**Test Plan Specification**

**Pizza Connection**

Revision History 2.0

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Comments** |
| 11/21/2023 | Kanta Islam  Matthew Krol  Mehedi Zihad  William Esparza | First Draft |
| 11/28/2023 | Second Draft |
| 12/05/2023 | Final |

Table of Contents

[1. Introduction 4](#_Toc152077591)

[**1.1 Purpose** 5](#_Toc152077592)

[**1.2 References** 6](#_Toc152077593)

[**1.3 Organization** 7](#_Toc152077594)

[2. Test Objective 7](#_Toc152077595)

[**2.2 Verifying System Performance** 7](#_Toc152077596)

[**2.3 Ensuring Security and Reliability** 7](#_Toc152077597)

[3. Test Strategy 8](#_Toc152077598)

[**3.1 Test Type and Tools** 8](#_Toc152077599)

[**3.2 Testing Levels** 8](#_Toc152077600)

[4. Test Scope 9](#_Toc152077601)

[5. Test Deliverables 11](#_Toc152077602)

[**5.1 Unit Tests** 11](#_Toc152077603)

[**5.1.1 Sign Up Unit Test Cases** 12](#_Toc152077604)

[5.1.2 Login Page Unit Test Cases 13](#_Toc152077605)

[**5.1.3 Profile Page Unit Test Cases** 15](#_Toc152077606)

[**5.1.4 Order Menu Page Unit Test Cases** 16](#_Toc152077607)

[**5.1.5 Cart Page Unit Test Cases** 18](#_Toc152077608)

[**5.1.6 Manage Employee Page Unit Test Cases** 23](#_Toc152077609)

[**5.1.7 Time Clock Page Unit Test Cases** 24](#_Toc152077610)

[**5.1.8 Inventory View Page Unit Test Cases** 26](#_Toc152077611)

[**5.1.9 Inventory Add Item Form Page Test Cases** 29](#_Toc152077612)

[**5.1.10 Inventory Edit Page Unit Test Cases** 31](#_Toc152077613)

[**5.1.11 Inventory Tracking Page Unit Test Cases** 32](#_Toc152077614)

[**5.2 Integration Tests** 34](#_Toc152077615)

[5.2.1 Sign Up Integration Test Cases 34](#_Toc152077616)

[**5.2.2 Login Page Integration Test Cases** 36](#_Toc152077617)

[**5.2.3 Profile Page Integration Test Cases** 37](#_Toc152077618)

[**5.2.4 Cart Page Integration Test Cases** 38](#_Toc152077619)

[**5.2.5 Manage Employee Page Integration Test Cases** 39](#_Toc152077620)

[**5.2.6 Time Sheet Page Integration Test Cases** 42](#_Toc152077621)

[**5.2.7 Order Track Page Integration Test Case** 43](#_Toc152077622)

[**5.2.8 Inventory View Page Integration Test Cases** 45](#_Toc152077623)

[**5.2.9 Inventory Add Item Page Integration Test Cases** 45](#_Toc152077624)

[**5.2.10 Inventory Edit Page Integration Test Cases** 46](#_Toc152077625)

[5.2.11 Inventory Add Tracking Integration Test Cases 47](#_Toc152077626)

[**5.2.12 Contact Us Page Integration Test Cases** 47](#_Toc152077627)

[**5.2.13 Forgot Password Page Integration Test Cases** 49](#_Toc152077628)

[**5.2.14 Reset Password Page Integration Test Cases** 49](#_Toc152077629)

[**5.3 System Tests** 51](#_Toc152077630)

[**5.3.1 Non-Functional Test Cases** 51](#_Toc152077631)

[**5.4 Test Data** 53](#_Toc152077632)

[**5.5 Test Reports** 53](#_Toc152077633)

[**5.6 Test Schedule** 56](#_Toc152077634)

[**5.7 Test Environment** 56](#_Toc152077635)

[**5.8 Test Entry and Exit Criteria** 57](#_Toc152077636)

[**5.9 Test Pass and Fail Criteria** 57](#_Toc152077637)

[**5.10 Test Suspension and Resumption Criteria** 58](#_Toc152077638)

[**5.11 Test Design and Execution** 59](#_Toc152077639)

[**5.12 Test Data & Defect Management** 60](#_Toc152077640)

[**5.13 Risk Analysis** 61](#_Toc152077641)

[**5.14 Roles and Responsibilities** 62](#_Toc152077642)

[6. Testing Types 64](#_Toc152077643)

[**6.1 Functional Testing** 64](#_Toc152077644)

[**6.2 Non-Functional Testing** 65](#_Toc152077645)

[**6.3 Unit Testing** 66](#_Toc152077646)

[**6.4 Integration Testing** 67](#_Toc152077647)

[**6.5 System Testing** 67](#_Toc152077648)

# **1. Introduction**

Test plan documents play a crucial role in the software testing process by offering a structured framework for planning, designing, and executing tests. Among these documents, the Test Plan is particularly significant as it delineates the overarching strategy, scope, resources, test cases, and procedures for the testing phase. Functioning as a comprehensive guide, it promotes clarity and understanding among all stakeholders involved in the testing process, facilitating effective communication.

The Test Plan acts as a blueprint, providing a roadmap that guides the test team throughout the entire testing lifecycle. This document serves as a reference point, ensuring that testing activities align with the established strategy and goals. Ultimately, the Test Plan lays the foundation for successful software testing, contributing to the overall quality and reliability of the application being tested.

## **1.1 Purpose**

**Bug Prevention:** The Test Plan places a strong emphasis on early and consistent software testing to identify and rectify errors and defects proactively. This approach is designed to enhance the overall quality and functionality of the software by addressing issues before they impact end-users or the development process.

**Cost Reduction:** A primary objective of the Test Plan is to save costs by systematically averting rework, delays, and failures during the development lifecycle. Timely detection and resolution of issues optimize development resources, mitigating the risk of project failure and associated financial implications.

**Performance Improvement:** The Test Plan outlines a systematic approach to measurement and evaluation, focusing on enhancing the speed, reliability, and efficiency of the software. Testing under diverse scenarios and environments ensures compatibility and usability, ultimately improving the software's overall performance.

**Enhanced Security:** Security validation is a paramount objective of the Test Plan. By subjecting the software to rigorous testing, we aim to identify vulnerabilities and threats, fortifying the software against unauthorized access, misuse, or damage and ensuring a secure computing environment.

**Increased Customer Satisfaction:** The Test Plan underscores the alignment of the software with customer and stakeholder expectations. Through comprehensive evaluation of quality, functionality, and usability, we strive to deliver a positive user experience, ultimately fostering elevated levels of customer satisfaction.

Additionally, the Test Plan addresses key attributes of software quality, providing a structured framework for testing processes:

* **Reliability:** Ensuring consistent and failure-free software performance over an extended period is a foundational aspect of our testing strategy.
* **Scalability:** The Test Plan includes assessments to determine the software's ability to handle increased workload without compromising performance or quality.
* **Portability:** Evaluating the ease with which the software can be transferred to different hardware or software environments with minimal adaptations is a crucial component of our testing approach.
* **Reusability:** The Test Plan tests the capacity of software components to be reused for different purposes or in various contexts without significant modifications.
* **Authenticity:** Verification of conformance to original specifications and requirements, ensuring the software is free from unauthorized or counterfeit elements.
* **Practicality:** The Test Plan evaluates the provision of useful and relevant functions and features that meet real-world needs and expectations of users and stakeholders.
* **Usability:** Assessing the ease and effectiveness with which software components can be utilized by intended users to achieve their goals forms an integral part of our comprehensive testing methodology.

## **1.2 References**

1.Nazneen Ahmad, *“What Is Software Testing:”* Test plan Purpose- [What is Software Testing - A Complete Software](https://www.lambdatest.com/learning-hub/software-testing)  Testing Tutorial with examples (lambdatest.com)

2. Mit Thakkar, “*Tips to Define Test Scope for Software Testing .”* Test Scope [Tips to Define Test Scope for Software Testing (kiwiqa.com)](https://www.kiwiqa.com/tips-to-define-test-scope-for-software-testing/)

3. Sari Talin, *“Black Box Vs White Box Testing”* [Black Box Vs White Box Testing Important Differences | PractiTest](https://www.practitest.com/resource-center/article/black-box-vs-white-box-testing/#:~:text=The%20Black%20Box%20Test%20is,into%20consideration%20its%20internal%20functioning.)

