***Software Engineering***

***Software Requirements Specification***

***(SRS) Document***

**[PIN Activated Gas Pump]**

**[2/14/2023]**

**[V1]**

**By: William Vaughan**

**Grayson Williams**

**Matthew Ingallinera**

**[Honor Code]**

**Table of Contents**

[1. Introduction 3](#_Toc126197756)

[1.1. Purpose 3](#_Toc126197757)

[1.2. Document Conventions 3](#_Toc126197758)

[1.3. Definitions, Acronyms, and Abbreviations 3](#_Toc126197759)

[1.4. Intended Audience 4](#_Toc126197760)

[1.5. Project Scope 4](#_Toc126197761)

[1.6. Technology Challenges 4](#_Toc126197762)

[1.7. References 4](#_Toc126197763)

[2. General Description 4](#_Toc126197764)

[2.1. Product Perspective 4](#_Toc126197765)

[2.2. Product Features 4](#_Toc126197766)

[2.3. User Class and Characteristics 5](#_Toc126197767)

[2.4. Operating Environment 5](#_Toc126197768)

[2.5. Constraints 5](#_Toc126197769)

[2.6. Assumptions and Dependencies 5](#_Toc126197770)

[3. Functional Requirements 5](#_Toc126197771)

[3.1. Primary 5](#_Toc126197772)

[3.2. Secondary 5](#_Toc126197773)

[4. Technical Requirements 6](#_Toc126197774)

[4.1. Operating System and Compatibility 6](#_Toc126197779)

[4.2. Interface Requirements 6](#_Toc126197780)

[4.2.1. User Interfaces 6](#_Toc126197781)

[4.2.2. Hardware Interfaces 6](#_Toc126197782)

[4.2.3. Communications Interfaces 6](#_Toc126197783)

[4.2.4. Software Interfaces 6](#_Toc126197784)

[5. Non-Functional Requirements 6](#_Toc126197785)

[5.1. Performance Requirements 6](#_Toc126197786)

[5.2. Safety Requirements 7](#_Toc126197787)

[5.3. Security Requirements 7](#_Toc126197788)

[5.4. Software Quality Attributes 7](#_Toc126197789)

[5.4.1. Availability 7](#_Toc126197790)

[5.4.2. Correctness 7](#_Toc126197791)

[5.4.3. Maintainability 7](#_Toc126197792)

[5.4.4. Reusability 7](#_Toc126197793)

[5.4.5. Portability 7](#_Toc126197794)

[5.5. Process Requirements 7](#_Toc126197795)

[5.5.1. Development Process Used 7](#_Toc126197796)

[5.5.2. Time Constraints 7](#_Toc126197797)

[5.5.3. Cost and Delivery Date 7](#_Toc126197798)

[5.6. Other Requirements 7](#_Toc126197799)

[5.7. Use-Case Model Diagram 8](#_Toc126197800)

[5.8. Use-Case Model Descriptions 8](#_Toc126197801)

[5.8.1. Actor: Actor Name (Responsible Team Member) 8](#_Toc126197802)

[5.8.2. Actor: Actor Name (Responsible Team Member) 8](#_Toc126197803)

[5.8.3. Actor: Actor Name (Responsible Team Member) 8](#_Toc126197804)

[5.9. Use-Case Model Scenarios 8](#_Toc126197805)

[5.9.1. Actor: Actor Name (Responsible Team Member) 8](#_Toc126197806)

[5.9.2. Actor: Actor Name (Responsible Team Member) 9](#_Toc126197807)

[5.9.3. Actor: Actor Name (Responsible Team Member) 9](#_Toc126197808)

[6. Design Documents 9](#_Toc126197809)

[6.1. Software Architecture 9](#_Toc126197810)

[6.2. High-Level Database Schema 9](#_Toc126197811)

[6.3. Software Design 9](#_Toc126197812)

[6.3.1. State Machine Diagram: Actor Name (Responsible Team Member) 9](#_Toc126197813)

[6.3.2. State Machine Diagram: Actor Name (Responsible Team Member) 9](#_Toc126197814)

[6.3.3. State Machine Diagram: Actor Name (Responsible Team Member) 9](#_Toc126197815)

[6.4. UML Class Diagram 9](#_Toc126197816)

[7. Scenario 10](#_Toc126197817)

[7.1. Brief Written Scenario with Screenshots 10](#_Toc126197818)

# Introduction

## Purpose

The goal is to implement a simple gas pump control program for use in a fleet environment.

