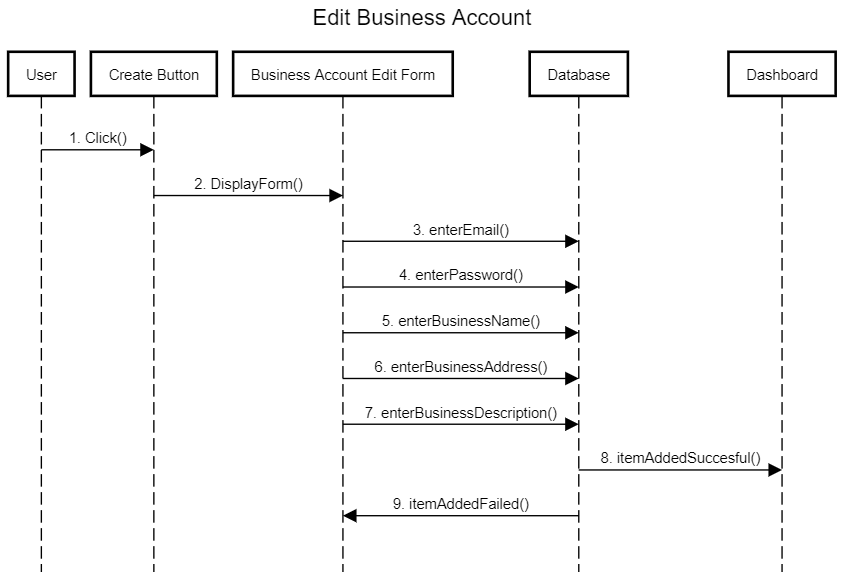
User stories for the development of GrizzTime:

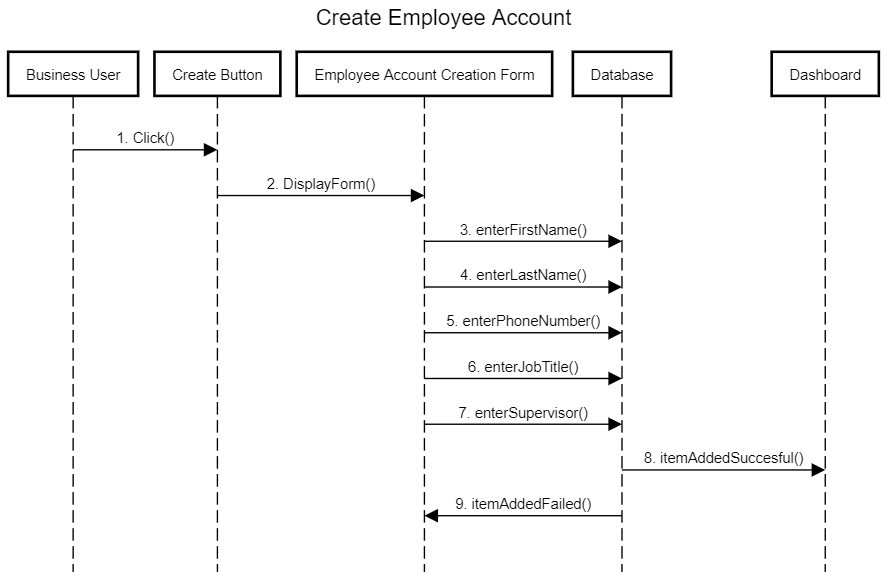
* As a business administrator, I can register an account.
* As a business administrator, I can register an account for my business.
* As a business administrator, I can modify my account.
* As a business administrator, I can modify my business.
* As a business administrator, I can view the users registered to my business.
* As a business administrator, I can assign an employee the role of project manager.
* As a business administrator, I can remove an employee from my business.
* As a business administrator, I can view the projects for my business.
* As a project manager, I can create a new project.
* As a project manager, I can assign a contractor to a specific project.
* As a project manager, I can remove a contractor from a project.
* As a project manager, I can generate an invoice for individual projects.
  1. contractors working on one project may have different hourly rates
* As a project manager, I can generate an invoice for individual contractors.
  1. One contractor may receive different pay rates for different projects.
* As a project manager, I can approve my contractors’ hours.
* As a project manager, I can approve my contractors’ expense reports.
* As a project manager, I should be able to click a button to remind a contractor to submit their time.
  1. ONLY ONCE per week.
* As a contractor, I can enter the time I spent working on a specific project in a week view.
  1. Contractor > Project > sub project categories (e.g. Design, Implementation) (maybe make customizable??)
* As a contractor, I can submit my time for approval on a weekly basis
* As a contractor, I can enter my project expenses into a form.
* As a contractor, I can submit my project expenses for approval.
* As a contractor, I can register a personal account.
* As a contractor, I can edit my personal information.
* As a contractor, I should be able to view my past hours/expenses.
* As a Business Admin, I should be able to view past invoices.
* As a contractor, I should be able to see my hours/expenses from projects I am no longer a part of.
* As a contractor, I can connect my account to my employer’s business.
* As any user, I should have a secured login that locks my account after too many failed attempts to log in.
* As any user, I should be able to change my password after getting locked out of my account.
* As a business administrator, I can view the users registered to my business.
* As a business administrator, I can assign an employee the role of project manager.
* As a business administrator, I can remove an employee from my business.
* As a business administrator, I can view the projects for my business.
* As a project manager, I can assign a contractor to a specific project.
* As a project manager, I can remove a contractor from a project.