4. *“System Testing Vs. Integration Testing: 6 Ways They Differ”*  [System Testing Vs. Integration Testing: 6 Key Dissimilarities - UTOR (u-tor.com)](https://u-tor.com/topic/system-vs-integration)

*5. “Difference between System Testing and Integration Testing”* [Difference between System Testing and Integration Testing - GeeksforGeeks](https://www.geeksforgeeks.org/difference-between-system-testing-and-integration-testing/)

## **1.3 Organization**

We set out as a group to have the app fully developed in three months. From the beginning of proposal of the project to the finish product. To do this we had to set clear goals for each of our team members.   
The first goal we set was to have our first prototype done by 9/26 with the features within the application of sign up, log in, forget password, and to have the main home page done. To follow this up during our second prototype we would like to have the features of employee editing, employee creation, employee deletion, menu, inventory, and the time clock. Then following this we set out to have our third prototype where we are done with all the major features. We also want to implement payment for orders and the tracking of sales. After we have done all that we fix a few bugs that we found and get ready to display the final project.

# **2. Test Objective**

**2.1 Validating Functional Requirements**

The goal of testing each feature in the web application is to ensure that every feature we have implemented works at the unit, integration and system level as laid out and intended in our design and requirements. This helps the team ensure that we deliver a product where each feature requirement has been met satisfactorily.

## **2.2 Verifying System Performance**

We want to ensure that our entire application can maintain consistency in terms of UI and logic. Our application will be tested to ensure UI elements respond as they should and that the logic for the data the application handles is consistent and validated through our system of input validation functions. We also want to ensure the entire application can handle the amount of requests/traffic within its reasonable limits without any loss of function or long loading times.

## **2.3 Ensuring Security and Reliability**

The application employs JWT for authentication and an external payment processing service for user payments. We will test that the tokens are generated and maintained for 3 days. We also will test that it is created only for that user and cannot be used by other users. We also have implemented encryption for user passwords so they cannot be deciphered from the code or element inspection or from within the database as well and will test this to ensure the account passwords are secured from bad actors that may gain access to the application’s internal systems.

# **3. Test Strategy**

## **3.1 Test Type and Tools**

The testing approach will primarily involve manual testing, where testers interact with the application's user interface to validate its functionalities. White box testing techniques will be employed. Testers will have access to the application's code and console logs to aid in debugging and investigation, but the focus will be on testing from an end-user perspective. The white-box approach will leverage code accessibility to facilitate more targeted testing, ensuring comprehensive coverage of critical components and potential edge cases. We are also using white box testing as our scope only allows for just us, the developers of the application, to test it within the time limits that we have.

While the primary testing approach is manual, certain specialized techniques or tools may be used to enhance testing efficiency or coverage. Test management tools can be utilized to track test cases, document test results, and generate reports. Browser-based testing tools can assist in cross-browser compatibility testing. Load testing tools may be employed to simulate heavy user loads and assess the application's performance under stressful conditions. In terms of our website, we will primarily use browser-based tools such as inspection and console logs. Browsers also have a built-in performance logger, and HTML/CSS advisories. These will be utilized to improve the site, but also to allow testers an amazingly straightforward way to see what is going on, and better report test passes or fails.

## **3.2 Testing Levels**

The testing effort will span multiple levels, including unit testing, integration testing, system testing, and acceptance testing. Unit testing will focus on testing individual components or functions in isolation to verify their correctness. All the unit tests are considered part of the functional testing and are located as expected in the test cases section. Integration testing will assess the interaction and compatibility between different components and external services. A lot of the functional tests are also integration based. System testing will involve end-to-end testing of the entire application to ensure its proper functioning as a whole. Acceptance testing will involve client participation to validate that the application meets their expectations and requirements. This last part will be much simpler than others, mainly being the professor/client using the website on his side and giving his stamp of approval or not.

# **4. Test Scope**

The established test scope is guided by a specific purpose, ensuring a comprehensive assessment of the functional aspects of the "Pizza Connection" web application. The primary goal is to carefully verify the functional requirements, conducting a systematic test of each specific functionality to ensure alignment with the defined requirements. This validation process guarantees that the functional components of the application meet the expected standards and provide the expected results.

A key focus of test scope is to achieve comprehensive coverage of all functional aspects through thorough testing. This involves creating and performing a broad set of functional testing cases that strictly practice different functions. The testing process systematically addresses a variety of situations and use cases, providing a robust assessment of the application's behavior with the goal of leaving any functional aspects unknown.

In addition, the test scope places significant emphasis on identifying errors or inconsistencies in the application's functional behavior and subsequent resolution. This involves an in-depth examination of the application's response to various inputs and conditions with the aim of uncovering any deviations from expected behavior. Cooperation with relevant stakeholders is integral to ensure timely and effective solutions while maintaining the integrity of operational aspects.

Effectively defining test scope includes a clear preference for features tested, focusing on frequently used features, supporting business-critical processes, being legally mandated, and tackling potentially unsafe aspects. Awareness of when to adapt to the testing opportunity is vital, recognizing that changes may be needed in the software application.

Despite the initial ambiguity of testing opportunities, a structured plan helps teams understand what needs to be done. This includes identifying testing processes such as experimental, load, and performance testing, and determining whether these processes should be performed manually or through automation tools. This structured approach helps to form better strategies for executing activities, providing stakeholders and customers with a clearer understanding of expectations. It is essential to be alert and highly aware when setting up testing opportunities to maximize the benefits of testing efforts.

Functional Scope:

The test for the Pizza Connection web application will cover all functional requirements and features for customers, employees, and owners. here is the feature and functionalities

|  |  |  |
| --- | --- | --- |
| **User Type** | **Feature & Functionality** | **Description & Requirements** |
| Customer, Employee, Owner | **Authentication:** Signup, Log In, Forgot Password, Reset Password, Contact Us | All users can sign up, log in, and utilize password-related features. User data is stored in the database according to their user type. The Contact Us feature is accessible to everyone, requiring minimal data input without the need for account creation. |
| Customer, Employee, Owner | **Pizza Ordering:** Pizza Menu, Add to Cart, Order Replacement, Payments | Users have the ability to explore the pizza menu, add items to their cart, and proceed with orders. However, order replacements and payments require a registered user account. |
| Employee, Owner | **Timekeeping:** Clock In, Clock Out, Download Pay Period Details | Both employees and owners can use the time clock functionality for attendance tracking. Owners and employees can download pay period details for effective payroll management. |
| Owner | **Business Management:** Employee Management, Sale Tracking, Inventory Management, Timesheet | Owners exclusively access features related to employee management, sale tracking, and inventory management. Owners can also edit employees' timesheets based on their working hours. |

Non-functional scope:

Our assessment will expand to non-functional aspects, evaluating performance, reliability, safety, usability, and accessibility of the Pizza Connection web application. This involves complying with standards of response time, throughput, system stability, minimum user data collection, user interface intuitiveness, and accessibility. External factors such as network problems or downtime, which are beyond our control, will not be subject to testing. Error management will be tested for network or database problems, the solution of these problems is beyond our scope.

Integration Scope:

Integration verification will focus on the seamless interaction of various components, subsystems, and external services within the web application. It is extremely important to ensure proper communication, data exchange and interoperability between different modules, web token and external APIs.

System scope:

End-to-end testing of the entire Pizza Connection web application will be conducted to verify overall functionality, performance, and user experience. Test cases, collectively covering the entire system, reject the need for direct-through testing, as most of the sites operate independently. A comprehensive review of test cases should be sufficient for evaluation.

User Acceptance Testing Scope:

Engaging users or stakeholders for User Acceptance Testing (UAT) aims to validate and obtain acceptance for the Pizza Connection web application. Feedback will be solicited on the user interface, usability, and overall satisfaction with the application. The users in this context are solely the client, who will follow the README instructions, install, and run the server and site, and provide feedback, ultimately influencing the Accept/Reject outcome.

