PARKING MANAGEMENT SYSTEM FOR SM MALL OF ASIA

A Testing Documentation Presented to the Faculty of Datamex College

of Saint Adeline, Inc.

In Partial Fulfillment of the Requirements for the

Degree of Bachelor of Science in Information Technology

By:

Torrazo, Kim Razel

Onde, Ben

Chua, Mallory

Panday, Sean Roscoe

August 2025

TESTING DOCUMENTATION

INTRODUCTION

The development of the Parking Management System has reached a critical juncture where the application must transition from a collection of coded features into a verified, reliable, and secure operational tool. This document outlines the testing phase, a methodical process of evaluation designed to serve as the ultimate quality gatekeeper. The purpose of this phase extends beyond simple bug detection; it is a foundational effort to build confidence in the system's core, ensuring that the software is robust and dependable enough to manage the sensitive task of user authentication and access control. Before any vehicle movements can be tracked or alerts generated, we must first prove that the system's digital front door is locked, stable, and functions exactly as specified.

The primary objective of this testing cycle is to validate the foundational security and usability of the user management workflow. This encompasses a full-circle evaluation of the user journey: from the critical first-time administrator setup, through the daily login process, to the secure recovery of a forgotten password. We will rigorously confirm that the system correctly distinguishes between authorized and unauthorized users, that all sensitive credentials are securely hashed before being committed to the database, and that the user interface provides clear and intuitive feedback under both success and failure conditions. The successful outcome of these tests will be a system that is not only functionally correct but also resilient against common user errors and predictable in its behavior.

The scope of this phase is precisely defined. Testing will be concentrated exclusively on the user authentication and account management modules (InitialSetup, Login, and ForgotPassword). We will meticulously verify every aspect of this identity layer, as it forms the bedrock upon which all future functionality will be built. Consequently, the core operational features of the system—such as the ParkingHistory and AlertHistory modules, integration with physical hardware like barrier gates or cameras, and advanced performance metrics—are explicitly out of scope for this initial cycle. These components will be subjected to their own dedicated testing phase once their development is complete.

This testing phase serves to certify the readiness of the system's user management core. The findings detailed in the subsequent sections, from test case results to bug tracking logs, will provide a transparent record of the system’s quality. By validating this foundational component, we ensure that the Parking Management System is built upon a stable, secure, and reliable platform, ready for the subsequent development of its primary operational features.

TESTING ENVIRONMENT

This section provides a detailed overview of the technical landscape—encompassing hardware, software, and data—established for the comprehensive evaluation of the Parking Management System. This well-defined environment ensures that all test outcomes are reproducible, directly attributable to the application's behavior, and free from external variables.

Hardware Specification

Device	Processor	RAM	OS	Storage
Desktop PC	Intel Core i5-9400	8 GB	Windows 10 Pro	256 GB SSD

Table 1. Hardware Specification

Software Requirements

- **Development Environment:** Microsoft Visual Studio 2010 (VB.NET)
- **Database Server:** Microsoft SQL Server 2019 Express
- **Database Admin Tool:** Microsoft SQL Server Management Studio 18 (SSMS) for database setup and query validation.
- **Frameworks:** Microsoft .NET Framework 4.0

Test Data

- Predefined user accounts with distinct roles ('Administrator' and 'Staff') to test login scenarios and role-based access.
- A set of unique security questions and answers tied to test accounts to validate the password recovery module.
- A corpus of invalid credentials, including incorrect usernames and mismatched passwords, to verify the system's error handling.
- Boundary case data, such as empty input fields and strings of maximum expected length, to ensure robust form validation.

TESTING METHODOLOGY

This section outlines the strategic approach adopted to ensure the quality, reliability, and security of the Parking Management System. It details the specific testing techniques applied, including functional (Black-Box), structural (White-Box), and user-centric (UAT) methods, to provide a comprehensive validation of the system's readiness.

Testing Process

The testing process employs a hybrid methodology, strategically blending different testing techniques to achieve exhaustive coverage. This integrated approach ensures a thorough evaluation of both the system's external behavior (what the user sees) and its internal code structure (how it works). The goal is to certify that the software is not only technically sound but also practical and effective for its intended operational environment.