## Document Conventions

The purpose of this Software Requirements Document (SRD) is to describe the client-view and developer-view requirements for the PIN Gas Pump (PGP). Client-oriented requirements describe the system from the client’s perspective. These requirements include a description of the different types of users served by the system. Developer-oriented requirements describe the system from a software developer’s perspective. These requirements include a detailed description of functional, data, performance, and other important requirements.

## Definitions, Acronyms, and Abbreviations

[Include any specialized terminology dictated by the application area or the product area.

For example:]

|  |  |
| --- | --- |
| Java | A programming language originally developed by James Gosling at Sun Microsystems. We will be using this language to build the Restaurant Manager. |
| CSV | Comma Separated Values file, used for storing data |
| SpringBoot | An open-source Java-based framework used to create a micro Service. This will be used to create and run our application. |
| Spring Web | Will be used to access the remote API. This is one of the dependencies of our system. |
| IntelliJ | An integrated development environment (IDE) for Java. This is where our system will be created. |
| API | Application Programming Interface. This will be used to implement a function within the software where the estimated price of delivered fuel is calculated. |

## Intended Audience

[Describe which part of the SRS document is intended for which reader. Include a list of all stakeholders of the project, developers, project managers, and users for better clarity.]

## Project Scope

The goal of the software is to provide a robust, simple pump control system for use in off grid fleet environments. This includes construction, oil fields, military, farms, and other harsh environments. This requires a simple system that requires little to no maintenance and can run off of a small solar array or generator. An interface with high legibility and contrast is also required due to possible use in low visibility conditions. Additionally, where an internet connection is available, a fuel truck driver can get an estimate of the cost for delivered fuel.

## Technology Challenges

Possibly the implementation of a Text or Graphics Based Interface (TUI or GUI). This may or may not be included based on time constraints. While not a concern in the first release, future version that are intended for use in integrated firmware applications would prioritize low power usage.

## References

Alred, F., Brusaw, C., and Oliu, W. (2003). Handbook of Technical Writing (7th ed.). Boston: Bedford/St. Martin’s.

# General Description

## Product Perspective

PGP is a pump control system that is intended to be used in varied environments where a simple, robust system is required.

## Product Features

The product features include the ability for fleet operators to create simple profiles to keep track of fuel use and the ability for administrators to manipulate those profiles. Normal users can access the pump and view lifetime fuel use through a simple PIN system. Fuel tanker truck drivers can also access the system to add fuel to the storage tank. Admins can add and remove users, check storage tank level, and enable or disable the pump. Possible additions include: low tank alerts for admin, capability to manage multiple pumps.

## User Class and Characteristics

The software is intended to be used with minimal to no training. Basic users only have access to between two and four options after entering their PIN including pumping gas and checking lifetime gas pumped. Tanker drivers would only have access to add gas to the tank and check lifetime gas added. Admins will have access to five options including adding and removing users, turning the pump on and off, checking tank level, and pumping gas.

## Operating Environment

The software is intended to be used in a simple, inexpensive system running a stripped down Windows or Linux OS. Ideally, a bootup script would automatically enable the program when the system is switched on.

In the future, the program could be redeveloped to work in a low power microcontroller based system.

## Constraints

[Any limiting factors that would pose challenge to the development of the software. These include both design as well as implementation constraints.]

## Assumptions and Dependencies

[A list of all assumptions that you have made regarding the software product and the environment along with any external dependencies which may affect the project]

The software will be dependent on Spring Web framework to create and execute the text based architecture that will be developed within IntelliJ. The application will also use a //gas price api// to estimate the cost for when fuel is delivered.

# Functional Requirements

[Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.]