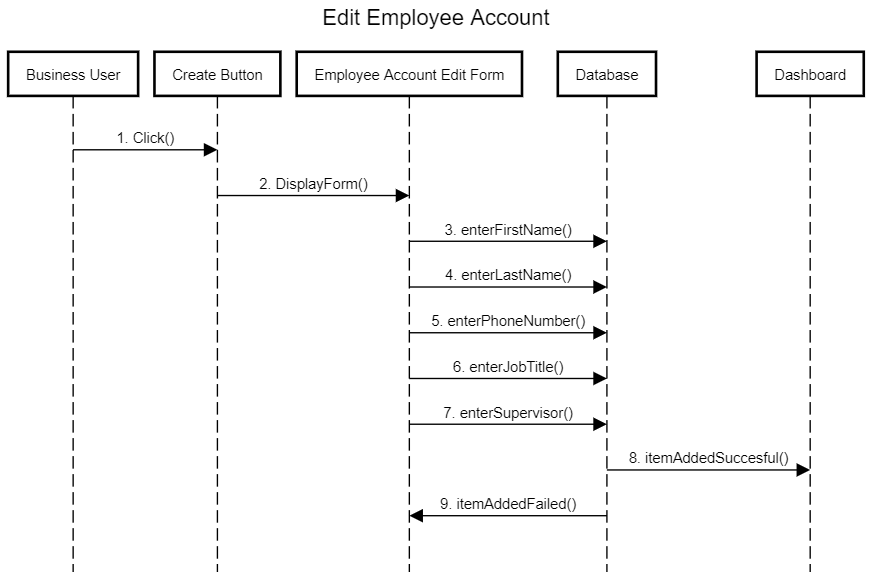
## System Sequence/Activity Diagrams

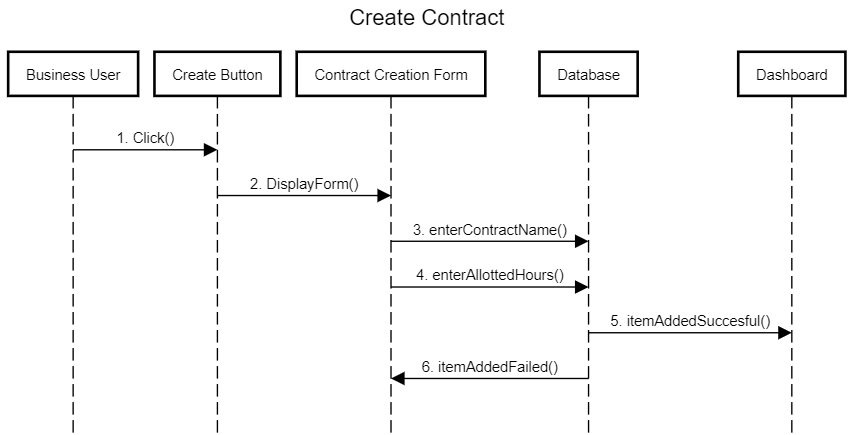
[insert here - screenshot]