Black-Box Testing

This technique focuses on testing the functionality of the system from an end-user perspective, without any knowledge of the internal code implementation. The system is treated as a "black box," where only inputs and their resulting outputs are evaluated. To verify that each feature of the system performs its specified function correctly and meets user expectations.

Application Examples

- Validating the Login Workflow: Confirming that the system grants access with valid credentials, denies access with invalid credentials, and provides clear error messaging.
- Testing the Initial Setup Form: Ensuring that a new administrator account can be created successfully by filling out the form and clicking "Confirm," and that all input validations (e.g., password mismatch) are triggered correctly.
- Verifying Form Navigation: Ensuring that the system transitions smoothly from the Initial Setup form back to the Login form, and from the Login form to the Forgot Password form.

2. White-Box Testing

Also known as structural or glass-box testing, this method involves a detailed examination of the internal code logic, data flows, and architectural pathways within the application.

Application Examples

- Analyzing the CheckForInitialSetup Logic: Inspecting the code to confirm that the conditional If userCount = 0 Then block is correctly executed, leading to the display of the setup form.
- Verifying Security Implementations: Tracing the code to ensure that the HashPassword() function is always called before a password or securityAnswer is saved to the database.
- Validating Exception Handling: Reviewing Try-Catch blocks within database connection and query execution functions to confirm that potential SQL exceptions are caught and handled gracefully without crashing the application.

3. User Acceptance Testing (UAT)

UAT is the final phase of testing, where the system is evaluated by its intended end-users in a simulated real-world context.

Execution Method

- A designated user performs the task of setting up the system for the first time by creating the primary administrator account.
- The user then logs in with the new account and attempts to use the password recovery feature.
- Feedback is gathered on the clarity of instructions, the ease of use of the interface, and the overall efficiency of the user management workflow.

Testing Tools and Frameworks

The testing process for the Parking Management System was conducted entirely through manual execution. No automated testing frameworks or tools were employed during this phase. Testers interacted directly with the application's graphical user interface (GUI) to simulate real-world user actions and validate system behavior.

This hands-on approach involved repeatedly performing all key user workflows, such as completing the initial system setup, logging in, and executing a password reset. Each feature was systematically tested by providing both valid and invalid data inputs to ensure the system's responses were correct and its error-handling mechanisms were effective. For example, valid credentials were used to confirm successful logins, while incorrect passwords and non-existent usernames were used to test the failure scenarios.

To verify the outcome of tests involving database transactions, SQL Server Management Studio (SSMS) was used as a back-end validation tool. After a test case, such as creating a new user, testers would execute SQL queries in SSMS to inspect the UserInformation table directly. This allowed for confirmation that data was being inserted correctly and that sensitive information, like passwords, was properly hashed. This combination of manual front-end testing and back-end verification ensured a thorough and accurate assessment of the system's functionality.

Testing Criteria

A formal set of criteria was established to govern the execution of test cases and the evaluation of their outcomes. This framework ensures that the assessment of the Parking Management System is objective, consistent, and aligned with project quality standards. The criteria are structured around three key areas: test case resolution, defect classification, and system readiness.

1. **Test Case Resolution: Pass/Fail Determination**

The resolution of each test case was binary and definitive. Every test was designed with a pre-written Expected Result, which described the precise state and behavior of the application under specific conditions. Upon execution, the observed Actual Result was meticulously compared against this benchmark. A test was formally marked as Pass only when the actual result was identical to the expected result in every respect. Any discrepancy, no matter how minor, resulted in the test being marked as Fail. This strict pass/fail protocol eliminates ambiguity and ensures that all deviations from the expected behavior are captured for review.

2. **Defect Management: Bug Prioritization and Severity**

Upon the failure of a test case, a defect was formally logged in the bug tracking system. To effectively manage and prioritize the resolution process, each bug was assigned a severity level based on its impact on the system's functionality and usability. The classification is as follows:

Critical: A show-stopping defect that renders a core system function entirely inoperable, with no possible workaround. An example would be a failure of the login system that blocks all user access.

