Addis Ababa University

Department of Computer Science

Introduction to Software Engineering

Project Title: Parking Lot Management System

Software Test Plan for Parking Lot Management System

Prepared by:

1. Natnael Mesfin UGR/8654/15
2. Mikiyas Fasil UGR/9231/15
3. Yosef Solomon UGR/7358/15

Submitted to: Ayalew Belay(PhD)

Submission date: January 2, 2025

Table of Content

List of Tables

1. Project Overview

Parking lot management system is a simple stand-alone application for managing the parking operations of a parking lot. It is an innovative solution developed to provide administrators, employee, and customer-friendly platform for managing parking spaces, vehicle entries and exits and related records.
It can be found at the git repository:
https://github.com/JosiSol/Parking-Lot-Management-System.git

The Parking lot management system is built using Java's Swing GUI framework, offering an intuitive and visually appealing interface. It incorporates Java's robust file management for data storage and retrieval. The system supports three types of users: Administrators, Employees, and Customers, each with specific roles and functionalities tailored to their needs and responsibilities.

1.1. Features and Functionalities

Admin

Admin oversees the entire system's operations and configuration. The admin ensures smooth functionality, maintain system integrity, and support business objectives by managing the following tasks:

- **Employee Management**: Admins can manage staff details by adding new employees, removing employees, and viewing records of current employees.
- **Report Generation**: Admins can create detailed reports showing important information such as revenue and parking usage to implement better decisions.

Employee

The employee acts as the on-site representative, responsible for interacting with customers and managing day–to-day parking lot operations. The employee's purpose is to assist customers and ensure efficient and orderly use of the parking facility. Key responsibilities of the employee include:

- **Vehicle Check in**: Employees can efficiently record vehicle entries into the parking lot system by entering essential details including license plate. The system verifies parking lot availability before confirming the check in to ensure an organized tracking system.
- **Vehicle Check out**: The system facilitates smooth processing of vehicle exits. Employees can calculate parking fees based on the duration of stay, finalize bills, and update the parking lot real-time status
- **Bill Management:** Employees can generate detailed parking bills by calculating charges based on the customer's duration of stay
- **Parking Lot Status Monitoring:** Employees can access a real-time overview of the parking lots availability, including occupied and vacant spaces. This helps in optimal space utilization and helps them guide customers efficiently

2. Scope of the tests

2.1. Software Package to Be Tested

The name of the project being tested is "Parking Lot Management System". The software is currently in Version 1. As of now, there have been no revisions made to the document, indicating that it is the initial version.

It is implemented using the Java File System. This is the first version as it exists on the Git repository. The documentation available includes the Github repository and the associated README file. The Github repository also includes the requirement analysis and system design documents.

The testing process is divided into **unit testing**, **integration testing**, and **system testing**. The unit testing section will be limited to verifying individual functionalities as developed by the programmers. Integration testing will validate the integration of various components of the Parking Lot Management System – such as parking slot management, ticketing, payments, and employee data record.

The following items (use cases, functional requirements, and non-functional requirements) have been identified as targets for testing. The following list outlines the scope of what will be tested, while the detailed test cases and scenarios will be determined and documented later:

System Testing

Admin Module

- Verifying Admin Login/Logout Functionality
- Verifying Add Employee Functionality
- Verifying Remove Employee Functionality
- Verifying View Employee Functionality
- Verifying Report Generation

Employee Module

- Verifying Employee Login/Logout Functionality
- Verifying Check in Functionality
- Verifying Check out Functionality
- Verifying Show Status Functionality

User Interface Testing

- Ensure seamless navigation for employees
- Ensure seamless navigation for admins

Performance Testing

- Verifying Response time for check in and check out

2.2. Documents Providing the Basis for the Planned Tests

These documents include critical materials such as the software's requirements specifications, design documentation, and use cases. They offer the necessary context and insights to ensure that testing efforts align with the software's intended functionality, user expectations, and design specifications. The following list includes essential documents, their versions, and the valuable insights they provide.

➢ **Requirement Analysis Document – Version 1**: This document describes the expected features. It serves as the foundation for the software's functionality and user interaction.

It includes:

▪ **Functional Requirements**: Defines the expected behaviour of the software, including core features such as vehicle real-time monitoring, billing, and parking lot status management.

