**D424 – Software Engineering**

**Task 3**

|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | http://www.idevnews.com/views/images/uploads/general/wgu_logo.png  Bike Shop Inventory Desktop Application |
| **Student Name:** | Fabio Tran |

**Table of Contents**

Contents

[Table of Contents 2](#_Toc151586064)

[Application Design and Testing 5](#_Toc151586065)

[Class Design 5](#_Toc151586066)

[UI Design 9](#_Toc151586067)

[Unit Test Plan 13](#_Toc151586068)

[Introduction 13](#_Toc151586069)

[Purpose 13](#_Toc151586070)

[Overview 13](#_Toc151586071)

[Test Plan 13](#_Toc151586073)

[Items: 13](#_Toc151586074)

[Deliverables: 13](#_Toc151586075)

[Pass/Fail Criteria: 14](#_Toc151586085)

[Specifications 15](#_Toc151586088)

[Procedures 16](#_Toc151586089)

[Results 16](#_Toc151586097)

[User Guide for Initial Setup & Running the Application 17](#_Toc151586098)

[User Guide for Running the Application from User Perspective 19](#_Toc151586099)

[Introduction 19](#_Toc151586100)

[Login 19](#_Toc151586101)

[Change Password Page 20](#_Toc151586102)

[Dashboard Page 21](#_Toc151586103)

[Inventory Page 22](#_Toc151586104)

[Add Spare Bike to Inventory 23](#_Toc151586105)

[Delete Bike Part 24](#_Toc151586106)

[Edit/modify Bike Part 25](#_Toc151586107)

[Searching For Bike Part 26](#_Toc151586108)

[Import/Export Bike Part Reports 27](#_Toc151586109)

[Activity Logs Page 28](#_Toc151586110)

[Users Page 29](#_Toc151586111)

[Add User 30](#_Toc151586112)

[GitLab Repository & Branch History 31](#_Toc151586113)

[GitLab Repository Link: 31](#_Toc151586114)

[GitLab Branch history: 31](#_Toc151586115)

[Panopto Video Link 31](#_Toc151586116)

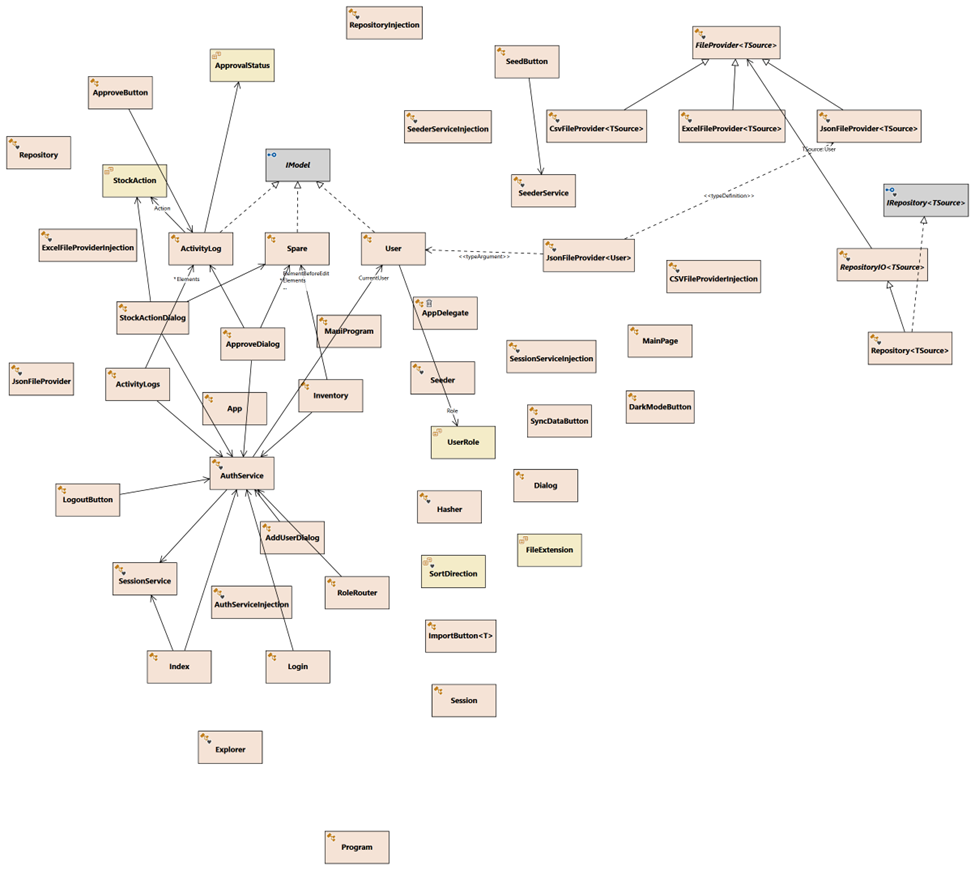
# Application Design and Testing

## Class Design

The Bike Shop Inventory Desktop Application harnesses the capabilities of the .NET MAUI Blazor framework to create a robust class structure focused on efficient inventory management. This design embodies a modular and well-organized approach, emphasizing encapsulation, inheritance, and polymorphism within the core classes: Inventory, User, AuthService for authentication, and FileProvider for auditing. These classes form the backbone of the system, enabling precise tracking of bike parts and ensuring secure user access while maintaining comprehensive audit trails.