High: A major functional defect that significantly impairs a key feature but does not halt all system operations. For instance, if the password reset mechanism fails, it is a high-severity issue, but the primary login function remains operational for users who remember their credentials.

Medium: A defect that causes incorrect or unexpected behavior in a non-critical feature or has a moderate impact on the user experience. An example would be the failure to clear the password field after an unsuccessful login attempt.

Low: A minor issue that does not affect functionality, such as a typographical error in a label, a UI element that is slightly misaligned, or an unclear error message.

3. **Exit Criteria: Determining Readiness for Production**

The transition of the system from the testing phase to the next stage of development (or deployment) was contingent upon meeting a strict set of exit criteria. The primary condition for readiness was the complete resolution of all documented Critical and High severity defects. While Medium and Low severity issues were not required to be fixed for phase completion, they had to be fully documented, assessed, and formally scheduled for resolution in a future release or maintenance cycle. This pragmatic approach ensures that the system is stable and functionally sound before moving forward, while allowing for minor imperfections to be addressed over time.

TEST CASES

This section provides detailed test cases designed to validate the core functionalities of the Parking Management System's user authentication and account management modules. Each test case includes a unique ID, a clear description, step-by-step actions, the expected outcome, and the actual result observed during testing. This structured format makes it easy to track system performance and identify any deviations from the expected behavior.

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
SETUP001	Successful Initial Account Creation	1. Enter all valid user details and a strong password. 2. Select a role. 3. Click 'CONFIRM'.	Account created, user redirected to Login page.	Account created, user redirected to Login page.	Pass	N/A
SETUP002	Missing Last Name	1. Leave Last Name empty. 2. Fill others valid. 3. Click 'CONFIRM'.	System says: Last Name is required.	System says: Last Name is required.	Pass	N/A
SETUP003	Missing First Name	1. Leave First Name empty. 2. Fill others valid. 3. Click 'CONFIRM'.	System says: First Name is required.	System says: First Name is required.	Pass	N/A
SETUP004	Missing Password	1. Leave Password empty. 2. Fill others valid. 3. Click 'CONFIRM'.	System says: Password is required.	System says: Password is required.	Pass	N/A
SETUP005	Missing	1. Leave Confirm	System	System	Pass	N/A

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
	Confirm Password	Password empty. 2. Fill others valid. 3. Click 'CONFIRM'.	says: Confirm Password is required.	says: Confirm Password is required.		
SETUP006	Mismatched Passwords	1. Enter different passwords in Password and Confirm Password. 2. Fill others valid. 3. Click 'CONFIRM'.	System says: Passwords do not match.	System says: Passwords do not match.	Pass	N/A
SETUP007	Missing Security Question	1. Leave Security Question empty. 2. Fill others valid. 3. Click 'CONFIRM'.	System says: Security Question is required.	System says: Security Question is required.	Pass	N/A
SETUP008	Missing Security Answer	1. Leave Security Answer empty. 2. Fill others valid. 3. Click 'CONFIRM'.	System says: Security Answer is required.	System says: Security Answer is required.	Pass	N/A
SETUP009	Unselected Role	1. Leave Role unselected. 2. Fill others valid. 3. Click 'CONFIRM'.	System says: Role is required.	System says: Role is required.	Pass	N/A
SETUP010	Account ID Field Is Not Editable	1. Observe 'Account ID' field. 2. Try to change its value.	'Account ID' field remains uneditable.	'Account ID' field remains uneditable.	Pass	N/A