# **5. Test Deliverables**

## **5.1 Unit Tests**

### **5.1.1 Sign Up Unit Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Sign up – No Inputs |
| **Test Case ID** | TC-1 |
| **Description** | Check if a user can sign up with no input |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. Open application. 2. Click on Account link on the sidebar. 3. Click on the Sign up button. 4. Click Sign up button. |
| **Expected Results** | 1. Error that pops up under the password field stating, “All fields must be filled!” in red |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. Error pops up correctly below the password field.   Fail:   1. Allows user to sign in without a password or email. 2. Does not show error under the password field upon clicking submit. |

|  |  |
| --- | --- |
| **Test Case Name** | Sign up – Only Email Input |
| **Test Case ID** | TC-2 |
| **Description** | Check if a user can sign up with only an email |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. Open application. 2. Click on Account link on the sidebar. 3. Click on the Sign-up button. 4. Enter “[testcase@email.com](mailto:testcase@email.com)” in the email field. 5. Click the sign-Up bottom. |
| **Expected Results** | 1. Error that pops up under the password field stating, “All fields must be filled!” in red |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. Error pops up under the password field.   Fail:  Allows the user to sign up with just an email.  Error does not pop up under the password field. |

|  |  |
| --- | --- |
| **Test Case Name** | Sign up – Only Password Input |
| **Test Case ID** | TC-3 |
| **Description** | Check if a user can sign up with only a password |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. Open application. 2. Click on Account link on the sidebar. 3. Click on the Sign-up button. 4. Enter P@ssw0rd! in the password field. 5. Click the Sign-Up button. |
| **Expected Results** | 1. Error that pops up under the password field stating, “All fields must be filled!” in red |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. Error pops up below the password field.   Fail:   1. Allows the user to sign up without an email. 2. Error does not pop up under the password field. |

### 5.1.2 Login Page Unit Test Cases

|  |  |
| --- | --- |
| **Test Case Name** | No Input Login |
| **Test Case ID** | TC-4 |
| **Description** | Leave both email and password fields blank and click on the login button. |
| **Pre-Condition** | Logged out of user account. |
| **Test Steps** | 1. Scroll down until you see the login button underneath the input fields 2. Click on Log In button |
| **Expected Results** | An error message appears between the password input field and the login button. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:   1. Error message appears under the password field   Fail:   1. User is allowed to login with no information entered. 2. Error message does not appear under the password field. |

|  |  |
| --- | --- |
| **Test Case Name** | Partially Filled Form Login |
| **Test Case ID** | TC-5 |
| **Description** | Either only input password or email and click on login button. |
| **Pre-Condition** | Logged out of user account. |
| **Test Steps** | 1. Type in either password or email in the respective field. 2. Click on the Log In button. |
| **Expected Results** | An error message appears between password field and login button asking user to input all information. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:   1. An error message appears below the password field.   Fail:   1. User is allowed to log in without an email or password. 2. No error message appears below the password field. |

|  |  |
| --- | --- |
| **Test Case Name** | Forgot Password Link |
| **Test Case ID** | TC-6 |
| **Description** | Click on the forgot password link (highlighted in blue) |
| **Pre-Condition** | User is not logged in. |
| **Test Steps** | 1. Click on forgot password link underneath the input fields |
| **Expected Results** | User is redirected to the forgot password page |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. User is redirected.   Fail:   1. User is not redirected |

|  |  |
| --- | --- |
| **Test Case Name** | Signup Button |
| **Test Case ID** | TC-7 |
| **Description** | User clicks on the button to sign up. |
| **Pre-Condition** | User is not logged in. |
| **Test Steps** | 1. Scroll all the way down until you see the signup button. 2. Click on the signup button. |
| **Expected Results** | User is redirected to the signup page. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:   1. User is redirected to the sign up page.   Fail:   1. User is not redirected to the sign up page. |

### **5.1.3 Profile Page Unit Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Profile – Invalid Phone |
| **Test Case ID** | TC-8 |
| **Description** | Checks to see if a user can enter invalid phone number |
| **Pre-Condition** | User is logged in |
| **Test Steps** | 1. Follow TC-14 2. Click pencil button next to phone number field 3. Enter “abc” into phone number field 4. Click Save |
| **Expected Results** | 1. No input should be in the field 2. Save button does not disappear |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. No information is entered into the phone number field. 2. The save button is still in sight, and the update profile button is greyed out   Fail:   1. Allows the user to save letters for a phone number. 2. Allows the user to enter no phone number. |

### **5.1.4 Order Menu Page Unit Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Order Menu – Add Pizza Item to Cart |
| **Test Case ID** | TC-9 |
| **Description** | Allow the user to add a pizza item to the cart |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. Click on Pizza Menu on the sidebar 2. Hover over the BBQ Chicken pizza 3. Click the Customize button 4. Click the Add to Cart button |
| **Expected Results** | 1. A popup at the top of the page in green with the saying, “Item added to the cart successfully!” 2. Navigate to the order menu page |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. A successful popup at the top of the page. 2. Item is added to the cart. 3. Redirected back to the menu page.   Fail:   1. No popup message at the top of the page. 2. Item is not added to the cart. |

|  |  |
| --- | --- |
| **Test Case Name** | Order Menu – Add Salad Item to Cart |
| **Test Case ID** | TC-10 |
| **Description** | Allow the user to add a salad item to the cart |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. Click on Pizza Menu on the sidebar 2. Hover over the Antipasto Salad item 3. Click the Customize button 4. Click the Add to Cart button |
| **Expected Results** | 1. A popup at the top of the page in green with the saying, “Item added to the cart successfully!” 2. Navigate to the order menu page |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. A successful popup at the top of the page. 2. Item is added to the cart. 3. Redirected back to the menu page.   Fail:   1. No popup message at the top of the page. 2. Item is not added to the cart. |

|  |  |
| --- | --- |
| **Test Case Name** | Order Menu – Add Side Item to Cart |
| **Test Case ID** | TC-11 |
| **Description** | Allow the user to add a side item to the cart |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. Click on Pizza Menu on the sidebar 2. Hover over the Breadstick item 3. Click the Add to Cart button |
| **Expected Results** | 1. A popup at the top of the page in green with the saying, “Item added to the cart successfully!” |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. A successful popup at the top of the page. 2. Item is added to the cart.   Fail:   1. No popup message at the top of the page. 2. Item is not added to the cart. |

|  |  |
| --- | --- |
| **Test Case Name** | Order Menu – Add Drink Item to Cart |
| **Test Case ID** | TC-12 |
| **Description** | Allow the user to add a drink item to the cart |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. Click on Pizza Menu on the sidebar 2. Hover over the Pepsi item 3. Click the Add to Cart button |
| **Expected Results** | 1. A popup at the top of the page in green with the saying, “Item added to the cart successfully!” |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. A successful popup at the top of the page. 2. Item is added to the cart.   Fail:   1. No popup message at the top of the page. 2. Item is not added to the cart. |

### **5.1.5 Cart Page Unit Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Pick-up Radio Button Selection |
| **Test Case ID** | TC-13 |
| **Description** | User selects the pickup radio button to set their order so the business knows the customer will pick-up their order. User enters name and phone info and any notes needed. |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. User clicks on the pickup radio button 2. User enters their name if no name is prefilled in the name field 3. User enters their phone number if no phone is prefilled in the phone number filled 4. User may enter any notes as necessary or leave the section blank |
| **Expected Results** | First, Last and Phone number fields appear and notes section as well. This will be saved as part of the order information once the order is placed successfully. |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. Entry for “first name,” “last name,” and “phone” will display   Fail:   1. No text field entries will display. 2. More or not enough text field entries are displayed. |

|  |  |
| --- | --- |
| **Test Case Name** | Delivery Radio Button Selection |
| **Test Case ID** | TC-14 |
| **Description** | User selects the deliver radio button to set their order, so the business knows to deliver the order to their customer. User enters name, address and phone info and any notes needed. |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. User clicks on the delivery radio button 2. User enters their name if no name is prefilled in the name field 3. User enters their phone number if no phone is prefilled in the phone number filled 4. User may enter any notes as necessary or leave the section blank |
| **Expected Results** | Name, address, and phone fields will be available for the user to edit/add to as needed or may be prefilled as well if they filled out their profile information. The user can then add any notes as necessary and the order information will be updated with the delivery option. |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. Entry for “first name,” “last name,” “phone,” and “address” will display   Fail:   1. No text field entries will display. 2. More or not enough text field entries are displayed. |