By leveraging .NET MAUI Blazor, the Bike Inventory System embraces advanced Object-Oriented Programming (OOP) principles to ensure scalability and maintainability. The architectural diagram delineates the relationships and methods among these classes, fostering code reusability and facilitating seamless feature integration to meet evolving business demands.

In essence, the Bike Shop Inventory Desktop Application represents a meticulous fusion of cutting-edge technology and proven software development methodologies by prioritizing a modular and systematic approach, it not only ensures efficient inventory management but also lays a strong foundation for future enhancements and innovations. This commitment to robust design principles establishes the application as a reliable solution for bike shop inventory management, poised to meet the dynamic needs of the industry while maintaining a high level of performance and usability.

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

## UI Design

The User Interface (UI) of the Bike Spares Inventory System is crafted with user-centric principles, offering an intuitive and visually appealing experience. Built on the .NET MAUI Blazor framework, the UI design prioritizes functionality without compromising on aesthetics. It incorporates a responsive layout with carefully crafted elements, providing seamless navigation and accessibility across various devices and screen sizes. The UI design initiates with a secure login screen, ensuring access control to the application.

A screenshot of a computer

Description automatically generated

Upon login, users can access an informative dashboard presenting a comprehensive graphed overview of bike parts data.A screenshot of a computer

Description automatically generated

The application's primary functions revolve around managing users and bike parts, with dedicated sections for each. Users can effortlessly add, modify, and remove bike parts and user profiles through user-friendly forms and actions. Prioritizing a seamless and effective user experience, the UI aims to simplify the process of managing and monitoring bike parts and user profiles.

A screenshot of a computer

Description automatically generated

Beyond its core functionalities, the application boasts a robust reporting section, empowering users to generate insightful reports based on bike parts and user data. These reports offer valuable insights, facilitating data-driven decisions to optimize operational strategies. Admins can also generate activity logs based on inventory changes, and control access privileges across Admins/staff accounts.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# Unit Test Plan

## Introduction

### Purpose

The purpose of this test plan is to validate the functionality of the **Inventory** class methods (**AddItem** and **DeleteItem**) using Xunit testing methods. The tests aimed to ensure that items can be added to and removed from the inventory accurately and efficiently. Remediation, if necessary, involved code adjustment to resolve failing tests and ensure expected behavior.

### Overview

## The Xunit tests were designed to validate the behavior of the Inventory class methods, specifically AddItem and DeleteItem, within the context of the project's inventory management system. These tests were unique to this specific module of the codebase, focusing on the core functionalities related to managing inventory items.

## Test Plan

### Items:

**Features:** The tests covered the following functionalities:

1. **-AddItem**: To verify the addition of an item to the inventory.
2. **-DeleteItem**: To validate the removal of an item from the inventory.

### Deliverables:

### The test results included Xunit test reports that documented the pass or fail status of each test case.

### Tasks:

### Define test cases for AddItem and DeleteItem.

### Implement Xunit test methods to validate the behaviors of these methods.

### Identify and address any failures in the test results by adjusting the code logic.

### Needs:

### Xunit testing framework.

### Development environment with access to the Inventory class.

### Visual Studio or other IDE for code modification. A screenshot of a computer Description automatically generated

### Pass/Fail Criteria:

## Pass Criteria: All test cases must pass without any failures.

## Fail Criteria: If any test case fails, the recourse involves debugging the code, identifying the issue, and rectifying it. Failed test cases trigger code modifications and retesting.

## Specifications

Provide sample code that represents what testing code was used. Screenshots are acceptable.

A computer screen shot of a program

Description automatically generated

## Procedures

## Test Case Preparation: Define test scenarios for AddItem and DeleteItem.

## Test Implementation: Write Xunit test methods based on defined scenarios.

## Execution: Execute the tests to verify the functionalities.

## Result Analysis: Review Xunit test reports to identify pass/fail results.

## Debugging (if required): If any test fails, debug the code and modify accordingly.

## Retesting: After code adjustments, rerun the failed test cases.

## Final Review: Ensure all test cases pass before finalizing the testing process.

## Results

The Xunit test results indicated that all test cases passed successfully. Screenshots of Xunit test reports are below, showcasing the green status indicating passing tests for both **AddItem** and **DeleteItem** methods in the **Inventory** class.