▪ **Non-Functional Requirements**: Specifies performance, security, and usability requirements related to quality standards.

▪ **Use Cases and Scenarios**: Details how users will interact with the system. It helps to validate user expectations and guide test case creation.

➢ **System Design Document – Version 1**: This document describes the system architecture, component designs, and interaction diagrams to provide insights into the structure and behaviour of the software.

It includes:

▪ **System Architecture**: Provides a high-level overview of the software's architecture, including diagrams of how different components interact.

▪ **Component Design**: Details the structure and functionality of each component. This includes parking slot management, employee management, vehicle management, and user interface.

▪ **Data Flow Diagrams**: Illustrates how data moves within the system, helping to identify potential data-related issues.

➢ **Code Documentation – Version 1**: This includes written explanations and annotations within the code that describes its functionality, logic, and structure. It helps in unit testing and provides a clear understanding of the codebase during testing

3. Testing Environment

3.1. Testing Sites

The Parking Lot Management System will be using a Macbook Air 2020 M1 and a Windows PC to test the system. It will run using the latest Java and JDK versions for our test environment.

3.2. Required hardware and firmware configuration

The required hardware is any modern personal computer capable of running executable applications. For these tests, a Macbook Air 2020 M1 with 8GB Ram and HP EliteBook x360 1040 G6. Multiple devices are used for testing to ensure that the application works for all types of machines and to not make the application machine dependent.

3.3. Software Requirements

- Java virtual environment
- openjdk 17.0.13
- JUnit for system testing
- TestFX for UI testing
- JMH (Java Microbenchmark Harness) for Performance testing

3.4. Manpower Requirements

To test the Parking Lot Management System, the testing team consists of three members.

Table 1: Team members and their estimated time requirement

#	Name of Team Member	Time Requirement (in Hrs)
1.	Natnael Mesfin	50
2.	Mikiyas Fasil	50
3.	Yosef Solomon	50

3.5. Preparation Requirements

The testing team needs to be familiar with the testing environment in order to ensure the success of the testing process. The testing team needs to be familiar with Java and its Swing GUI components while also being able to understand the Java file management system. The team should be familiar with the testing module.

The team should also review project documentation to further understand the project architecture, functionality and workflow.

4. Test Details

4.1. System Testing

Testing of the Parking Lot Management System should focus on requirements that can be directly traced to use cases. The primary goal of these tests is to verify the proper acceptance, processing, and retrieval of data as well as the correct implementation of the project. Testing will rely on black box techniques, which means the system's internal processes will be validated by interacting with the application through the GUI and analyzing the resulting output.

Below is an outline of the recommended testing for the Parking Lot Management System:

Admin Module

The admin module testing ensures the proper functionality of all CRUD operations for employees, including adding, removing, and viewing employees, as well as generating reports. The objective is to verify that the module performs as expected and that the dashboard operates seamlessly. Testing involves validating data management, ensuring the system is operational, and recording successful outcomes for all operation

Table 2: Admin Module Testing

Test Identification	T001
Test Objective	Ensure the admin module is working properly. Verify accurate data input, processing, and retrieval Verify generation of accurate reports Confirm access to real-time monitoring of parking operations
Cross-reference to relevant design and requirement analysis document	Reference the system's admin features from requirements analysis and system design document for CRUD operations of employee management and report generation
Test Class	Functional Testing
Test Level	System Test (integrates employee management and report generation functionalities)
Test Case Requirements	Admin login, employee details, date range for report. The system is up and running.
Special Requirements	Ensure security for admin privileges, including access control. Time measurement for report generation
Data to be Recorded	Successful addition of employee to the system Successful update of the employee information in the system Successful removal of employee from the system

Expected Results	All operations are executed perfectly when valid data is used

Employee Module

The Employee Module testing ensures the functionality of operations such as check-in, check-out, and viewing status. The objective is to validate that employees can perform these actions accurately and that the system records and processes the data correctly. Testing focuses on verifying proper data handling and ensuring seamless performance under normal conditions.