|  |  |
| --- | --- |
| **Test Case Name** | Order Item Edit Button |
| **Test Case ID** | TC-15 |
| **Description** | User clicks on the edit button to trigger the fields to be edited to appear on the right-hand side. |
| **Pre-Condition** | There must be at least one item added to the cart. |
| **Test Steps** | 1. User clicks on the edit button for the desired item to edit 2. User selects quantity 3. User selects size (If applicable) 4. User selects crust or dressing (varies depending on the type of item being edited) |
| **Expected Results** | Input fields appear on the right-hand side on top of where the subtotal and total are shown |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. Entries to edit the quantity, size, crust, or dressing will appear on the right side of the screen.   Fail:   1. No entries will be displayed on the right side of the screen. 2. More entries will be displayed than necessary for the item. |

|  |  |
| --- | --- |
| **Test Case Name** | Order Item Edit Input Fields |
| **Test Case ID** | TC-16 |
| **Description** | The user can input or use dropdowns to select an input for the editing of an order. |
| **Pre-Condition** | There must be at least one item added to the cart and the edit button must have been pressed. |
| **Test Steps** | 1. User selects quantity. 2. User selects size (If applicable) 3. User selects crust or dressing (varies depending on the type of item being edited) |
| **Expected Results** | Input fields are populated with the user’s selection |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. User can select quantity between 1-20. 2. User can select a different size for pizza or salad 3. User can select a different crust or dressing   Fail:   1. User cannot select a different quantity. 2. User cannot select a different size for pizza or salad. 3. User cannot select a different crust or dressing. |

|  |  |
| --- | --- |
| **Test Case Name** | Order Item Editing Save Button |
| **Test Case ID** | TC-17 |
| **Description** | User clicks on the save button |
| **Pre-Condition** | There must be at least one item added to the cart. And the edit button must have been clicked. |
| **Test Steps** | 1. User clicks on the save button |
| **Expected Results** | The order in the cart updates with the edits the user made for each respective field |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. The updated part of the order saves to the item selected.   Fail:   1. The button does nothing. 2. The item is not updated with the latest information. |

|  |  |
| --- | --- |
| **Test Case Name** | Order Item Editing Cancel Button |
| **Test Case ID** | TC-18 |
| **Description** | User clicks on the cancel button to exit the item editing state |
| **Pre-Condition** | There must be at least one item added to the cart and the user should have clicked the edit button to be in the item edit state. |
| **Test Steps** | 1. User clicks on the cancel button |
| **Expected Results** | Input fields are removed, and any changes made are not saved. |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. The item is not updated with the latest information.   Fail.   1. The button does nothing. 2. The button will still update the item information. |

|  |  |
| --- | --- |
| **Test Case Name** | Place Order Button with Empty Cart |
| **Test Case ID** | TC-19 |
| **Description** | User clicks on the place order button with no items in the cart. |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. User clicks on the place order button. |
| **Expected Results** | The user gets a message that they must add an item in the cart before placing an order. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User gets an error saying the need to enter an item to order.   Fail:   1. Let’s the user place the order with no items. 2. Nothing happens up clicking the button. |

|  |  |
| --- | --- |
| **Test Case Name** | Place Order Button Without Being Logged In |
| **Test Case ID** | TC-19 |
| **Description** | User clicks on the place order button without being logged in. |
| **Pre-Condition** | The user must not be logged in. |
| **Test Steps** | 1. User clicks on the place order button. |
| **Expected Results** | The user is redirected to the log in page to log in before placing an order. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User is redirected to the log in page.   Fail:   1. Allow the user to place the order without being logged in. 2. Nothing happens upon pressing the button. |

### **5.1.6 Manage Employee Page Unit Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Employee Delete |
| **Test Case ID** | TC-20 |
| **Description** | This test case is designed to verify that an admin user can successfully delete an employee from the database. It checks the functionality of the delete operation and the system's response to a successful or failed deletion. |
| **Pre-Condition** | 1. The tester is logged in as an admin user. 2. The tester is on the “Manage Employees” page. 3. There are employee records available for deletion. |
| **Test Steps** | 1. Click on the any employee firstname(e.g., Mehedi) 2. Click on the 'Delete' button. 3. Confirm the deletion when a confirmation prompt appears. |
| **Expected Results** | 1. The system should allow the admin to navigate and select an employee for deletion. 2. Upon confirmation of deletion "Are you sure to delete this employee”, the system should process the request. 3. The deleted employee record should no longer be present in the database or employee list. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. The employee record is successfully deleted from the database with an appropriate success message displayed.   Fail:   1. The system fails to delete the employee record, does not display the correct message, or deletes the wrong record. |

|  |  |
| --- | --- |
| **Test Case Name** | Employee Edit Button |
| **Test Case ID** | TC-21 |
| **Description** | This test case is designed to verify that an admin user can successfully Edit Employee page after clicking Edit button on the manage employee page. |
| **Pre-Condition** | The tester is logged in as an admin user.  There are employee records available for deletion. |
| **Test Steps** | 1. Click on the any employee firstname(e.g., Kanta) 2. Click on the 'Edit' button. |
| **Expected Results** | The system should redirect to the edit employee page, where the admin can modify the employee’s details. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:  User redirects to the Edit Employee page.  Fail:   1. The Button does nothing. 2. Button redirects to the wrong page. |

|  |  |
| --- | --- |
| **Test Case Name** | Employee Time Sheet Button |
| **Test Case ID** | TC-22 |
| **Description** | This test case is designed to verify that an admin user can successfully redirect to the Timesheet page after clicking button “Time Sheet” on the “Manage Employees” page. |
| **Pre-Condition** | The tester is logged in as an admin user.  The tester is on the “Manage Employees” page. |
| **Test Steps** | Click on the any employee firstname(e.g., William)  Click on the 'Timesheet' button. |
| **Expected Results** | The system should redirect to the time sheet page, where the admin can add, edit clock in and clock out. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:  User redirects to the “Time Sheet” page.  Fail:   1. The Button does nothing. 2. Button redirects to the wrong page. |

|  |  |
| --- | --- |
| **Test Case Name** | Employee Search Bar Functionality - First Name Filter |
| **Test Case ID** | TC-23 |
| **Description** | This test case is designed to verify the functionality of the Employee Search Bar in the “Manage Employees” section, particularly focusing on its capability to filter and display employees based on their first name. |
| **Pre-Condition** | The tester must be logged in with admin privileges.  The tester is on the “Manage Employees” page where the enhanced Employee Search Bar is present. |
| **Test Steps** | Enter a specific first name (e.g., "William") into the Employee Search Bar. |
| **Expected Results** | The Employee Search Bar should accurately filter and display a list of employees whose first name matches the entered name (e.g., all employees named “William”). |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:  The Employee Search Bar successfully filters and displays employees based on the first name entered.  Fail:  The Employee Search Bar fails to correctly filter employees by first name, either not displaying the correct employees or showing an incorrect list. |

|  |  |
| --- | --- |
| **Test Case Name** | Employee Search Bar Functionality - Last Name Filter |
| **Test Case ID** | TC-24 |
| **Description** | This test case is designed to verify the functionality of the Employee Search Bar in the “Manage Employees” section, particularly focusing on its capability to filter and display employees based on their last name. |
| **Pre-Condition** | The tester must be logged in with admin privileges.  The tester is on the “Manage Employees” page where the enhanced Employee Search Bar is present. |
| **Test Steps** | Enter a specific last name (e.g., "Esparza") into the Employee Search Bar. |
| **Expected Results** | The Employee Search Bar should accurately filter and display a list of employees whose last name matches the entered name (e.g., all employees named “Esparza”). |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:  The Employee Search Bar successfully filters and displays employees based on the last name entered.  Fail:  The Employee Search Bar fails to correctly filter employees by last name, either not displaying the correct employees or showing an incorrect list. |