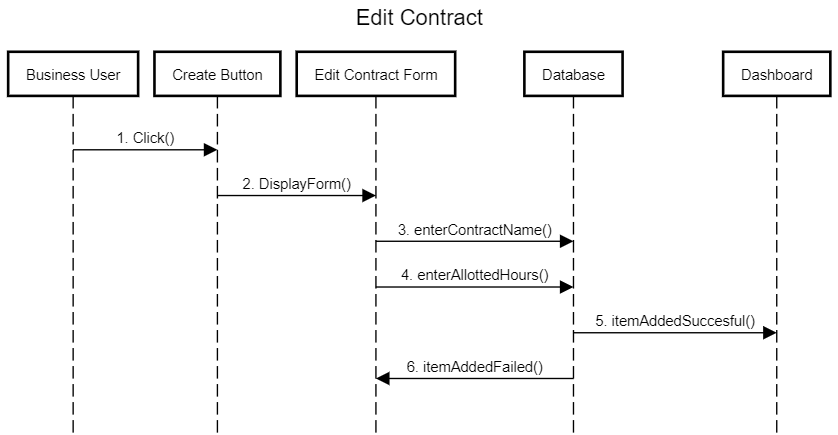


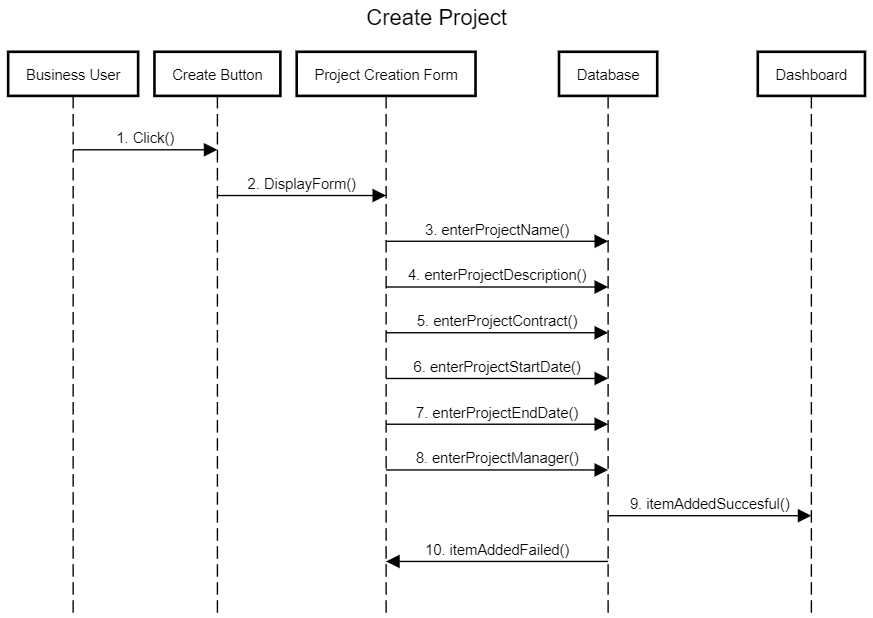


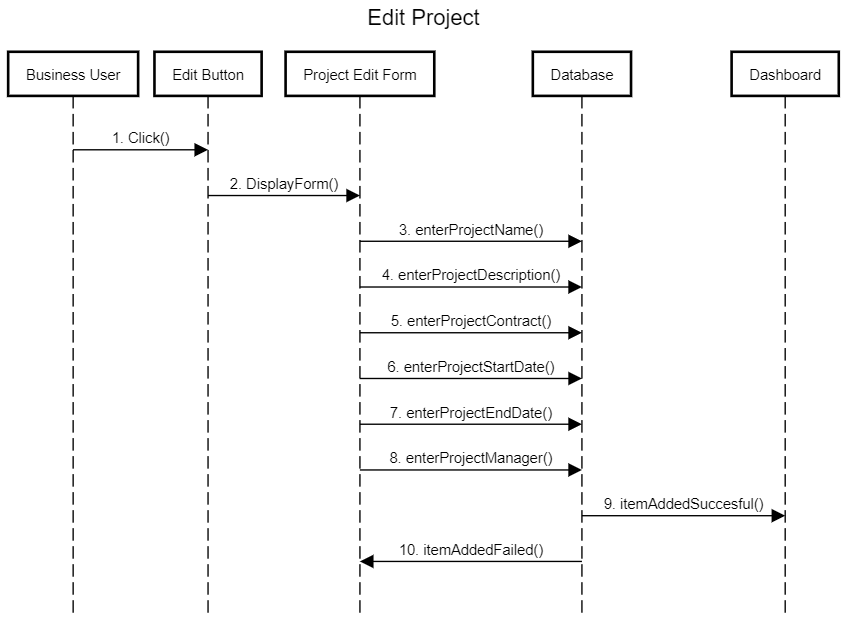


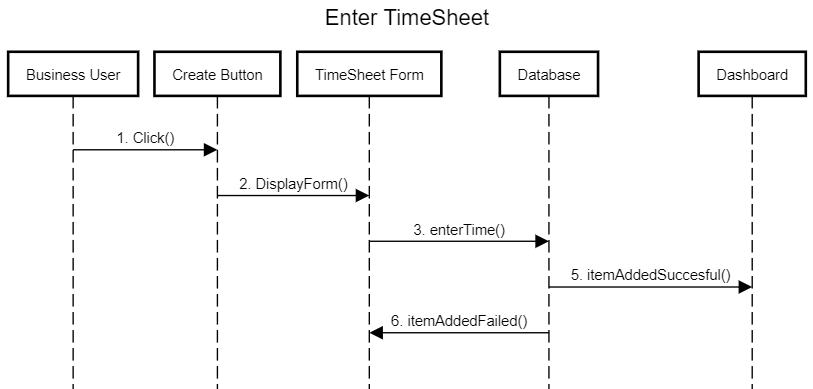


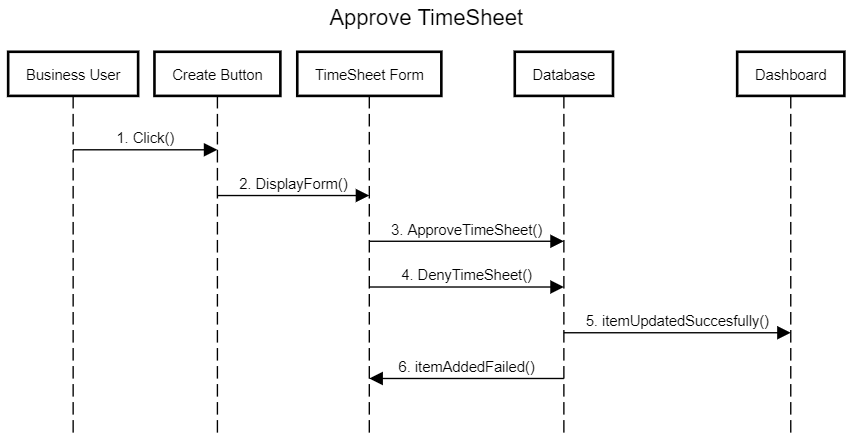












# User Interface Specification

## Preliminary Design

[insert here - screenshot]

## User Effort Estimation

[insert here]

# Static Design

## Class Model

[insert here - screenshot]

## System Operation Contracts

[insert here]

## Mathematical Model

[insert here - screenshot]

## Entity Relation

The following image shows the ERD for the database. This is how the database is structured and how each piece of data is related.

# Dynamic Design

## Sequence Diagrams

[insert here - screenshot]

## Interface Specification

[insert here]

## State Diagrams

[insert here - screenshot]

# System Architecture and System Design

## Subsystems/Component/Design Pattern Identification

[insert here]

## Mapping Subsystems to Hardware (Deployment Diagram)

[insert here - screenshot]

## Persistent Data Storage

To store data, the first action we took was storing the database on a local computer. Once a basic structure was determined, the database was migrated to cloud storage using microsoft azure. This will allow the database to run without interruption.

## Network Protocol

A network protocol is simply a set of rules that the network follows. So, ASP.net primarily follows the HTTP protocol, but it also follows the TCP/IP protocol as well. To enable the communication between the servers and browsers, the HTTP protocol is needed. By following the HTTP protocol, HTML code will be displayed on the browser of the local computer.

## Global Control Flow

[insert here - requires a diagram]