Table 3: Employee Module Testing

Test Identification	T002
Test Objective	Ensure the employee module is working properly Validate updates to slot statuses Verify real-time updates to slot availability reflected in the system Test ticket validation at check-in and check out
Cross-reference to relevant design and requirement analysis document	Reference the employee management operations from the requirement analysis and system design document.
Test Class	Functional Test
Test Level	System Test (all check in and checkout functionalities are tested together)
Test Case Requirements	Employee account is created and the system is up and running
Special Requirements	Ensure secure login and check-in/check-out processes. Response time measurement for check-in/check-out actions.

Data to be Recorded	Successful validation of ticket with corresponding parking details Successful update of parking slot status in real-time Correctly reflected changes in the database and dashboard Successful procedure of payment and ticket validation at check out Successful login and log out from the system Successful update of employee information (password)
Expected Results	Expected results occur when valid data is used

4.2. User Interface Testing

The testing of user interfaces focuses on validating their functionality, usability, and responsiveness to ensure efficient operations for users. The Employee Dashboard is tested to confirm it provides accurate information about parking availability and supports check-in/check-out actions, while the Admin Dashboard is evaluated for managing employees and generating parking reports effectively. Both tests emphasize verifying proper data handling, intuitive navigation, and compatibility across devices to ensure the system meets the intended design and requirements.

Table 4: User Interface Testing for Admin

Test Identifications	T003
Test Objective	Ensure that the employee dashboard functions correctly, displaying available parking spots, check-in/check-out options, and accurate information.
Cross-reference to Relevant Design and	Requirements: Employees should have an intuitive and accessible dashboard for check-in/check-out operations.

Requirement Analysis Document	Design: Dashboard layout for employee operations.
Test Class	User Interface Test
Test Level	System Test
Test Case Requirements	All the users are created and the system is up and running
Special Requirements	UI should be responsive on various devices
Data to be Recorded	Verify smooth navigation across the user interface
Expected Results	Navigation is smooth, and aligns with system design standards

Table 5: User Interface Testing for Employee

Test Identification	T004
Test Objective	Ensure the admin dashboard is fully functional for managing employees and generating parking reports.
Cross-reference to Relevant Design and Requirement Analysis Document	Requirements: Admin should be able to manage employees, view employee status, and generate reports.
	Design: Admin dashboard layout and functionality
Test Class	User Interface Test
Test Level	System Test
Test Case Requirements	Admin login, employee management tools, report generation feature.
Special Requirements	Ensure clarity in design and ease of navigation for all admin tasks.
Data to be Recorded	Admin interactions, time taken for tasks, and any UI issues.

4.3. Performance Testing

Performance testing focuses on evaluating the system's responsiveness, stability, and scalability under varying levels of load and usage. The goal is to ensure that the application can handle expected traffic, process transactions efficiently, and perform optimally under real- world conditions. By simulating different usage scenarios, performance testing ensures that the system maintains high availability and delivers a reliable user experience even under stress.

Table 6: Performance Testing

Test Identification	T005
Test Objective	Validate system response time for key transactions under normal and worst case loads
Cross-reference to Relevant Design and Requirement Analysis Document	Reference the system's performance and scalability requirements in the requirement analysis document.
Test Class	Performance Test
Test Level	System Testing
Test Case Requirements	System should simulate multiple users performing actions like check-in/check-out and report generation.
Special Requirements	Measure system response times and transaction rates under peak load conditions. Ensure system remains stable under heavy load.
Data to be recorded	System response time during increased load
Expected Results	The system maintains good response time and performance under normal and worst case loads

5. Test Schedules

The test schedule is created for preparing and executing tests for the Parking Lot Management System. The test team will prepare and perform the tests, and the development team will handle any error encountered during the testing process.

5.1. Preparation Phase

In this phase, the test team begins by defining clear objectives for testing the parking lot management system. They identify test cases covering core functionalities such as check in/checkout management, and parking slot allocation. Roles are assigned, test scripts are prepared, and necessary test data is gathered. The testing environment is configured, including hardware and software infrastructure.

Start Date: December 26, 2024 Duration: 2 days

5.2. Testing Phase

During this phase, the test team systematically evaluates all features of the parking lot management system. Test covers all key scenarios, and results are logged accurately. Bugs and Errors are documented with detailed descriptions and categorized by severity.