|  |  |
| --- | --- |
| **Test Case Name** | Filter by Role Functionality - Owner |
| **Test Case ID** | TC-25 |
| **Description** | This test case assesses the capability of the Employee Search Bar within the “Manage Employees” section to filter and display employees based on their designated role, specifically focusing on the "Owner" role. |
| **Pre-Condition** | The tester must be logged in with admin privileges.  The tester is on the “Manage Employees” page. |
| **Test Steps** | Filter owner role from the filter. |
| **Expected Results** | The Employee Filter Bar should efficiently filter and display only those employees who are designated as 'Owners'. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:  The Employee Filter Bar accurately filters and displays a list of employees who have the role of 'Owner'.  Fail:  The Employee Filter Bar fails to correctly filter employees by the 'Owner' role, either not displaying the correct employees, showing employees with different roles, or presenting an inaccurate or incomplete list. |

|  |  |
| --- | --- |
| **Test Case Name** | Filter by Role Functionality - Manager |
| **Test Case ID** | TC-26 |
| **Description** | This test case assesses the capability of the Employee Filter Bar within the “Manage Employees” section to filter and display employees based on their designated role, specifically focusing on the "Manager" role. |
| **Pre-Condition** | The tester must be logged in with admin privileges.  The tester is on the “Manage Employees” page. |
| **Test Steps** | Select “Owner” role from the filter. |
| **Expected Results** | The Employee Filter Bar should efficiently filter and display only those employees who are designated as 'Manager'. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:  The Employee Filter Bar accurately filters and displays a list of employees who have the role of 'Manager'.  Fail:  The Employee Filter Bar fails to correctly filter employees by the 'Owner' role, either not displaying the correct employees, showing employees with different roles, or presenting an inaccurate or incomplete list. |

### **5.1.7 Time Clock Page Unit Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Clock In Button |
| **Test Case ID** | TC-27 |
| **Description** | This test case will verify that an employee can successfully clock in using the application. |
| **Pre-Condition** | 1. The user is logged in as an employee, manager, or owner. 2. Follow TC-22 |
| **Test Steps** | 1. Navigate to the time clock page. 2. Click on the 'Clock In' button. 3. Observe the response. |
| **Expected Results** | After clicking 'Clock In', the user should see a pop-up message indicating a successful clock-in. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. A successful pop up displayed. 2. User was successfully clocked in.   Fail:   1. No pop up message being displayed. 2. User was not clocked in. |

|  |  |
| --- | --- |
| **Test Case Name** | Clock Out Button |
| **Test Case ID** | TC-28 |
| **Description** | This test case will verify that an employee can successfully clock out using the application. |
| **Pre-Condition** | The user is logged in as an employee, manager, or owner.  The user has clocked in and is on the application's home page. |
| **Test Steps** | 1. Navigate to the time clock page. 2. Click on the 'Clock Out' button. 3. Observe the response. |
| **Expected Results** | After clicking 'Clock Out', the user should see a pop-up message indicating a successful clock-out.  The system should record the clock-out time and date accurately. |
| **Priority** | High |
| **Pass/Fail** | Pass:   1. A successful pop up displayed. 2. User was successfully clocked out.   Fail:   1. No pop up message being displayed. 2. User was not clocked out. |

### **5.1.8 Inventory View Page Unit Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Add Item Button |
| **Test Case ID** | TC-29 |
| **Description** | Redirects to the add inventory item form page. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. |
| **Test Steps** | 1. Click on the add item button |
| **Expected Results** | User redirected to add item form page. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User is redirected to the add inventory page.   Fail:   1. Button does nothing. 2. Button redirects them to the wrong page. |

|  |  |
| --- | --- |
| **Test Case Name** | Edit Inventory Button |
| **Test Case ID** | TC-30 |
| **Description** | User clicks on this button to go to the edit inventory page. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. |
| **Test Steps** | 1. User clicks on the edit inventory button |
| **Expected Results** | User redirected to the edit inventory page. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:   1. User is redirected to edit inventory page.   Fail:   1. Button does nothing. 2. Button redirects them to the wrong page. |

|  |  |
| --- | --- |
| **Test Case Name** | Track Inventory Button |
| **Test Case ID** | TC-31 |
| **Description** | User clicks on the track inventory button to be redirected to the track inventory page. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. |
| **Test Steps** | 1. User clicks on track inventory button. |
| **Expected Results** | The user is redirected to the track inventory page. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:   1. User is redirected to track inventory page.   Fail:   1. Button does nothing. 2. Button redirects them to the wrong page. |

|  |  |
| --- | --- |
| **Test Case Name** | Inventory Item Search |
| **Test Case ID** | TC-32 |
| **Description** | User inputs the name of any item in to search up a specific item in the inventory, results come up matching the letter(s) or word input in the search bar. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. There must also be at least two items in the inventory list to see the effects of the searching feature. Additionally, the user should be in the first page within the inventory list (Page 1) |
| **Test Steps** | 1. Input any letter(s) or word that may be in the inventory list 2. List of items matching the input will appear 3. Refine the search more by typing a specific word/name of an inventory item |
| **Expected Results** | Results matching the input in the search bar will only be visible |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. User is displayed only matching inventory to search.   Fail:   1. User is displayed nonmatching inventory to search. |

|  |  |
| --- | --- |
| **Test Case Name** | Inventory Item Sort |
| **Test Case ID** | TC-33 |
| **Description** | User clicks on the arrows next to the ‘name’ field in the inventory item list header to sort the list by name (ascending or descending) |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. There must also be at least two items in the inventory list to see the effects of the sorting feature. |
| **Test Steps** | 1. Click on the arrows to sort the inventory item list in descending order. 2. Click again on the arrows to sort the inventory item list in ascending order. |
| **Expected Results** | Inventory item list sorted by ascending or descending order |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. Inventory list is sorted in ascending or descending order.   Fail:   1. Inventory list is not sorted in any order. |

|  |  |
| --- | --- |
| **Test Case Name** | Inventory Item List Pagination |
| **Test Case ID** | TC-34 |
| **Description** | When the number of items in the list exceeds 10 rows it will send additional items to the next page and will continue to do so for subsequent pages. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. There must also be at least 11 items on the inventory list. |
| **Test Steps** | 1. Click on the bottom right arrow next to ‘Page 1’ to proceed to the next page. 2. On page 2 there will be the option to go back to page 2 by clicking on the left arrow next to ‘Page 2’ on the bottom 3. If there are more than 20 items, then you will have the option to proceed to the next page (page 3) as well. 4. Any subsequent pages (in increments of 10) will become available to page through until you get to the end of the list. |
| **Expected Results** | Page # with arrows on both sides of the ‘page #’ on the bottom of the list will appear. Only the right arrow will be enabled to go to the next page. If on page 2 but list ends there only the left arrow will be enabled, otherwise both arrows will be enabled, and this applies to any subsequent page until the final page is shown where only the left arrow will be enabled. |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. User will go to a different page based on the button they selected.   Fail:   1. User will remain on the same page. 2. User will be redirected to the wrong page. |

### **5.1.9 Inventory Add Item Form Page Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Select Category Field |
| **Test Case ID** | TC-35 |
| **Description** | The list of available categories appears in the dropdown selection and the user selects one of these categories. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. |
| **Test Steps** | 1. Click on the select category input field. 2. Select a category to add to the input field. |
| **Expected Results** | The selected category populates the input field. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User is able to select a category.   Fail:   1. User is not able to select a category. 2. The wrong category (based on selection) is not loaded into the field. |

|  |  |
| --- | --- |
| **Test Case Name** | Select Unit Type |
| **Test Case ID** | TC-36 |
| **Description** | A list of unit types appears based on the category selected in the category input field, from which the user can select and apply it to the unit type field. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. TC-45. |
| **Test Steps** | 1. Click on the input field to show the options. 2. Select the unit type desired. |
| **Expected Results** | The selected unit type populates the unit type field. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. The selected unit type populates the field   Fail:   1. The selected unit type does not populate the field. 2. The wrong unit type populates the field. |