A screenshot of a computer

Description automatically generated

**User Guide for Initial Setup & Running the Application**

**Introduction:**

This User Guide aims to provide comprehensive instructions for installing and running the Bike Shop Inventory Desktop Application. It includes step-by-step procedures for installing Visual Studio 2019/2022, building the solution, and running the application on the Windows platform through the debugger. The guide is designed to assist users, particularly those with a basic IT background, in setting up the application flawlessly.

**Installation and Using the Application:**

**1. Installing Visual Studio 2019/2022:**

* Download Visual Studio from the provided link: <https://visualstudio.microsoft.com/vs/>
* Double-click the downloaded file to initiate the installation process.

A screenshot of a computer

Description automatically generated

* Follow the on-screen instructions to install Visual Studio 2019/2022. Select required components (ensure ".NET MAUI Blazor" is included).
* Complete the installation and launch Visual Studio 2019/2022.

**2. Setting Up the Bike Shop Inventory Application:**

* Clone the application ZIP file from the GitLab repository or proceed from the provided zip file.
* GitLab Link - <https://gitlab.com/DevolaRe/bikeshopproj>A blue button with white text

  Description automatically generated
* Extract the contents of the ZIP file to your desired location.
* Open Visual Studio.
* Choose "Open a project or solution" and navigate to the extracted folder.
* Select the solution file (usually with a **.sln** extension) and click "Open" to load the solution.

**3. Building the Solution:**

* Once the solution is loaded, navigate to the "Build" menu at the top.
* Select "Build Solution" to compile the project. Ensure there are no build errors.

A screenshot of a computer error

Description automatically generated

**4. Running the Application:**

* Make sure a Windows machine is selected as the target platform.
* Set the startup project by right-clicking on the project file and choosing "Set as StartUp Project."
* Hit the "Start Debugging" button (or press F5) to run the application on the debugger.
* The application should now launch, displaying the login screen.

**5. Logging In and Using the Application:**

* Firs, click on “PRESS TO SEED USER DATA” on the upper-left of the screen.
* Use the initial provided credentials to login, as this is a demo account.
* Click LOGIN.
* Upon successful login, you'll be directed to the dashboard.
* Explore the features as outlined in the application's documentation or feature list.

**Conclusion:**

This User Guide provides a streamlined process for installing Visual Studio, setting up, building, and running the Bike Shop Inventory Application on a Windows machine. Following these steps meticulously should ensure a smooth and error-free setup, enabling users to access and utilize the application's functionalities seamlessly.

**User Guide for Running the Application from User Perspective**

**Introduction**

This user guide is designed to guide you through the login process and offer a comprehensive overview of all available functions within the application.

**Login**

To assist with the application demo, data seeding for testing was included. This function initializes admin-user auto-creation upon the user’s first application loading.

Use the generated Username/Password that is seeded from the upper-left “PRESS TO SEED USER DATA” button. This account should have admin-user privileges.

A black square with white text

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated

**Change Password Page**

Once first logged into application, after account creation, there is the option of replacing the auto-generated password for the current admin account. This design directs the new user to change their password, if initial/system-generated password detected.

Skip the request to change the password, or change it accordingly, but it’s not required for the demo of the application.

A screenshot of a computer

Description automatically generated

**Dashboard Page**

To visualize the current Inventory Items, simply navigate to the Dashboard page by selecting the "Dashboard" option tab from the left-hand menu.

The dashboard metrics includes:

* Approved Deducted Items Quantity
* Pending Deducted Items Quantity
* Available Items Quantity
* Disapproved Deducted Items Quantity

A screenshot of a computer

Description automatically generated

**Inventory Page**

To manage Bike Parts, such as adding, deleting, editing, searching, or importing/exporting reports, simply navigate to all Inventory page by selecting the "Inventory" option tab from the left-hand menu.

A screenshot of a computer

Description automatically generated

**Add Spare Bike to Inventory**

Select the "ADD SPARE" button to begin the process of adding a bike part. Please ensure all fields are completed on this page.

Upon clicking the "ADD SPARE" button, the application will redirect you to the "ADD SPARE" form, upon completion of filling the form and adding the new part, there should be a completion notification.

1.A blue and purple rectangle with text

Description automatically generated

2.A screenshot of a phone

Description automatically generated3. A green rectangle with white text

Description automatically generated

**Delete Bike Part**

To delete an existing bike, navigate to the "Inventory page" and locate the bike part you wish to update. Use the upper-right search bar if required.

A screenshot of a computer

Description automatically generated

Click the "-" button below the Action category corresponding to the desired bike part to delete the bike part. \*\*\*Important Note: Deletion of bike parts is constrained by weekdays MON-FRI and Work time 9AM - 4PM. This is mimicked in real time to emulate a production environment at the bike company.