Start Date: December 28, 2024 Duration: 1 day

5.3. Error Correction Phase

In this phase, the development team implements fixes for the earlier identified issues. They resolve problems without introducing new ones and fix patches are deployed to the test environment for validation. The test team retests these scenarios to confirm successful resolution. Clear communication between the test and development team is vital.

Start Date: December 29, 2024 Duration: 2 days

5.4. Regression Testing Phase

The focus during this phase is on ensuring that fixes, patches and updates have not disrupted other functionalities of the parking lot management system. Test cases previously affected by changes are re-executed, to evaluate the overall system stability. Key components are verified and made sure they are up-to quality standards. The results are documented in a report for final review

Start Date: January 1, 2025 Duration: 1 day

Table 7: Test Schedule

Phase	Starting Date	Time Estimation (in days)	Details
Preparation	December 26, 2024	2	Setting up test environment and ensuring all components are ready
Testing	December 28, 2024	1	Executing system functionality tests, UI interaction tests, and performance tests under load.
Error Correction	December 29, 2024	2	Resolving any issues identified during testing, including bugs in functionality and UI performance.
Regression Tests	January 1, 2025	1	Re-test to ensure fixes don't introduce new problems and that the system functions as expected.

6. Deliverables

Throughout the testing process, the following key documents will be produced:

- Test Plan Documentation: This document will provide an overview of the testing methodology, the scope, and detailed test cases to ensure all elements of the system are thoroughly evaluated.
- Test Report: A comprehensive report summarizing the outcomes of all testing phases, including issues encountered, steps taken to resolve them, and confirmation that all specified requirements are successfully met.

7. Roles and Responsibilities

This section outlines the key roles and responsibilities involved in the testing process for the project. The table below provides a detailed overview of the responsibilities assigned to each role:

Table 8: Role and Responsibilities of Team Members

Roles	Name of Team Members	Responsibilities
Test Manager	Natnael Mesfin	Provides management oversight Allocates resources, ensures appropriate time management and resolves high-level issues
Test Designers	Mikiyas Fasil Yosef Solomon	Designs the test cases and evaluates the result of each test case while making sure it aligns with the functional and non-functional requirements
System Testers	Yosef Solomon Natnael Mesfin Mikiyas Fasil	Executes the tests. Logs results, finds document issues, and informs development team to find solutions

8. References

Bruegge, Bernd, and Allen H. Dutoit. *Object-Oriented Software Engineering*: *Using UML*, Patterns, *and Java*. 3rd ed., Pearson, 2010.

"Test Plan Template." GeeksforGeeks, https://www.geeksforgeeks.org/test-plan-template/#one page-test-plan-template.

Oracle. *The Java™* Tutorials. Oracle, https://docs.oracle.com/javase/tutorial/.

M. Mekonnen and S. Girma, *Software Test Plan and Report for Student Management System*. Addis Ababa University, College of Natural and Computational Sciences, Department of Computer Science, June 2022.

Addis Ababa University

Department of Computer Science

Introduction to Software Engineering

Project Title: Parking Lot Management System

Software Test Description for Parking Lot Management System

Prepared by:

1. Natnael Mesfin UGR/8654/15
2. Mikiyas Fasil UGR/9231/15
3. Yosef Solomon UGR/7358/15

Submitted to: Ayalew Belay(PhD)

Submission date: January 2, 2025

Table of Contents

List of Tables

1. Scope of the Tests

The software to be tested is the Parking Lot Management System, implemented using the Java File System. This is the first revision as it exists on the Git repository. The documentation available includes the Github repository and the associated README file. The Github repository also includes the requirement analysis and system design documents.

The testing process is divided into **unit testing**, **integration testing**, and **system testing.** The unit testing shall be limited to verifying functionalities developed by the development team. The integration testing shall test the components of the parking lot management system and make them work seamlessly together.

2. Testing Environment

The tests to be done have been described in the Software Test Plan for **Parking Lot Management System (Version 01)** document and further description of the test cases will be done in the **Test Cases** Section of this document.