### **5.1.10 Inventory Edit Page Unit Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Inventory Edit Item Row |
| **Test Case ID** | TC-37 |
| **Description** | User can click any row to activate the input fields for that selected row. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. There must also be at least one item on the list. |
| **Test Steps** | 1. Click on any item in the list. |
| **Expected Results** | Input fields appear for the selected row. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User can click on any item in the inventory list.   Fail:   1. User cannot click on any item in the inventory list. 2. It will select the wrong item in the list. |

|  |  |
| --- | --- |
| **Test Case Name** | Inventory Edit Page Save Button |
| **Test Case ID** | TC-38 |
| **Description** | User clicks on the save button to save any edits. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. There must also be at least one item on the list. |
| **Test Steps** | 1. Make any edit in any field for an item. 2. Click on save. |
| **Expected Results** | The user can save changes on any row/item in the table. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. Save changes will display for the edited item.   Fail:   1. Changes will not be saved for the item. 2. The button will not do anything. |

|  |  |
| --- | --- |
| **Test Case Name** | Inventory Edit Page Delete Button |
| **Test Case ID** | TC-39 |
| **Description** | User clicks on the delete button to eliminate that item/row. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. There must also be at least one item on the list. |
| **Test Steps** | 1. Click on the delete button |
| **Expected Results** | The item/row is deleted from the list. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. The item in the list is deleted.   Fail:   1. The item in the list is not deleted. 2. A wrong item in the list is deleted. |

### **5.1.11 Inventory Tracking Page Unit Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Add Tracking Entry Button |
| **Test Case ID** | TC-40 |
| **Description** | User clicks on the add tracking entry button to redirect to the add tracking entry form. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. |
| **Test Steps** | 1. User clicks on the add tracking entry button |
| **Expected Results** | The user redirected to the add tracking entry form page. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User is redirected to the add tracking page.   Fail:   1. User is not redirected to the add tracking page. 2. The user is redirected to the wrong page. 3. The button does nothing. |

|  |  |
| --- | --- |
| **Test Case Name** | Inventory Tracking Inventory View Button |
| **Test Case ID** | TC-41 |
| **Description** | User clicks on inventory view button to go back to the inventory view page. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. |
| **Test Steps** | 1. Click on inventory view button |
| **Expected Results** | The user is redirected to inventory view page. |
| **Priority** | Low |
| **Pass/Fail Criteria** | Pass:   1. User is redirected to the inventory view page.   Fail:   1. User is not redirected to the inventory view page. 2. The user is redirected to the wrong page. 3. The button does nothing. |

|  |  |
| --- | --- |
| **Test Case Name** | Inventory Tracking Page Delete Button |
| **Test Case ID** | TC-42 |
| **Description** | User clicks on the delete button to eliminate that item/row. |
| **Pre-Condition** | 1. Must be logged in and have owner/manager privileges. 2. There must also be at least one item on the list. |
| **Test Steps** | 1. Click on the delete button |
| **Expected Results** | The item/row is deleted from the list. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:   1. The item in the list is deleted.   Fail:   1. The item in the list is not deleted. 2. A wrong item in the list is deleted. |

## **5.2 Integration Tests**

### **5.2.1 Sign Up Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Sign up-Access Token in Local Storage |
| **Test Case ID** | TC-43 |
| **Description** | Allow user to sign up |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. Open application. 2. Click on Account link on the sidebar. 3. Click on the Sign-up button. 4. Enter “[testcase@email.com](mailto:testcase@email.com)” in the email field. 5. Enter P@ssw0rd! in the password field. 6. Click Sign up. 7. JWT assigned a web token to the user in local storage. 8. Redirects to the home page |
| **Expected Results** | 1. Popup saying, “You have been successfully login.” 2. Redirect to the home page. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User is allowed to sign up. 2. User is redirected to the home page.   Fail:   1. An error message is displayed. 2. Nothing happens when clicking the button. |

|  |  |
| --- | --- |
| **Test Case Name** | Sign up – Email Already Used |
| **Test Case ID** | TC-44 |
| **Description** | Check if a user can sign up with a registered email |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. Open application. 2. Click on Account link on the sidebar. 3. Click on the Sign-up button. 4. Enter “[testcase@email.com](mailto:testcase@email.com)” in the email field. 5. Enter P@ssw0rd! in the password field. 6. Click Sign up. |
| **Expected Results** | 1. Error that pops up under the password field stating, “Email already in use” in red |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. Error displaying that “Email already in use”   Fail:   1. Allows the user to sign up anyway. 2. Button does not do anything. |

|  |  |
| --- | --- |
| **Test Case Name** | Sign up – Weak Password |
| **Test Case ID** | TC-45 |
| **Description** | Check if a user can sign up with a weak password |
| **Pre-Condition** | N/A |
| **Test Steps** | 1. Open application. 2. Click on Account link on the sidebar. 3. Click on the Sign-up button. 4. Enter “[testcase@email.com](mailto:testcase@email.com)” in the email field. 5. Enter password in the password field. 6. Click Sign up. |
| **Expected Results** | 1. Error that pops up under the password field stating, “Password does not meet requirements” in red |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. Error displaying that “Password does not meet requirements”   Fail:   1. Allows the user to sign up anyway. 2. Button does not do anything. |

### **5.2.2 Login Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | User Login |
| **Test Case ID** | TC-46 |
| **Description** | User logs into their account. |
| **Pre-Condition** | User is not logged in. |
| **Test Steps** | 1. Enter email in the email field 2. Enter password in the password field 3. Click/press the login button 4. User is redirected to the home page |
| **Expected Results** | User successfully logs in and is redirected to the home page. The sidebar should also populate with the options conferred to their account based on their access level. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User is allowed to log in. 2. User is redirected to the home page.   Fail:   1. An error message is displayed. 2. Nothing happens when clicking the button. |

|  |  |
| --- | --- |
| **Test Case Name** | User JWT Created |
| **Test Case ID** | TC-47 |
| **Description** | User’s JSON Web Token is created upon the start of a login session. |
| **Pre-Condition** | User is not logged in and authentication context state is null. |
| **Test Steps** | 1. Fill out the login page form 2. Click on the login button 3. Open the console using F12 4. Click on the console option (may be hidden in the >> dropdown options) |
| **Expected Results** | Once the user’s login session begins a JWT is generated and assigned to the user’s account which can be seen in the console results by checking the authentication context state set to the user’s email and displays the user’s token. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User is allowed to log in. 2. User is redirected to the home page.   Fail:   1. An error message is displayed. 2. Nothing happens when clicking the button. |

### **5.2.3 Profile Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Profile – Entering Information |
| **Test Case ID** | TC-48 |
| **Description** | Checks to see if a user can enter profile information |
| **Pre-Condition** | User is logged in |
| **Test Steps** | 1. Follow TC-13 2. Click pencil button next to first name field. 3. Enter “Matthew” into first name field. 4. Click Save 5. Click pencil button next to last name field. 6. Enter “Doe” into the last name field. 7. Click Save 8. Click the pencil button next to the phone number field. 9. Enter “5454546464” into the phone number field. 10. Click Save 11. Click the pencil button next to the address field. 12. Enter “42 W Warren Ave, Detroit, MI 48202” into address field. 13. Click Save 14. Click pencil button next to date of birth field 15. Click on the calendar 16. Navigate to December 15, 1999 17. Click Save 18. Click Update Profile button |
| **Expected Results** | 1. A popup at the top of the page stating “You have updated your profile” will appear in green at the top for a second and a half. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. All the information is updated and displayed on the page.   Fail:   1. One or more of the pieces of information is not displayed on the page. 2. The information is displayed in the wrong spot. |

### **5.2.4 Cart Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Cart Checkout |
| **Test Case ID** | TC-49 |
| **Description** | User clicks on the place order button and is redirected to the stripe payment page. |
| **Pre-Condition** | There must at least be one item in the cart and the user must be logged in. |
| **Test Steps** | * User selects either pickup or delivery option * User inputs their information (name, address and phone) * User may edit order items * User clicks on the place order button |
| **Expected Results** | The user is redirected to the stripe payment page. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User is redirected to the stripe payment page.   Fail:   1. User is not redirected to the stripe payment page. 2. The button does not do anything. |