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
SETUP011	Navigation to Login via 'Go to Login' Button	1. Click 'Go to Login' button.	User redirected to Login page.	User redirected to Login page.	Pass	N/A
SETUP012	Navigation to Login via 'Cancel' Button	1. Click 'Cancel' button.	User redirected to Login page.	User redirected to Login page.	Pass	N/A
LOG001	Successful Login	1. Enter a valid Account ID. 2. Enter a valid Password. 3. Click 'LOG IN'.	User successfully logs in and is directed to the main application dashboard.	User successfully logs in and is directed to the main application dashboard.	Pass	N/A
LOG002	Login with Empty Account ID	1. Leave the 'Account ID' field empty. 2. Enter a valid Password. 3. Click 'LOG IN'.	System displays a message: Account ID is required.	System displays a message: Account ID is required.	Pass	N/A
LOG003	Login with Empty Password	1. Enter a valid Account ID. 2. Leave the 'Password' field empty. 3. Click 'LOG IN'.	System displays a message: Password is required.	System displays a message: Password is required.	Pass	N/A
LOG004	Login with Empty Account ID and Password	1. Leave both 'Account ID' and 'Password' fields empty. 2. Click 'LOG IN'.	System displays messages for both Account ID and Password being required.	System displays messages for both Account ID and Password being required.	Pass	N/A

Test Case ID	**Test Description**	**Test Steps**	**Expected Output**	**Actual Output**	**Status**	**Remarks**
LOG005	Login with Invalid Account ID	1. Enter a non-existent or incorrect Account ID. 2. Enter a valid Password. 3. Click 'LOG IN'.	System displays a message: Invalid Account ID or password.	System displays a message: Invalid Account ID or password.	Pass	N/A
LOG006	Login with Invalid Password	1. Enter a valid Account ID. 2. Enter an incorrect Password. 3. Click 'LOG IN'.	System displays a message: Invalid Account ID or password.	System displays a message: Invalid Account ID or password.	Pass	N/A
LOG007	Navigation to 'Forgot Password'	1. Click on the 'Forgot Password' link.	User is redirected to the 'Forgot Password' page.	User is redirected to the 'Forgot Password' page.	Pass	N/A
FORGOT001	Successful Password Change	1. Enter a valid Account ID. 2. Enter the correct Security Question. 3. Enter the correct Security Answer. 3. Enter the correct Current Password. 5. Enter a strong New Password. 6. Enter the same New Password in Confirm New Password. 7. Click	Password successfully changed and user taken to a confirmation or login page.	Password successfully changed and user taken to a confirmation or login page.	Pass	N/A

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
		'CONFIRM'.				
FORGOT002	Missing Account ID	1. Leave Account ID empty. 2. Fill in all other required fields correctly. 3. Click 'CONFIRM'.	System says: Account ID is required.	System says: Account ID is required.	Pass	N/A
FORGOT003	Missing Security Question	1. Leave Security Question empty. 2. Fill in all other required fields correctly. 3. Click 'CONFIRM'.	System says: Security Question is required.	System says: Security Question is required.	Pass	N/A
FORGOT004	Missing Security Answer	1. Leave Security Answer empty. 2. Fill in all other required fields correctly. 3. Click 'CONFIRM'.	System says: Security Answer is required.	System says: Security Answer is required.	Pass	N/A
FORGOT005	Missing Current Password	1. Leave Current Password empty. 2. Fill in all other required fields correctly. 3. Click 'CONFIRM'.	System says: Current Password is required.	System says: Current Password is required.	Pass	N/A
FORGOT006	Missing New Password	1. Leave New Password empty. 2. Fill in all other	System says: New Password is	System says: New Password is	Pass	N/A

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
		required fields correctly. 3. Click 'CONFIRM'.	required.	required.		
FORGOT007	Missing Confirm New Password	1. Leave Confirm New Password empty. 2. Fill in all other required fields correctly. 3. Click 'CONFIRM'.	System says: Confirm New Password is required.	System says: Confirm New Password is required.	Pass	N/A
FORGOT008	New Passwords Don't Match	1. Enter a New Password. 2. Enter a *different* password in Confirm New Password. 3. Fill in all other required fields correctly. 4. Click 'CONFIRM'.	System says: New Passwords do not match.	System says: New Passwords do not match.	Pass	N/A
FORGOT009	Incorrect Security Answer	1. Enter a valid Account ID. 2. Enter correct Security Question. 3. Enter an *incorrect* Security Answer. 3. Fill in other password fields correctly.	System says: Incorrect security information.	System says: Incorrect security information.	Pass	N/A

