Team 18 (Super Project X 3000)

1. **Sean Dang**
   * **Student ID**: 026938455
   * **Email**: sean.dang@student.csulb.edu
2. **Alvin Nguyen**
   * **Student ID**: 017825533
   * **Email**: alvin.nguyen05@student.csulb.edu
3. **Justin To**
   * **Student ID**: 026215915
   * **Email**: justin.to@student.csulb.edu
4. **Nhu Nguyen**
   * **Student ID**: 028870632
   * **Email**: nhu.nguyen07@student.csulb.edu
5. **Brian Anaya**
   * **Student ID**: 028766489
   * **Email**: briant.anaya01@student.csulb.edu

## Preface

(Describe the purpose of this document, its expected readership and its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.)

|  |  |  |
| --- | --- | --- |
| Version | Date | Changes |
| 1.0 | 2/31/XXXX | Initial Version/Github |
| 1.1 | 15th of Augtember, YYYY | Added Glossary |
| 2.0 | 5/10/24 | Fixed version date notation, … |

### Purpose

This document serves as a comprehensive guide for the development and understanding of the software project titled "Super Project X 3000."

### Audience

The intended audience of this document includes project stakeholders, developers, testers, and anyone involved in the project lifecycle.

## Introduction

(Introduce the software project, its goals, and the problem it aims to solve)

### Project Overview

"Car Finder" is a web-based system designed for anyone who is looking for a car. Users can choose what car they want to look for such as the make, model or year, the general area through a zip code, and a price range.

### Project Goals

* Implement efficient search algorithm for cars with specific criteria set by the user to provide car listings.
* Implement a web scraper to search for cars on the web for the most up-to-date car deals.
* Make the front-end simple and intuitive for everyone to use.

## Glossary

(Define key terms and acronyms used throughout the document, unless they are commonly known to each possible stakeholder (e.g., “Cell phone”) AND used with their common meaning. Do not expect your stakeholders to be experts. If in doubt, define a term.)

* **Listing/List:** The list of cars provided to users after the system searched all available cars based on user’s preferences.

## User Requirements and Use Cases

(Outline what the system must do from the user's perspective. User stories need to use the format discussed in class and on our slides. Use cases provide detailed scenarios of system interactions.)

### User Stories

(A collection of user stories that apply to the project.)

1. As a registered user, I want to log in securely so that I cannot be held responsible for someone else’s actions.
2. As an inventory manager, I need to add new products to the inventory so that the database matches the physical inventory.
3. As a sales manager, I want to generate sales reports by date range so that the upper management can be kept up to date on the company’s performance.

### Use Case: Adding a New Product

|  |  |
| --- | --- |
| Identifier | UC-2 Add new Product to Inventory |
| Purpose | Update inventory with new product name and quantity |
| Requirements | User Story #2 |
| Development Risks | None |
| Pre-conditions | Inventory manager is logged in and on inventory management page |
| Post-conditions | New product is added to inventory |

**Table 1: Typical Course of Action**

|  |  |  |
| --- | --- | --- |
| Seq# | Actor’s Action | System’s Response |
| 1 | Inventory manager selects “Add new product” |  |
| 2 | Enters product details (name, SKU, quantity) |  |
| 3 | Clicks “save” | Runs plausibility checks on submitted information |
|  |  | Checks are successful |
|  |  | Saves submitted information in inventory database |
|  |  |  |

**Table 2: Alternate Course of Action**

|  |  |  |
| --- | --- | --- |
| Seq# | Actor’s Action | System’s Response |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Table 3: Exceptional Course of Action**

|  |  |  |
| --- | --- | --- |
| Seq# | Actor’s Action | System’s Response |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

## System Architecture

(Describe the high-level design of the software.)

### Components

1. **Frontend**: Web-based user interface (built with React/HTML/CSS/JavaScript).
2. **Backend**: RESTful API (built with Node.js and Express).
3. **Database**: MySQL for data storage.
4. **Authentication**: OAuth 2.0 for user authentication.

### Deployment Diagram