## Primary

* FR0: The system will allow the user to pump a predetermined amount of gas.
* FR1: The system will allow the user to check how much gas they have pumped in total
* FR2: The system will allow a tanker truck diver to add gas to the storage tank
* FR3: The system will allow a tanker truck diver to estimate the cost of the added gas after adding
* FR4: The system will allow a tanker truck diver to check the total amount of gas that has been added in total
* FR5: The system will allow an admin to add or remove users from the system
* FR6: The system will allow an admin to enable or disable the pump
* FR7: The system will allow an admin to check the amount of gas in the storage tank

## Secondary

* PIN access for authorized users only.
* API access to retrieve fuel price

# Technical Requirements



## Operating System and Compatibility

The application will be compatible with any operating system that is able to run a Java based program.

## Interface Requirements

### User Interfaces

Will be primarily text based but a GUI is a possibility.

### Hardware Interfaces

A standard keyboard for the admin, although a nine digit keypad is all that a normal user or truck driver needs.

### Communications Interfaces

A normal internet connection to access the API, but it is not strictly required as the program will run without one.

### Software Interfaces

[The interaction of the software to be developed with other software components such as frontend and the backend framework to the used, the database management system and libraries describing the need and the purpose behind each of them.]

We will use React and Spring Boot ThymeLeaf to help build the frontend, as well as JPA for the backend database functionality. We will also use Spring Boot with Java to connect the frontend to the backend.

# Non-Functional Requirements

[Constraints on the services or functions offered by the system (e.g., timing constraints, constraints on the development process, standards, etc.). Often apply to the system as a whole rather than individual features or services.]

## Performance Requirements

[The performance requirements need to be specified for all the functional requirements.]

* NFR0(R): The local copy of the user CSV file will consume less than 5 MB of memory
* NFR1(R): The system (including the local copy of the user CSV file) will consume less than 50MB of memory

## Safety Requirements

[List out any safeguards that need to be incorporated as a measure against any possible harm the use of the software application may cause.]

## Security Requirements

[Privacy and data protection regulations that need to be adhered to while designing of the product.]

* NFR4(R): The system will only be usable by authorized users.

## Software Quality Attributes

### Availability

24/7, as long as there is a power supply, it can function.

### Correctness

Not doing any complex math or algorithms so not an issue.

### Maintainability

None needed as software is intended to be used in an isolated environment. Any future upgrades would be introduced as a whole new stand-alone hardware unit.

### Reusability

Very reusable, code is meant to be easy to upgrade and add functionality to.

### Portability

Very portable, anything running Java can run the software.

## Process Requirements

### Development Process Used

[Software Process Model]