Test Case ID	**Test Description**	**Test Steps**	**Expected Output**	**Actual Output**	**Status**	**Remarks**
		5. Click 'CONFIRM'.				
FORGOT010	Incorrect Current Password	1. Enter a valid Account ID. 2. Enter correct Security Question and Answer. 3. Enter an *incorrect* Current Password. 4. Enter valid New and Confirm Passwords. 5. Click 'CONFIRM'.	System says: Incorrect current password.	System says: Incorrect current password.	Pass	N/A
FORGOT011	New Password Not Strong Enough	1. Fill all fields correctly except for New Password. 2. Enter a weak New Password. 3. Enter the same weak password in Confirm New Password. 4. Click 'CONFIRM'.	System says: New Password needs to be stronger.	System says: New Password needs to be stronger.	Pass	N/A

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
DASH001	Display of Available Units Count	1. Navigate to the Dashboard page. 2. Observe the 'Available' units display.	The 'Available' count correctly shows the number of available units.	The 'Available' count correctly shows the number of available units.	Pass	N/A
DASH002	Display of Unavailable Units Count	1. Navigate to the Dashboard page. 2. Observe the 'Unavailable' units display.	The 'Unavailable' count correctly shows the number of unavailable units.	The 'Unavailable' count correctly shows the number of unavailable units.	Pass	N/A
DASH003	Display of Occupied Units Count	1. Navigate to the Dashboard page. 2. Observe the 'Occupied' units display.	The 'Occupied' count correctly shows the number of occupied units.	The 'Occupied' count correctly shows the number of occupied units.	Pass	N/A
DASH004	Display of Total Units Count	1. Navigate to the Dashboard page. 2. Observe the 'Total' units display.	The 'Total' count correctly shows the total number of units.	The 'Total' count correctly shows the total number of units.	Pass	N/A
DASH005	Display of Alert Notification Panel	1. Navigate to the Dashboard page. 2. Observe the 'Alert Notification' panel.	The 'Alert Notification' panel is visible and ready to display	The 'Alert Notification' panel is visible and ready to display	Pass	N/A

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
			alerts.	alerts.		
DASH006	Display of Action History Panel	1. Navigate to the Dashboard page. 2. Observe the 'Action History' panel.	The 'Action History' panel is visible and ready to display history.	The 'Action History' panel is visible and ready to display history.	Pass	N/A
SLOT001	Display of Current Area	1. Navigate to the area selection screen. 2. Observe the title at the top.	The title clearly shows "Area 1".	The title clearly shows "Area 1".	Pass	N/A
SLOT002	Click Previous Button	1. From "Area 1", click the 'PREVIOUS' button.	User is taken to the previous area (e.g., "Area 0" or an earlier area if exists).	User is taken to the previous area (e.g., "Area 0" or an earlier area if exists).	Pass	N/A
SLOT003	Previous Button disabled on first area	1. Navigate to the very first area in the sequence. 2. Observe the 'PREVIOUS' button.	The 'PREVIOUS' button is disabled.	The 'PREVIOUS' button is disabled.	Pass	N/A
SLOT004	Click Next Button	1. From "Area 1", click the 'NEXT' button.	User is taken to the next area (e.g., "Area 2").	Disabled	Fail	N/A
SLOT005	Next Button disabled on last area	1. Navigate to the very last area in the sequence. 2. Observe the 'NEXT' button.	The 'NEXT' button is disabled.	Disabled	Fail	N/A
SLOT006	Click Back	1. Click the 'Back'	User is taken back to the	User is taken back to the	Pass	N/A

