CENTRAL CAR POLICY

TESTING REPORT DOCUMENT

ADVISOR DINDAR OZ

TOLGA ÇOBANOĞLU

INTRODUCTION

In order to prevent all errors that may occur in the program, we must test the smallest microstructure until we find it, so that we can provide a problem-free product to the customer.

In order to provide this service, we must choose the test objectives, test scenarios and which tests we will use properly.

TEST OBJECTIVES

- To find the errors that may occur in the program in the most general way and to solve them as soon as possible with early diagnosis.-I will test whether the actions that users will perform within the application can reach the correct results.

- Investigating that users can manipulate the system with false information.

- Is the information going to the database correctly and securely?

- Verify that the items in the project proposal are provided.

- Improve performance and quality of product.

FUNCTIONAL TESTS

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Function | Description | Testing Method | Success / Fail Status | Automated / Manual | Remarks |
| cityCodeCheckerTest | This method tests the accuracy of the combinations of letters and numbers of the license plate created by the admin. | Unit Testing with JUNIT | Success | Manual | Case for 1 letter-4 digit, 2 letter-3,4 digit and 3 letter 2 digit. |
| cityCodeChecker | It tests the number group in which the license plate with a sub-feature belongs to the city. | Unit Testing with JUNIT | Success | Manual | It success case for [ 01,81]. |
| letterGroupChecker | It checks whether the letters on the plate are characters. | Unit Testing with JUNIT | Success | Manual | It prevents (W,X,Q,İ,Ş,Ü,Ğ) characters not to be on the plate. |
| hashString | While saving users to the database, it checks the accuracy of the hashing method so that their passwords are not decrypted in case of theft of user information. | Unit Testing with JUNIT | Success | Manual | It uses the SHA256 algorithm and generates lowercase letters. |

P.S. The source code of the methods whose test results are written above is available in zip form in the folder of the uploaded file.

NON-FUNCTIONAL TESTS

1.SECURITY

According to Firebase (Google Service) Documentation, “Firebase services encrypt data in transit using HTTPS and logically isolate customer data. In addition, several Firebase services also encrypt their data at rest: Cloud Firestore.” Our data is safe thanks to this process.

Likewise, the passwords of our users are stored as hashed with SHA256.

In the query application that is open to the public, there is a timeout for the security of the data and to prevent the data from being copied (the duration can be changed according to the customer's request).

In order not to sabotage the data with the data coming from the users, every footprint of the institutions is recorded and the admin will be able to view all the transactions easily whenever she/he wants.

In order to prevent external attacks, we send the user an e-mail after registration and ask her to verify her/his e-mail.

2.SURVIVABILITY

In situations that will disrupt the functioning of the system, for example, situations that cannot be communicated while waiting for a request from the database, it is used with an exception and shows the system error to the user. In this way, we aimed to ensure that the system does not stop.

3.AVAILABILITY  
It is possible to use the system daily, with 50,000 reads, 20,000 writes and 20,000 data deletes. Our extra uses are charged by Google.

4.INTEROPERABILITY

Our product is work with Google Services as Firebase Firestore and Authentication.

5.EFFICIENCY

The time limit for the operation is 270 seconds with an idle time of 60 seconds.

Firestore places a limit of 1MB on the amount of data that can be stored in a single document. (Document is similar to object).

Maximum size for a document name 6 KiB.

Maximum writes per second per database 10,000 (up to 10 MiB per second).

Maximum API request size 10 MiB.

6.FLEXIBILITY

It is designed for all Android devices with SDKs between 23 and 31 and able to connect to the internet.

7.PORTABILITY

The application part is written in Java for Android. But the database part is generic. After programming the application part with Swift,Flutter or React Native for iOS, the same database requests will work in the same way after making a request by changing the language.

8.REUSABILITY

I designed the program to be as high-quality as I could, for example packaging, naming, input controls, to guide me when I'm designing an application next time, and so that I can reuse the registration, login, encryption methods.

USE CASE TESTS FOR USE CASES

ADMIN CLIENT

1. ADD NEW LICENSE PLATE

Basic Flow:

1. Admin goes to the page where the operation will be done
2. Enters the desired properties of the plate to be created.
3. Saved data to database.
4. Use case ends.

Alternate Flows:

2A Invalid input

1. Gets the information that the information entered is invalid
2. Use case continues from second step.

3A Save Error

1. Displays an error message about the error it receives.
2. Use case ends.

2.REMOVE EXIST LICENSE PLATE

Basic Flow:

1. Admin goes to the page where the operation will be done
2. Enters the desired properties of the plate to be deleted.
3. Request to database for delete the object from database.
4. Use case ends.

Alternate Flows:

2A Invalid input

1. Gets the information that the information entered is invalid
2. Use case continues from second step.

3A Save Error

1. Displays an error message about the error it receives.
2. Use case ends.

3. VIEW FEEDBACKS

Basic Flow:

1. Admin goes to the page where the operation will be done
2. The data coming from the related table from the database are displayed in time order.
3. Use case ends.

Alternate Flows:

2A Error not getting data

1. The corresponding error is returned
2. Use case ends.

4. VIEW REPORT’S ACTIVITIES

Basic Flow:

1. Admin goes to the page where the operation will be done
2. The data coming from the related table from the database are displayed in time order.
3. Use case ends.