A red rectangular sign with white text

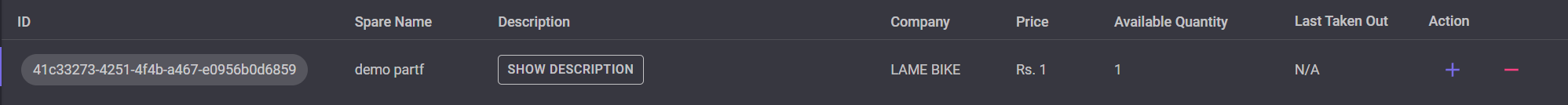
Description automatically generated

**Edit/modify Bike Part**

To edit an existing bike part, navigate to the "Inventory page" and locate the bike part you wish to update. Use the upper-right search bar if required. Click the open space in the selected row of the part, which corresponding to the desired part. The menu should open to allow real-time edits to the information currently available.

A screen shot of a computer

Description automatically generated

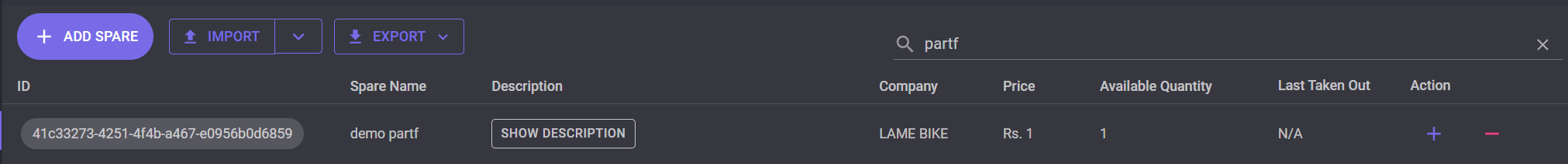
Ensure all fields are completed on the editing page. Once you click the Check button below the Action Tab, the application completes and saves the edit of the current selected bike part on the "Inventory page.” The updated information for the selected bike part is now saved. 

A green rectangle with white text

Description automatically generated

**Searching For Bike Part**

To search for an existing bike part, navigate to the "Inventory page" and utilize the upper-right search menu. Click the open space in the selected menu and type related keywords, which correspond to the desired part. The menu should now display the corresponding matching search query if there is a matching bike part.



**Import/Export Bike Part Reports**

To import/export bike reports, navigate to the "Inventory page" and locate import/export button located to the upper-left portion of the menu.

Upon clicking “Import” button, the new bike report must be in .csv, .json, or .xlsx, and Windows file manager would open to allow the user to select their file to import.



Upon clicking “Export” button, select one of the three option formats, and the current Bike Inventory report will be saved automatically to the user’s desktop.

A screenshot of a computer

Description automatically generated A white and green square with a green x and a black background

Description automatically generated

**Activity Logs Page**

To view, import/export Inventory Activity Logs, go to the "Activity Logs" tab on the left-hand menu.

A screenshot of a computer

Description automatically generated

Upon clicking “Import” button, the new inventory activity report must be in .csv, .json, or .xlsx, and Windows file manager would open to allow the user to select their file to import.

A blue arrows in a square

Description automatically generated with medium confidence

Upon clicking “Export” button, select one of the three option formats, and the current inventory activity report will be saved automatically to the user’s desktop.

A screenshot of a computer

Description automatically generated A white and green square with a green x and a black background

Description automatically generated

**Users Page**

To view, import/export Users Logs, go to the " Users " tab on the left-hand menu.

A screenshot of a computer

Description automatically generated

Upon clicking “Import” button, the new Users report must be in .csv, .json, or .xlsx, and Windows file manager would open to allow the user to select their file to import.

A blue arrows in a square

Description automatically generated with medium confidence

Upon clicking “Export” button, select one of the three option formats, and the current Users report will be saved automatically to the user’s desktop.

A screenshot of a computer

Description automatically generated A white and green square with a green x and a black background

Description automatically generated

**Add User**

To add a new User, click the "Add User" button. All fields are required, and valid information must be added to each field constraints. Once you click the "Add" button, the application takes you back to the original Users page and you should see your newly added Users.

A screenshot of a login form

Description automatically generated A green and black rectangle with white text

Description automatically generated

# GitLab Repository & Branch History

## GitLab Repository Link:

<https://gitlab.com/DevolaRe/bikeshopproj>

## GitLab Branch history:

**A screenshot of a computer

Description automatically generated**

**Panopto Video Link**

Name: Fabio T - Capstone Vid

Panopto Video Link:

<https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=94b389cd-99fd-4c50-8fc5-b0c2013e63d8>