ByteType Test plan

Changelog

Version	Change Date	By	Description
0.1	31.10.2020	JK	Initial document creation

CONTENTS

1. TERMS/ACRONYMS

A mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
API	Application Program Interface

2. INTRODUCTION

The project chose the testing method of the WaterFall model and followed the sequence of "Analysis – Design – Implementation -Testing" for software development. During the testing process, automated testing tools are used to perform unit testing on the software, so as to ensure accurate positioning of errors and timely modification.

2.1. Scope

2.1.1. In scope

Back-end:

Application test

AuthController test

LockerController test

ParcelController test

UserController test

Front-end:

Navbar, Locker and Login&Sign up:

Element display, user input, button click, toggle switch,login&sign up feature

2.1.2. Out of scope

Front-end: user panel and driver panel

Compatibility test

Stability test

2.2. Quality Objective

The test project's objective is to expect the application to meet the functional requirements planned during software development, and to ensure that the application under test meets the functional requirements. In addition, one of the goals of software testing is to avoid errors. Effective testing helps to provide a seamless wrong application.

2.3. Roles and Responsibilities

Peng:

- Developed the driver page UI.
- Crafted Auth, User, Parcel, and Locker APIs.
- Conducted API integration tests.

Zhang:

- Designed the User, Locker, and form page UI.
- Integrated User, Locker, and form pages with the back-end API.
- Participated in React testing.

3. TEST METHODOLOGY

3.1. Overview

The test methodology of the WaterFall model was chosen for this project. First of all, since the development of this project has a clear requirement, it will not change during the development process. Secondly, the waterfall model is implemented in a fixed order of " Analysis – Design –Implementation -Testing". This development model can clearly express the whole process of project development. Finally, the waterfall model can enter the next stage of development after completing each stage. Developers only need to focus on the current development stage, which is convenient for the division of labor and cooperation.

3.2. Test levels

Mainly to execute automated unit tests on the AUT. A unit is the smallest testable part of a system or application by which each part of the software can be tested in isolation. Unit tests are easy to find problems in detail, easy to modify and make tracking problems more convenient.

3.3. Bug triage

Classifying bug can quickly determine the priority of bugs that need to be solved, thereby reducing the impact of bugs on development efficiency, and can also evaluate software quality by the severity of bugs found.

Bugs can be categorized as code errors, design flaws, UI page optimization, function implementation errors, and deployment issues

1: Errors such as system crashes, garbled characters or blank pages need to be resolved immediately.

2: Errors that do not affect the operation of the application, Such as failure to implement designed functions or errors in code logic that affect subsequent tests should be resolved within one day.

3: Although an error occurs, the designed function or logic can be realized. Such bugs can be fixed in a few days.

3.4. Suspension Criteria and Resumption Requirements

During the test, the test can be suspended when a bug is found that prevents the test from proceeding normally or when the part under the test needs to be suspended for modification.

Testing can resume after the bug is resolved or the code is modified.

3.5. Test Completeness

80% test coverage

The criteria for completing the test are:

Complete the testing of existing functions and the code has no logical errors.

All manual and automated test cases were executed and all open bugs were fixed.

4. TEST DELIVERABLES

Here mention all the Test Artifacts that will be delivered during different phases of the testing lifecycle.

Test Plan (this document filled)*

Test Cases

Requirement Traceability Matrix

Bug Reports

Test Strategy

Test Metrics

*Documents that are in bold are minimum requirement for this course!

5. RESOURCE & ENVIRONMENT NEEDS

Backend Testing:

Hardware requirement: a device with CPU, RAM and any operating system.

Environment requirement: java 17 sdk or higher

Frontend testing:

Hardware requirement: a device with CPU, RAM and any operating system.

Environment Requirement: Node.Js environment

5.1. Testing Tools

Backend Testing:

Description: The backend apis is test by JUnit with utilization of Spring Boot for comprehensive API testing and use OpenAPI for monitoring API status.

Launch test with **gradlew test**

Frontend testing:

Description: Test by Jest and react test library. This process includes the utilization of mock fetch data to simulate real-world scenarios and assess component functionality accurately.

Launch test with **npm test**