The system that is being tested is a stand-alone application, it only needs a computer capable of running executable files. The operating systems and the hardware that is going to be used for the tests are:

Hardware

- Device 1: Macbook Air
    - Processor: Apple M1 Chip 2020
    - Memory: 16 GB RAM
    - Storage: 512GB SSD
    - Operating System: macOS Sequoia version 15.2.0
- Device 2: HP EliteBook x360 1040 G6
    - Processor: Intel(R) Core(™) i7-8665U CPU @ 1.90 GHz
    - Memory: 16 GB RAM
    - Storage: 1 TB SSD
    - Operating System: Windows 11 Pro version 23H2

Software

- Backend: Java
- Frontend: Java Swing GUI
- Database: Java File Management System
- Testing Tools: JUnit for system testing, TestFX for UI testing, JMH (Java Microbenchmark Harness) for Performance testing

3. Testing Process

The main principles to consider when testing the application are the techniques used and the criteria for knowing when to complete the test and conclude the log. Testing should only be executed using known and controlled datasets and databases while in a secured testing environment. The test strategy ensures the requirements listed in the Scope of the tests section of the Parking Lot Management System Test Plan document are substantially verified.

3.1. System Testing

System testing verifies whether data is accepted, processed, and retrieved while also checking the appropriate implementation of the business policies.

Testing will rely on black box techniques, which means the system's internal processes will be validated by interacting with the application through the GUI and analyzing the resulting output. Execute each use case, use case flow, or function with valid and invalid data to verify:

- Expected results only occur when valid data is entered
- Appropriate error/ warning messages are displayed when invalid data is used
- Each business rule is properly applied

3.2. User Interface Testing

User Interface testing verifies that the application's navigation and components align with platform functionality and adhere to usability standards.

The following approach will be used:

- Create / Modify tests for each window to verify proper navigation and behavior, including button functionality, layout responsiveness, and content visibility
- Verify the coercion of menus, forms, data, and elements across all windows.

3.3. Performance Testing

Performance Testing ensures the platform meets response time, transaction rate and other time requirements. This test is used to validate the performance requirements that have been achieved. These tests will be executed numerous times with increasing data load to measure response times and failure rates. The following steps will be taken:

- Gradually increase the total number of virtual users to increase number of transactions
- Record system response and failure rates due to the increase of workload on the system
- Run benchmark tests to establish the reference performance
- Repeat tests with increasing load to test system scalability and stability

Performance tests will be performed on dedicated machines on a dedicated time frame to ensure accurate measurements.

4. Test Cases

4.1. System Testing

Admin Module Testing

Table 1: Test Cases for Admin Module Test

Test Case	TCID	Description	Input Data	Expected Result
Login/ Logout	TC01	Admin tries to login and logout of system	-Correct credentials -Incorrect credentials -Click the logout button	-Login successful with valid credentials -"Invalid username or password" error message displayed for invalid input -Logout successful when logout button is clicked
Add Employee	TC02	Admin tries to add an employee into the system	-Correct Employee Information -Incorrect Employee Information (Invalid Fields include numbers and punctuation marks, leaving fields empty) -	- Employee user should be created successfully with correct information -"All fields are required" error message displayed if one or more fields are left empty -If password length is not equal to or greater than the minimum or if the password does not fulfill UpperCase, lowercase and number criteria, "Password must contain at least 8 characters with a mix of uppercase, lowercase, and a number." error message is displayed for invalid input
Remove Employee	TC03	Admin tries to remove and employee from the system	-Inputs Correct Username -Inputs Incorrect Username	-Successfully removes the employee with the given username from the system -"Employee with username 'incorrect username' not found" error message will be displayed for invalid input -"Username is required" error message is displayed if username field is empty
Show Employee	TC04	Admin accesses the database to view profiles of employees	-Click "Show Employee"	-Each Employee profile is displayed for the admin in the order in which they were added to the system -If no employee is there, "no employees found" message is displayed.

Status	TC05	Admin views the status of the parking lot	-Empty Parking Lot -Non-Empty Parking Lot	-”no parking data available” error message is displayed followed by an empty sheet where the parking data of customers should be -Parking data (Name, Phone Number, Plate Number, Parking Spot, Check in time) of each customer is displayed in chronological order. Also, the amount of free and taken spots is displayed at the bottom right of the screen.
Report	TC06	Admin views the daily report of the parking lot	-Empty report -Non-Empty report	-”no report data found” error message is displayed followed by an empty sheet where the daily report should be -Parking data (Name, Phone Number, Plate Number, Parking Spot, Check in time, and bill calculated) of each customer is displayed in chronological order. Also the total money earned over the day is calculated and displayed at the bottom of the screen.