### **5.2.5 Manage Employee Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Create Employee |
| **Test Case ID** | TC-50 |
| **Description** | This test case verifies the functionality of adding a new user to the database using the 'Employee Create' feature, ensuring that an admin user can successfully add new employees and receive appropriate confirmation. |
| **Pre-Condition** | 1. The tester has logged in as an admin user. 2. The tester is on the employee management page. 3. No employee with the test data's (email) exists in the system |
| **Test Steps** | 1. Click on the 'Add Employee' link. 2. Enter the required information for a new Employee  * Enter first name: John, last name: Wick. (required) * Select Role (Owner, Employee, Manager) (required) * Enter the address (123 main St, Detroit, MI 48212). * Enter hourly rate. (e.g., $20). * Enter unique email ([johnwick99@gmail.com](mailto:johnwick99@gmail.com) * Enter at least 8-character password with the required (e.g., Employee$911) * (ensure to test with valid and invalid data, including future dates, invalid hourly rates, and passwords).  1. Click the 'Add Employee' button. |
| **Expected Results** | 1. Upon entering valid data and submitting, a popup message saying, “Employee created successfully,” should appear. 2. The system should remain on the same employee management page after the process. 3. In the case of invalid data (future date, invalid hourly rate, existing email, and password), an appropriate error message should be displayed. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. The employee is successfully created with valid data, and an appropriate success message is displayed. 2. In the case of invalid data entry, the system should correctly display an error message.   Fail:   1. The employee is not created with valid data, no success message is displayed, or the system fails to correctly handle invalid data entries. |

|  |  |
| --- | --- |
| **Test Case Name** | Manage Employee |
| **Test Case ID** | TC-51 |
| **Description** | This test case evaluates the ability of the system to manage existing employee data. The test will cover updating employee details, employee timesheet, employee Edit, and employee delete. |
| **Pre-Condition** | 1. Follow TC-44. 2. The tester has logged in as an admin user. 3. The tester is on the employee management page. |
| **Test Steps** | 1. Navigate to the 'Employee Management' section. 2. Select an existing employee from the employee list. (e.g., Jhon Wick) 3. Available actions are Timesheet, Edit or Delete to manage employee. |
| **Expected Results** | 1. The system should allow the selection and viewing of employee details. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:   1. All buttons should work properly.   Fail:   1. If buttons are not working properly. |

|  |  |
| --- | --- |
| **Test Case Name** | Employee Edit |
| **Test Case ID** | TC-52 |
| **Description** | This test case aims to validate the functionality allowing an admin user to edit the information of an existing employee. It ensures that changes can be made accurately, and the system provides appropriate feedback upon successful update. |
| **Pre-Condition** | 1. The tester is logged in as an admin user. 2. Follow TC-44 |
| **Test Steps** | 1. Click on the 'Employee Management' button. 2. Search for the employee to be edited. Click the 'Edit' button by clicking on the employee’s first name. (Jhon) 3. Change the desired information (e.g., employee moved to new address. 4. Click on the address field and edit new address (e.g., 11345 Casmere St, Detroit, MI 4822) 5. Click on the 'Save' button. Observe the response from the system. |
| **Expected Results** | 1. The system should allow the admin to search and select an employee for editing. 2. Editable fields should be accessible and allow for changes. 3. Upon clicking 'Save', a popup should appear with the message, “Employee updated successfully,” and the system should remain on the same page. 4. The system should accurately reflect the changes made in the employee's details. 5. In case of invalid data entry, an appropriate error message should be displayed. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. The employee information is successfully updated with valid data, and the system shows a success message. 2. In cases of invalid data entry, the system correctly displays an error message.   Fail:   1. The system fails to update the employee information, does not show a success message, or does not correctly handle invalid data entries. |

### **5.2.6 Time Sheet Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Add Time Clock in |
| **Test Case ID** | TC-53 |
| **Description** | This test case verifies that an administrator can successfully add a time clock entry for an employee in the system to clock in |
| **Pre-Condition** | 1. The user is logged in as an admin, owner, or manager. 2. The administrator has access to the employee time clock management section. 3. The employee for whom the time clock entry is to be added exists in the system. |
| **Test Steps** | 1. Navigate to the employee time clock management section. 2. Select the employee for whom the time clock entry is to be added to clock in 3. Click on the 'Clock In' button. 4. Observe the response. |
| **Expected Results** | After clicking 'Clock In', the user should see a pop-up message indicating a successful clock-in. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Clock in time and date to be filled after test execution. |

|  |  |
| --- | --- |
| **Test Case Name** | Add Time Clock Out |
| **Test Case ID** | TC-54 |
| **Description** | This test case verifies that an administrator can successfully add a time clock entry for an employee in the system to clock out. |
| **Pre-Condition** | 1. The user is logged in as an admin, owner, or manager. 2. The administrator has access to the employee time clock management section. 3. The employee for whom the time clock entry is to be added exists in the system. |
| **Test Steps** | 1. Navigate to the employee management section. 2. Select the employee for whom the time clock entry is to be added to clock out 3. Click on the 'Clock out' button. 4. Observe the response. |
| **Expected Results** | After clicking 'Clock Out', the user should see a pop-up message indicating a successful clock-out. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Clock out time and date to be filled after test execution. |

### **5.2.7 Order Track Page Integration Test Case**

|  |  |
| --- | --- |
| **Test Case Name** | Order Track |
| **Test Case ID** | TC-55 |
| **Description** | This test case ensures that an admin can successfully track orders as a table format from the database after cart checkout from stripe payment gateway. |
| **Pre-Condition** | The user is logged in as an admin.  Database is available. |
| **Test Steps** | 1. Click on the Order Track page. 2. Observe the interface for the orders history table. 3. Verify the accuracy and completeness of the order history displayed. 4. Search order using email (e.g., hj1338@wayne.edu). 5. Apply filter according to the pickup and delivery method. |
| **Expected Results** | 1. The user should successfully navigate to the Order Track section, where the order history table is displayed. 2. The order history table should be user-friendly and display all necessary information, including email, order type, items, total price, date, and time. 3. After searching by email, orders associated with hj338@wayne.edu should appear. 4. When filtering by Pick Up or Delivery, only orders corresponding to the selected type (either Pick Up or Delivery) should appear." |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:   1. The user successfully navigates to the Order Track section and the order history table is displayed. 2. The order history table is user-friendly and displays all necessary information including email, order type, items, total price, date, and time. 3. When a search is conducted using an email (e.g., [hj338@wayne.edu](mailto:hj338@wayne.edu)), the system correctly displays orders associated with that email. 4. When applying the filter for Pick Up or Delivery, the system accurately displays only the orders that correspond to the selected type (either Pick Up or Delivery).   Fail:   1. Failure to navigate to the Order Track section or if the order history table does not display. 2. The order history table lacks user-friendliness or does not display all required information such as email, order type, items, total price, date, and time. 3. The search function fails to display orders associated with the specified email ([hj338@wayne.edu](mailto:hj338@wayne.edu)). 4. The filter function for Pick Up or Delivery does not accurately filter the orders, showing incorrect or mixed results. |

### **5.2.8 Inventory View Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Inventory List View |
| **Test Case ID** | TC-56 |
| **Description** | The inventory list appears if there are any inventory items existing. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. There must also be at least one item in the inventory list to view it. |
| **Test Steps** | 1. Ensure you can view the items in the inventory list |
| **Expected Results** | A list of variable amounts of inventory items appears in the table with the proper information appearing under its corresponding header. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. The inventory list is displayed on the page.   Fail:   1. No inventory is displayed on the page. |

### **5.2.9 Inventory Add Item Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Add Item to Inventory |
| **Test Case ID** | TC-57 |
| **Description** | User fills out the form and clicks on the save button to add the item to the inventory list. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. All form fields must be populated as well. |
| **Test Steps** | 1. Fill out all the input fields. 2. Click on the save button. |
| **Expected Results** | The form is saved, and user redirected to the inventory view page and the item is added to the inventory list. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. A new item is added to the inventory.   Fail:   1. New item is not added to the inventory. 2. Wrong item is added to the inventory. |

### **5.2.10 Inventory Edit Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Inventory Editing |
| **Test Case ID** | TC-58 |
| **Description** | User makes changes to inventory items. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. There must also be at least one item on the list. |
| **Test Steps** | 1. Click on any inventory item row. 2. Input any edits needed. 3. Click on the save button. 4. Inventory list is updated. |
| **Expected Results** | The item row is updated with whatever information the user selected/modified. This is displayed in the inventory list. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. The inventory is updated based on changes made.   Fail:   1. The inventory is not updated based on changes made. 2. The inventory is updated wrongly based on changes made. |