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
	Button	button.	previous screen or the main dashboard.	previous screen or the main dashboard.		
PARK001	Display of Refresh Button	1. Navigate to the Park screen. 2. Observe the 'Refresh' button.	The 'Refresh' button is visible.	The 'Refresh' button is visible.	Pass	N/A
PARK002	Functionality of Refresh Button	1. Navigate to the Park screen. 2. . Click the 'Refresh' button.	The screen content reloads or updates, if any dynamic data is present.	The screen content reloads or updates, if any dynamic data is present.	Pass	N/A
PARK003	Display of Back Button	1. Navigate to the Park screen. 2. Observe the 'Back' button.	The 'Back' button is visible.	The 'Back' button is visible.	Pass	N/A
PARK004	Functionality of Back Button	1. Navigate to the Park screen. 2. Click the 'Back' button.	User is taken back to the previous screen or the main dashboard.	User is taken back to the previous screen or the main dashboard.	Pass	N/A
ALERT001	Display of Back Button	1. Navigate to the Alert screen. 2. Observe the 'Back' button.	The 'Back' button is visible.	The 'Back' button is visible.	Pass	N/A
ALERT002	Functionality of Back Button	1. Navigate to the Alert screen.	User is taken back to the previous	User is taken back to the previous	Pass	N/A

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
		2. Click the 'Back' button.	screen or the main dashboard.	screen or the main dashboard.		
ALERT003	Display of Alert Content Area	1. Navigate to the Alert screen. 2. Observe the main content area.	The main content area is visible and ready to display alerts.	The main content area is visible and ready to display alerts.	Pass	N/A
VEHIC-MODAL001	Successful Vehicle Exit	1. All fields are pre-filled based on entry (Time In, Vehicle Registration Number, Car Maker Model, Body Type, Primary Color, Secondary Color). 2. Enter a valid Time Out. 3. Enter a valid Contact Number. 4. Click 'Confirm'.	Vehicle exit recorded, modal closes, and relevant records are updated.	Vehicle exit recorded, modal closes, and relevant records are updated.	Pass	N/A
VEHIC-MODAL002	Time In field is not editable	1. Observe the 'Time In' field. 2. Attempt to input or modify its value.	The 'Time In' field remains uneditable.	The 'Time In' field remains uneditable.	Pass	N/A
VEHIC-MODAL003	Missing Time Out	1. Leave 'Time Out' empty. 2. Ensure all other required fields are filled with valid data. 3. Click 'Confirm'.	System displays a message: Time Out is required.	System displays a message: Time Out is required.	Pass	N/A
VEHIC-MODAL004	Missing Vehicle Registration Number	1. Leave 'Vehicle Registration Number' empty. 2. Ensure all other required fields are filled with valid data. 3. Click	System displays a message: Vehicle Registration Number is	System displays a message: Vehicle Registration Number is	Pass	N/A

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
		'Confirm'.	required.	required.		
VEHIC-MODAL005	Missing Contact Number	1. Leave 'Contact Number' empty. 2. Ensure all other required fields are filled with valid data. 3. Click 'Confirm'.	System displays a message: Contact Number is required.	System displays a message: Contact Number is required.	Pass	N/A
VEHIC-MODAL006	Invalid Time Out Format	1. Enter an invalid format in 'Time Out' 2. Fill other required fields valid 3. Click 'Confirm'.	System displays a message: Invalid Time Out format or future time.	System displays a message: Invalid Time Out format or future time.	Pass	N/A
VEHIC-MODAL007	Invalid Vehicle Registration Number Format	1. Enter an invalid format in 'Vehicle Registration Number'. 2. Fill other required fields valid. 3. Click 'Confirm'.	System displays a message: Invalid Vehicle Registration Number format.	System displays a message: Invalid Vehicle Registration Number format.	Pass	N/A
VEHIC-MODAL008	Invalid Contact Number Format	1. Enter an invalid format in 'Contact Number'. 2. Fill other required fields valid. 3. Click 'Confirm'.	System displays a message: Invalid Contact Number format.	System displays a message: Invalid Contact Number format.	Pass	N/A
VEHIC-MODAL009	Click Cancel button	1. Make some changes or leave fields empty. 2. Click 'Cancel'.	The modal closes without saving any vehicle exit information.	The modal closes without saving any vehicle exit information.	Pass	N/A
VEHIC-	Update Car	1. Fill all required	Vehicle exit	Vehicle exit	Pass	N/A