Employee Module Testing

Table 2: Test Cases for Employee Module Test

Test Case	TCID	Description	Input Data	Expected Result
Login/Logout	TC07	Employee tries to login/logout of the system	-Correct credentials -Incorrect credentials -Click the logout button	-Login successful with valid credentials -”Invalid username or password” error message displayed for invalid input -Logout successful when logout button is clicked -If password length is not equal to or greater than the minimum or if the password does not fulfill UpperCase, lowercase and number criteria, “Password must contain at least 8 characters with a mix of uppercase, lowercase, and a number.” error message is displayed for invalid input

Check In	TC08	Employee tries to check in a customer to their parking slot	-Correct Customer Information -Incorrect Customer Information (Invalid Fields include numbers and punctuation marks, leaving fields empty) -If parking slot is full	-Customer Parking Slot should be created successfully with correct information and the employee will direct the customer to their assigned slot -”All fields are required” error message displayed if one or more fields are left empty - “This plate is already in use. Please enter a unique plate number” error message is displayed if the same plate number tries to check in more than once at the same time
Check Out	TC09	Employee checks out a customer	-Correct Plate Number -Incorrect Plate Number (plate number not checked in, empty field)	-Employee shows the customer the generated receipt and prompts the customer to pay. After successful payment, the employee then checks out the customer and removes them from the database while updating the report. -”Car with this plate number not found” error message is displayed if incorrect plate number is imputed -”Please enter a plate number” error message is displayed if field is empty
Status	TC10	Employee views the status of the parking lot	-Empty Parking Lot -Non-Empty Parking Lot	-”no parking data available” error message is displayed followed by an empty sheet where the parking data of customers should be -Parking data (Name, Phone Number, Plate Number, Parking Spot, Check in time) of each customer is displayed in chronological order. Also, the amount of free and taken spots is displayed at the bottom right of the screen.
Update Password	TC11	Employee changes their password	-Correct old password -Incorrect old password (empty field)	-Successful update of employee password -”All fields are required” error message is displayed for empty field -”Old password is incorrect” error message is shown for incorrect old password -”new password and confirm password do not match” error message is

				displayed for inconsistent password between the two fields -If any of the three password lengths is not equal to or greater than the minimum or if the password does not fulfill UpperCase, lowercase and number criteria, “Password must contain at least 8 characters with a mix of uppercase, lowercase, and a number.” error message is displayed for invalid input

4.2. User Interface Testing

The testing of user interfaces focuses on validating their functionality, usability, and responsiveness to ensure efficient operations for users.

Table 3: Test Cases For User Interface Test

Test Case	TCID	Description	Expected Result
Ease of Navigation	TC12	The user test navigation through all windows and sections of the system	Users should navigate through the app seamlessly and locate sections of the app with ease
Missing Functionality	TC13	Verify that all documented functionalities are implemented as described	Any missing functionalities should be documented and reported as defects so that they can be fixed by the development team

4.3. Performance Testing

It aims to verify system response times under various workloads to ensure it meets performance requirements.

Table 4: Test Cases for Performance Test

Test Case	TCID	Description	Input Data	Expected Results
Response Time	TC14	Measure the time taken for the system to process and respond to a request under intense workload	-Input the number of concurrent users -Input endpoints to be tested	The response time should be within the acceptable limits even as the concurrent users increase

5. Actions to be taken in case of program failure/cessation

In the event of a program fails during testing, the following actions must be taken by the testers:

a. Document the failure:
   i. Record the use case being tested at the time of failure
   ii. Note the input data and any error message encountered by the program
b. Attempt with a different test case:
   i. Run the program again using the different test case
   ii. If new test case does not fail, return to the original test case to replicate the failure
c. Confirm repeatability:
   i. If failure happens under similar conditions, record it as a repeatable failure.
   ii. If failure does not reoccur under similar conditions, classify it as non-repeatable failure and investigate the possibility of it being caused by external factors
d. Record observations:
   i. Document all observations, repeatability and conditions under which failure occurred
   ii. Provide detailed explanation to the development team for further analysis

6. References

Bruegge, Bernd, and Allen H. Dutoit. *Object-Oriented Software Engineering*: *Using UML*, Patterns, *and Java*. 3rd ed., Pearson, 2010.