## Hardware Requirement

Since this is a web application, the only requirement needed is access to the internet and an internet browser. By fulfilling these requirements, access to the website should be easy. This implies that the hardware needed is minimal. A standard computer should accomplish this task.

# Algorithms and Data Structures

## Algorithms

[insert here]

## Data Structures

[insert here]

# User Interface Design and Implementation

## User Interface Design

[insert here - screenshot]

## User Interface Implementation

[insert here]

# Testing

## Unit Test Architecture and Strategy/Framework

The unit test strategy did not focus on bugs, but it focused on examining each part of the code separately.

## Unit Test Definition, Test Data Selection

[insert here]

## System Test Specification

[insert here]

## Test Report per Sprint

A report for each sprint is documented in a QA Testing spreadsheet that tracks each tested use case. In addition, this spreadsheet tracks many other tests for software.

# Project Management

## Project Plan

The project is divided into six sprints as mentioned in section 2.3. Once those sprints are accomplished, the project will be complete.

## Risk Management

The strategy used for risk management is shown below:

* Identify the risk
* Analyze the risk
* Evaluate the risk
* Rank the risk
* Treat the risk
* Monitor the risk

If any risks came up, it was dealt with swiftly by first identifying if there is a risk at all. Then, the risk is analyzed and evaluated to see how critical it could be. The risk is then ranked by priority. Once that is determined, the risk is then treated. After that, the risk will be monitored in case it causes an issue again.