Test Case ID	Test Description	Test Steps	Expected Output	Actual Output	Status	Remarks
MODAL010	Maker Model	fields valid. 2. Change 'Car Maker Model' to a new valid value. 3. Click 'Confirm'.	recorded and 'Car Maker Model' updated.	recorded and 'Car Maker Model' updated.		
VEHIC-MODAL011	Update Body Type	1. Fill all required fields valid. 2. Change 'Body Type' to a new valid value. 3. Click 'Confirm'.	Vehicle exit recorded and 'Body Type' updated.	Vehicle exit recorded and 'Body Type' updated.	Pass	N/A
VEHIC-MODAL012	Update Primary Color	1. Fill all required fields valid. 2. Change 'Primary Color' to a new valid value. 3. Click 'Confirm'.	Vehicle exit recorded and 'Primary Color' updated.	Vehicle exit recorded and 'Primary Color' updated.	Pass	N/A
VEHIC-MODAL013	Update Secondary Color	1. Fill all required fields valid. 2. Change 'Secondary Color' to a new valid value. 3. Click 'Confirm'.	Vehicle exit recorded and 'Secondary Color' updated.	Vehicle exit recorded and 'Secondary Color' updated.	Pass	N/A

Table 2. Test Case Execution Log

BUG TRACKING & ISSUE LOG

This section documents all bugs, defects, or unexpected issues encountered during the testing phase of the Parking Management System. Each identified issue is carefully recorded with a unique identifier, a clear description, its assessed severity, the reporter, its current status, and the eventual resolution. This structured approach ensures efficient tracking and prioritization of all issues, facilitating a timely and effective resolution process.

Bug Severity Categories

- **Critical.** Issues that prevent core system functionalities from working and must be fixed before the next development phase
- **High.** Major issues affecting the stability or security of the authentication modules.
- **Medium.** Moderate issues that do not halt operations but impact user experience or data correctness.
- **Low.** Minor issues or cosmetic problems that do not impact core functionality .

Status Tracking

- **Open.** The issue has been reported and validated but is not yet being worked on.
- **In Progress.** A developer is actively working on a fix for the issue.
- **Resolved.** The fix has been implemented, verified by a tester, and the issue is now closed.
- **Deferred.** The issue is acknowledged as valid but will be addressed in a future release.

BUG TRACKING LOG

Table 3. Bug tracking Log

Bug ID	Description	Severity	Reported By	Status	Resolution
B001	Application crashes with an unhandled exception if the database server is offline at startup.	Critical	Tester A	Resolved	Added a global Try-Catch block within the DBConnection.GetConnection() function to gracefully handle SqlException and display a user-friendly error message.
B002	The login process is case-sensitive. If a user's username is 'admin', they cannot log in by typing 'Admin'.	Medium	Tester B	Open	The SELECT query in the Login.vb form needs to be modified to perform a case-insensitive comparison (e.g., using LOWER() or COLLATE).
B003	After a successful password reset, the ForgotPassword form remains open, forcing the user to close it manually.	Low	Tester A	Resolved	Added Me.Close() to the end of the btnResetPassword_Click event after the "Success" message box is displayed.
B004	In the InitialSetup form, the password fields do not obscure the typed characters (they are not masked).	High	Tester A	Resolved	Set the PasswordChar property on both txtPassword and txtConfirmPassword TextBoxes to '*' in the form designer.
B005	The "Cancel" button on the InitialSetup form opens the ForgotPassword form instead of	Medium	Tester A	Resolved	Corrected the logic in the btnForgotPassword_Click event on InitialSetup.vb to call Application.Exit() after user confirmation.

Bug ID	Description	Severity	Reported By	Status	Resolution
	exiting the application.				

Table 3. Bug tracking Log

USER ACCEPTANCE TESTING

User Acceptance Testing (UAT) was the final validation step, performed to ensure that the Parking Management System's authentication modules function correctly under practical usage scenarios and meet the requirements of its intended end-users, such as system administrators. The testing focused on simulating the complete user lifecycle, from initial system setup to routine access and password recovery, to evaluate overall usability, functionality, and clarity.