"Test Plan Template." GeeksforGeeks, https://www.geeksforgeeks.org/test-plan-template/#one page-test-plan-template.

Oracle. *The Java™* Tutorials. Oracle, https://docs.oracle.com/javase/tutorial/.

M. Mekonnen and S. Girma, *Software Test Plan and Report for Student Management System*. Addis Ababa University, College of Natural and Computational Sciences, Department of Computer Science, June 2022.

Addis Ababa University

Department of Computer Science

Introduction to Software Engineering

Project Title: Parking Lot Management System

Software Test Report for Parking Lot Management System

Prepared by:

1. Natnael Mesfin UGR/8654/15
2. Mikiyas Fasil UGR/9231/15
3. Yosef Solomon UGR/7358/15

Submitted to: Ayalew Belay(PhD)

Submission date: January 2, 2025

Table of Contents

List of Tables

1. Test Identification, Site, Schedule, and Participation

This is the test report document for the Parking Lot Management System. It provides a detailed result of the test performed using the Parking Lot Management System Test Plan document. This is the first version, and its corresponding document used as the basis for the test report is Parking Lot Management System Test Plan document version 01.

The tests were conducted on a personal computer. Testing was not a live deployment but conducted on the devices of the testers. The devices used for testing are Macbook Air 2020 M1 and HP EliteBook x360 1040 G6. The application runs using Java and its SWING GUI.

Table 1: Team Member with their roles

Role	Team Member
Test Manager	Natnael Mesfin
Test Designers	Mikiyas Fasil Yosef Solomon
System Testers	Natnael Mesfin Mikiyas Fasil Yosef Solomon

2. Test Environment

The test environment was set up on software and hardware configurations suitable for the most optimal functioning of the application's functionality, performance and ease of navigation.

The system that is being tested is a stand-alone application, it only needs a computer capable of running executable files. The testing team used Macbook Air 2020 M1 and HP EliteBook x360 1040 G6.

Software requirements for the project are:

- Java
- JUnit for system testing

- TestFX for UI testing
- JMH (Java Microbenchmark Harness) for Performance testing

All the above software requirements were installed on the hardware of the tester machines.

3. Testing Results

3.1. System Testing

Admin Module Testing

Table 2: Admin Module Testing Result

Test Case	TCID	Executed	Passed	Priority	Number of Defects	Defect ID
Login/Logout	TC01	Yes	Yes	High	0	-
Add Employee	TC02	Yes	Yes	High	1	DEF-001
Remove Employee	TC03	Yes	Yes	Medium	0	-
Show Employee	TC04	Yes	Yes	Medium	0	-
Status	TC05	Yes	No	High	2	DEF-002, DEF-003
Report	TC06	Yes	Yes	Medium	0	-

Testing Tools Used: JUnit/ TestNG

- Failures (e.g. TCO5 - Status) estimated due to file-based data retrieval in complex scenarios.

Employee Module Testing

Table 3: Employee module testing result

Test Case	TCID	Executed	Passed	Priority	Number of Defects	Defect ID
Login/ Logout	TC07	Yes	Yes	High	0	-
Check in	TC08	Yes	Yes	High	1	DEF-001
Check out	TC09	Yes	Yes	High	1	DEF-002
Status	TC10	Yes	No	Medium	2	DEF-003, DEF-004
Update Password	TC11	Yes	Yes	Medium	0	-

3.2. User Interface Testing

Table 4: User Interface testing result

Test Case	TCID	Executed	Passed	Priority	Number	Defect ID
Ease of Navigation	TC12	Yes	Yes	Medium	0	-
Missing functionality	TC13	Yes	No	High	1	DEF-004

Testing Tools Used: TestFX

- "Missing Functionality" defect estimated due to incomplete UI elements in Admin Dashboard.

3.3. Performance Testing

Performance testing for the Parking Lot Management System was conducted using the JUnit, TestFX, and JMH tools on two endpoints: Login and Employee Dashboard. The test results recorded the average response time and number of failures as the number of concurrent users increased.