Alternate Flows:

2A Error not getting data

1. The corresponding error is returned
2. Use case ends.

CAR SERVICE CLIENT

1.SIGN IN

BASIC FLOW:

1.The user goes to the relevant page.

2.Enters e-mail and e-mail address information.

3. System checks status of account.

3.Provides access to the home page of the application.

4.Use case ends.

ALTERNATE FLOW:

2A Unregistered User

1.Take relevant error.

2.Use case ends.

3A Unauthorized User

1.Take relevant error.

2.Use case ends.

2.SIGN UP

BASIC FLOW:

1.The user goes to the relevant page.

2. Enters the information requested by the application into the system.

3. A confirmation mail is sent to the entered e-mail address.

4.Use case ends.

ALTERNATE FLOW:

2A Invalid inputs

1. Take relevant error.

2.Use case ends.

3.CREATE NEW REPORT FOR LICENSE PLATE

BASIC FLOW:

1. The user completes the sign in process.

2. Goes to the relevant page.

3.She/he prepares and saves the report on the work done to the vehicle that came to the service.

4.Use case ends.

ALTERNATE FLOW:

3A Invalid inputs

1. Take relevant error.

2.Use case ends.

3.MAKE A QUERY FOR LICENSE PLATE

BASIC FLOW:

1.The user enters the relevant query field and determines the query type.

2. It appears on the screen where it will enter the vehicle license plate to be inquired and enters the information.

3. Relevant data comes from the database.

4.Use case ends.

ALTERNATE FLOW:

2A Invalid inputs

1. Take relevant error.

2.Use case ends.

EXPERT CLIENT

1.SIGN IN

BASIC FLOW:

1.The user goes to the relevant page.

2.Enters e-mail and e-mail address information.

3. System checks status of account.

3.Provides access to the home page of the application.

4.Use case ends.

ALTERNATE FLOW:

2A Unregistered User

1.Take relevant error.

2.Use case ends.

3A Unauthorized User

1.Take relevant error.

2.Use case ends.

2.SIGN UP

BASIC FLOW:

1.The user goes to the relevant page.

2. Enters the information requested by the application into the system.

3. A confirmation mail is sent to the entered e-mail address.

4.Use case ends.

ALTERNATE FLOW:

2A Invalid inputs

1. Take relevant error.

2.Use case ends.

3.CREATE NEW REPORT FOR LICENSE PLATE

BASIC FLOW:

1. The user completes the sign in process.

2. Goes to the relevant page.

3.She/he prepares and saves the report on the car expert done to the vehicle that came to the expert service.

4.Use case ends.

ALTERNATE FLOW:

3A Invalid inputs

1. Take relevant error.

2.Use case ends.

3.MAKE A QUERY FOR LICENSE PLATE

BASIC FLOW:

1.The user enters the relevant query field and determines the query type.

2. It appears on the screen where it will enter the vehicle license plate to be inquired and enters the information.

3. Relevant data comes from the database.

4.Use case ends.

ALTERNATE FLOW:

2A Invalid inputs

1. Take relevant error.

2.Use case ends.

PUBLIC CLIENT

1.MAKE A QUERY FOR LICENSE PLATE

BASIC FLOW:

1.The user enters the relevant query field and determines the query type.

2. It appears on the screen where it will enter the vehicle license plate to be inquired and enters the information.

3. Relevant data comes from the database.

4.Use case ends.

ALTERNATE FLOW:

2A Invalid inputs

1. Take relevant error.

2.Use case ends.

2B Time Out

1. Access to the button, which allows the user to go to the screen where she/he will query, is blocked until the time determined from the previous operation has elapsed.

2.Use case continue with first step.

BUGS

• Total number of bugs found = 3

• Total number of fixed bugs = 3

• Fixed bugs explanation,

1. After the update, the text colors of the tests written in the inputs entered by the users became white and their properties were corrected by writing black.

2. Added brief summaries of the actions performed above the buttons to make them more meaningful for users.

3. The content error that occurs when viewing the operations performed in the Admin section has been fixed.

• Total number of unfixed bugs = 0

LESSON LEARNT

In general, I have deduced that the tests are the most important step in revealing the product we are doing during the testing phase.

In the design and implementation part, I discovered that it is possible to reduce some overlooked logic and implementation errors to lower levels by using test methodologies.

By testing the program, I learned some lessons that product quality and performance can be improved.

I realized that it is more important to make more testable software by increasing the code quality and code readability during the implementation phase.

The biggest problem I faced in general was that I was a little mentally exhausted as I took on all the steps alone.

Since I don't have a teammate who can take on the places I get stuck and the tasks I can't do, it taught me to search a lot on Google and dig deeper into the articles written.

SUGGESTED ACTIONS

I have no suggestions.

TEST SUMMARY

To summarize, in this document, we aimed to test the application we have implemented with different methods.

All the actions of the users who will use some functions and applications in the application have been tested.

I tested some of the methods using JUnit, and I did the integrations and their work without encountering any big problems, as I tested the other database operations by dividing them into each part of the program and then combining them.