Test Scenarios for End-Users

1. **First-Time System Initialization**: Starting the application with an empty database to trigger and complete the Initial Setup process, thereby creating the primary administrator account for the system.
2. **Standard Login Procedure**: Closing and relaunching the application to log in using the credentials that were just created during the initial setup. This included testing both successful and intentionally failed login attempts.
3. **Password Recovery Simulation:** Pretending to have forgotten their password and using the "Forgot Password" feature to successfully reset it by providing the correct security question answer.

Feedback from Actual Users

- The user found the Initial Setup process to be straightforward and the on-screen instructions clear and easy to follow.
- The Login screen was described as clean and intuitive. The error message for an incorrect password was clear, and the automatic clearing of the password field was noted as a helpful feature.
- The Password Recovery process was functional but felt slightly disconnected. The user noted that after a successful reset, they had to manually close the window and then re-type their username on the login screen.
- A point of confusion was the consistent use of the label "Account ID", which the user felt was too technical and suggested "Username" would be more universally understood.

Necessary Improvements or Fixes

- Relabel "Account ID": Change the text of the labels on the Login, InitialSetup, and ForgotPassword forms from "Account ID" to "Username" to improve clarity for non-technical users.
- Streamline Password Reset Flow: Modify the ForgotPassword form so that after a successful password reset, the user's username is automatically passed back to the Login form's username field, saving them a step.
- Add "Show Password" Functionality: Implement a checkbox next to the password fields on the InitialSetup form to allow users to temporarily view their password, reducing the chance of typos during account creation.

CONCLUSION AND RECOMMENDATIONS

This section summarizes the results of the comprehensive testing phase for the Parking Management System's foundational modules, highlights key observations, and provides actionable recommendations for the next stage of development. It aims to certify that the user management subsystem is stable, secure, and ready for integration with the core operational features.

Overall Test Results

- The core authentication functionalities—including Initial System Setup, User Login, and Password Recovery—are performing correctly and reliably according to the specified requirements.
- All identified Critical and High severity bugs, such as the application crash on database connection failure and the unmasked password fields, have been successfully resolved and verified.
- Feedback from User Acceptance Testing (UAT) has highlighted several minor usability improvements that, while not critical, will significantly enhance the end-user experience and operational efficiency.

Key Observation and Insights

- The system demonstrates a high degree of stability and robustness in handling both valid user workflows and common error conditions, such as invalid credentials and duplicate data entry.
- The implementation of security best practices, specifically the hashing of passwords and security answers, was verified as functioning correctly, forming a strong security foundation for the application.
- Direct end-user feedback (UAT) proved invaluable for identifying practical usability issues that were not apparent during purely technical testing.
- Addressing both functional bugs and user-centric enhancements is essential for building a professional and trustworthy system.

Recommendations for Further Improvements

Based on the outcomes of this testing phase, the following actions are recommended before proceeding to the next development cycle:

1. **Resolve All Open Medium/Low Bugs:** Prioritize fixing the remaining open issues, such as the case-sensitive login bug (B002), to finalize the authentication module.
2. **Implement UAT-Driven Usability Enhancements:** Action the feedback from User Acceptance Testing by relabeling "Account ID" to "Username" across all relevant forms and streamlining the password recovery workflow.
3. **Proceed with Core Feature Development**: With the user management foundation now verified, development should commence on the primary operational modules, including the Main dashboard and the forms for managing ParkingHistory and AlertHistory.
4. **Integrate Role-Based Access Control (RBAC):** As new features are developed, ensure that they are designed to respect the accountRole ('Administrator' vs. 'Staff') stored in the user record. This will be critical for controlling access to sensitive functions.
5. **Plan the Next Testing Cycle:** A new, dedicated testing phase should be planned to cover the forthcoming parking management functionalities. This will ensure that new features are subjected to the same level of rigorous quality assurance as the foundational modules.