Login

Table 5: Performance testing result for login window

Test Case	TCID	Number of users	Average Response Time (in ms)	Number of failures	Defect ID
Response Time	TC14	1	120	0	-
		10	250	1	DEF-004
		25	550	2	DEF-005
		50	1100	4	DEF-006, DEF-007

Employee Dashboard

Table 6: Performance testing result for Employee dashboard window

Test Case	TCID	Number of users	Average Response Time (in ms)	Number of failures	Detect ID
Response Time	TC14	1	150	0	-
		10	300	1	DEF-008
		25	600	2	DEF-009, DEF-010
		50	1000	4	DEF-011, DEF-012

Testing Tools Used: JMH (Java Microbenchmark Harness)

4. Summary

4.1. Summary of Current Tests

Table 7: Summary of Tests

Total Tests	% of Tests passed	% of Tests failed	# of Defects
15	80%	20%	12

4.2. Summary of Defects

Table 8: Summary of Defects

Defect ID	Description	Suggestion for Correction
DEF-001	Employee was created with a weak password (single character) and with an invalid name.	Add an additional check to ensure that passwords meet a minimum length requirement and that names follow a valid format before creating a staff member.
DEF-002	Status retrieval fails under high concurrency, causing incorrect status display.	Optimize the status retrieval logic to handle high-concurrency scenarios efficiently, possibly by introducing caching or queuing mechanisms.
DEF-003	The Admin Dashboard is missing critical Ul elements affecting navigation.	Review the Admin Dashboard layout and include necessary Ul components such as navigation buttons, and labels for improved usability.
DEF-004	The system crashes when invalid employee details are entered.	Implement better validation and error handling for employee details, ensuring proper feedback is provided and errors are managed gracefully.
DEF-005	Increased response times when 25 users are logged in concurrently.	Optimize database queries and review server resources to improve performance and reduce response times under load.
DEF-006	Response times spike with 50 users logged in, leading to timeouts.	Scale server capacity or implement load balancing to distribute user traffic evenly and prevent timeouts during peak usage.

DEF-007	The system fails to handle concurrent employee dashboard interactions.	Optimize the employee dashboard to handle multiple concurrent user interactions by improving database query performance and server resource allocation.
DEF-008	Parking system fails to detect duplicate license plate numbers during check-in.	Implement a check to prevent duplicate license plate numbers from being entered in the system, ensuring proper tracking of vehicles.
DEF-009	There is a delay in syncing data after check-out, leading to status inconsistency.	Improve data synchronization by using real-time updates or periodic synchronization intervals to ensure parking lot status remains accurate.
DEF-010	Employee logout results in delayed status updates.	Implement an event-driven architecture to immediately update the status after an employee logs out, ensuring real-time consistency.
DEF-011	User authentication fails with certain username formats, causing access issues.	Implement more comprehensive username format validation and error messaging to allow proper access for all valid username formats.
DEF-012	Billing information is not being saved correctly during checkout.	Improve the checkout process to ensure billing details are properly saved

5. Special Events and Testers' Proposals

While developing the parking management system, one of the primary challenges we encountered was implementing a file-based database system to manage the data for checking in and checking out cars. Initially, we chose to store this data in a plain text file. However, we faced difficulties in efficiently fetching and displaying the current status of the parking area due to the limitations in processing unstructured text data. To resolve this, we switched to using a CSV (Comma-Separated Values) format, which provided a structured way to store the data. This change allowed the program to easily retrieve and display all necessary information, improving both functionality and user experience. The transition taught us the importance of choosing the right data format for specific use cases.

6. Reference

Bruegge, Bernd, and Allen H. Dutoit. *Object-Oriented Software Engineering*: *Using UML*, Patterns, *and Java*. 3rd ed., Pearson, 2010.

"Test Plan Template." GeeksforGeeks, https://www.geeksforgeeks.org/test-plan-template/#one page-test-plan-template.

Oracle. *The Java™* Tutorials. Oracle, https://docs.oracle.com/javase/tutorial/.

M. Mekonnen and S. Girma, *Software Test Plan and Report for Student Management System*. Addis Ababa University, College of Natural and Computational Sciences, Department of Computer Science, June 2022.