### Time Constraints

Must be finished by April 18, 2023

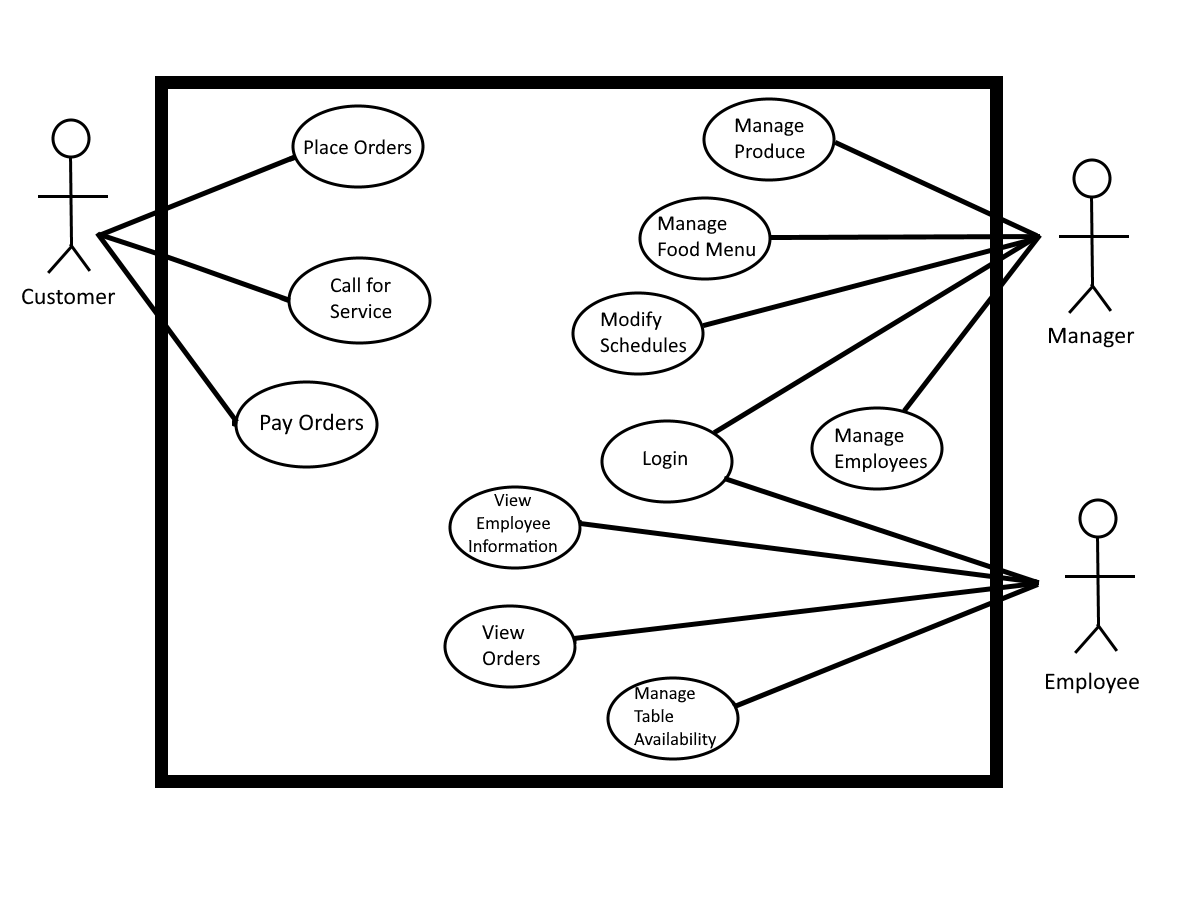
### Cost and Delivery Date

April 18, 2023

## Other Requirements

TBD

## Use-Case Model Diagram



## Use-Case Model Descriptions

### Actor: Admin (William Vaughan)

* **Use-Case Name**: [Brief Use-Case Description]
* **Use-Case Name**: [Brief Use-Case Description]
* **Use-Case Name**: [Brief Use-Case Description]
* **Use-Case Name**: [Brief Use-Case Description]
* **Use-Case Name**: [Brief Use-Case Description]

### Actor: User (Greyson Williams)

* **Use-Case Name**: [Brief Use-Case Description]
* **Use-Case Name**: [Brief Use-Case Description]

### Actor: Tanker Driver (Matthew Ingallinera)

* **Use-Case Name**: [Brief Use-Case Description]
* **Use-Case Name**: [Brief Use-Case Description]

## Use-Case Model Scenarios

### Actor: Actor Name (Responsible Team Member)

* **Use-Case Name**:
  + **Initial Assumption**:
  + **Normal**:
  + **What Can Go Wrong**
  + **Other Activities**:
  + **System State on Completion**:
* **Use-Case Name**:
  + **Initial Assumption**:
  + **Normal**:
  + **What Can Go Wrong**:
  + **Other Activities**:
  + **System State on Completion**:

### Actor: Actor Name (Responsible Team Member)

* **Use-Case Name**:
  + **Initial Assumption**:
  + **Normal**:
  + **What Can Go Wrong**:
  + **Other Activities**:
  + **System State on Completion**: Log-out back to PIN menu
* **Use-Case Name**:
  + **Initial Assumption**:
  + **Normal**:
  + **What Can Go Wrong**:
  + **Other Activities**:
  + **System State on Completion**: Return to user menu

### Actor: Actor Name (Responsible Team Member)

* **Use-Case Name**:
  + **Initial Assumption**:
  + **Normal**:
  + **What Can Go Wrong**:
  + **Other Activities**:
  + **System State on Completion**:
* **Use-Case Name**:
  + **Initial Assumption**:
  + **Normal**:
  + **What Can Go Wrong**:
  + **Other Activities**:
  + **System State on Completion**:

# Design Documents

## Software Architecture

## High-Level Database Schema

## Software Design

### State Machine Diagram: Actor Name (Responsible Team Member)

### State Machine Diagram: Actor Name (Responsible Team Member)

### State Machine Diagram: Actor Name (Responsible Team Member)

## UML Class Diagram

# Scenario

## Brief Written Scenario with Screenshots