### **5.2.11 Inventory Add Tracking Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Add Tracking Entry |
| **Test Case ID** | TC-59 |
| **Description** | User clicks on the add tracking entry button to redirect to the add tracking entry form. Then the user inputs/populates each field. Then the user saves it to add a new tracking entry. |
| **Pre-Condition** | Must be logged in and have owner/manager privileges. |
| **Test Steps** | 1. User selects shift. 2. User selects date (today or any date in the past only) 3. User selects the inventory item to track. 4. User inputs quantity that will change the inventory item quantity for the inventory item selected in step 3. 5. Status is selected to determine whether to add or remove the quantity input in step 4 (inventory item quantity cannot be below 0) 6. Save button is clicked to save the form and create the tracking entry. |
| **Expected Results** | Inventory tracking entry is created in the inventory track page. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. Inventory tracking entry is created. 2. Inventory is updated on the inventory page.   Fail:   1. Inventory tracking is not created. 2. Inventory is not updated on the inventory page. 3. Nothing happens when the button is pressed. |

### **5.2.12 Contact Us Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Contact us |
| **Test Case ID** | TC-60 |
| **Description** | This test ensures the proper functionality of the "Contact Us" feature on the website, guaranteeing a seamless user experience for sending messages. The focus is on validating that all required fields work as intended, that users are appropriately notified of any missing information, and that the message submission process is smooth. |
| **Pre-Condition** | Ensure the website is accessible, no need to user log in |
| **Test Steps** | 1. Navigate to the "Contact Us" page. 2. Attempt to submit the form without entering any information to test the presence of accurate validation. 3. Fill in the contact form with valid information, including name, email, and a message. 4. Submit the form. |
| **Expected Results** | 1. The "Contact Us" page should load without errors, ensuring users can access the feature effortlessly. 2. If attempting to submit the form without entering any information, a clear validation error message should be displayed, indicating the necessity to complete all required fields. 3. The form should accept valid information without any issues, including proper handling of the required fields. 4. If any of the required fields are left blank during submission, a validation error should prevent the process, and the user should be informed about the necessity of completing all required fields. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:   1. Successful pop up message displayed to send out the email.   Fail:   1. No pop up happens when pushing the submit button. |

### **5.2.13 Forgot Password Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Forgot Password |
| **Test Case ID** | TC-61 |
| **Description** | The test case aims to validate the functionality of the "Forgot Password" page, ensuring that users can successfully reset their passwords. |
| **Pre-Condition** | The user should have an existing account with a forgotten password. |
| **Test Steps** | 1. Navigate to the "Forgot Password" page. 2. Enter the registered email address. 3. Click on the "Submit" or "Reset Password" button. 4. Check the registered email for a password reset link. |
| **Expected Results** | 1. The "Forgot Password" page should load without errors. 2. The user should successfully enter their registered email address. 3. After clicking the "Reset Password" button, a confirmation message should appear, indicating that a password reset link has been sent to the provided email address. 4. The user should receive an email containing a valid and functional password reset link |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass:   1. It will redirect to the Reset Password page.   Fail:   1. It will show an error to redirect to the reset password page |

### **5.2.14 Reset Password Page Integration Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Reset Password |
| **Test Case ID** | TC-62 |
| **Description** | This test case aims to validate the functionality of the general "Reset Password" page, ensuring users can successfully reset their passwords. |
| **Pre-Condition** | The user must have initiated a password reset process and received a valid password reset link via valid email. |
| **Test Steps** | 1. Open the password reset link received via email. 2. Verify if the user is directed to the correct password reset page. 3. Enter a new password and confirm the changes. |
| **Expected Results** | 1. Clicking on the password reset link should correctly direct the user to the password reset page without errors. 2. The user should be presented with a clear and user-friendly interface for resetting the password. 3. After entering and confirming the new password, a success message should be displayed, confirming the successful password reset. |
| **Priority** | Medium |
| **Pass/Fail Criteria** | Pass: 1. The message popup should clearly say "successfully send email"   1. Received a valid email verification link to reset the password. 2. Using the verification link user supposed to able to update the new password.   Fail:  1.Not sending any invalid email.  2.shows error and not redirect the page. |

## **5.3 System Tests**

### **5.3.1 Non-Functional Test Cases**

|  |  |
| --- | --- |
| **Test Case Name** | Security |
| **Test Case ID** | TC-63 |
| **Description** | Check to see if our security is good |
| **Pre-Condition** | TC-12, TC-14 |
| **Test Steps** | 1. TC-16 2. User clicks on the cart icon button. 3. User enters "SELECT \* FROM ORDERS” in the notes section 4. User clicks Place Order button 5. User enters credit card number “4242424242424242” 6. User enters “01/26” expiration date 7. User enters “123” as CVV code 8. User enters “Matthew” as name 9. User enters “48184” as zip code 10. User hits pay button |
| **Expected Results** | 1. Taken to the order confirmation page 2. User sees "SELECT \* FROM ORDERS” |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User’s payment is accepted. 2. User is redirected to the order confirmation page. 3. User does not have information on all of our orders in the note section.   Fail:   1. User’s payment is not accepted. 2. User is not redirected to the order confirmation page. |

|  |  |
| --- | --- |
| **Test Case Name** | Stripe API |
| **Test Case ID** | TC-64 |
| **Description** | Checking to make sure Stripe API can handle a lot of users |
| **Pre-Condition** | TC-12, TC-14 |
| **Test Steps** | 1. TC-16 2. User clicks on the cart icon button. 3. User clicks Place Order button 4. User enters credit card number “4242424242424242” 5. User enters “01/26” expiration date 6. User enters “123” as CVV code 7. User enters “Matthew” as name 8. User enters “48184” as zip code 9. User hits pay button 10. Repeat process 100 times |
| **Expected Results** | 1. Taken to the order confirmation page |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. User’s payment is accepted. 2. User is redirected to the order confirmation page.   Fail:   1. User’s payment is not accepted. 2. User is not redirected to the order confirmation page. |

|  |  |
| --- | --- |
| **Test Case Name** | Performance |
| **Test Case ID** | TC-65 |
| **Description** | Checking the performance of our application. |
| **Pre-Condition** | 1. User is not logged in and on the home page.  2. Have a stopwatch.  3. Good internet connection. |
| **Test Steps** | 1. Start stopwatch 2. Click on the menu page. 3. Stop stopwatch 4. User clicks on the home button. 5. Repeat process 100 times |
| **Expected Results** | Load times are under 3 seconds upon page loading. |
| **Priority** | High |
| **Pass/Fail Criteria** | Pass:   1. The load time is under 3 seconds each time.   Fail:   1. The load time is over 3 seconds 3 or more times. |

## **5.4 Test Data**

In the framework of this exhaustive testing document for Pizza Connection, all necessary test data has been meticulously incorporated into the respective test cases. Given the predominant use of white box testing methodologies, minimal supplementary information is required for testers to proficiently address the testing prerequisites.

For those examining the codebase, the GitHub repository serves as a centralized resource, providing accessibility to all testers and developers. This access facilitates a comprehensive understanding of the intricate internal dynamics of the system, thereby supporting effective and insightful code inspection.

In the domain of practical queries and the evaluation of formulated data, testers and developers will be endowed with administrative privileges within the MongoDB database. This elevated status empowers them to scrutinize pertinent collections, documents, and storage files directly within the database. Such granted access ensures a comprehensive exploration of the system's data structures, thereby enhancing the depth and breadth of our testing methodology within the Pizza Connection - MERN Stack with MongoDB environment.

## **5.5 Test Reports**

To ensure the success of the entire project, it is imperative that all test cases pass, and proper documentation is essential. Testers are required to submit test reports, following a straightforward pattern. Each test case will be documented by its unique ID, which testers will reuse for reporting purposes to distinguish between cases. The documentation will include details about the testing process, whether the test passed or failed, and the associated dates.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Date** | **Pass/Fail** | **Explanation (if failed)** |
| TC1 | 11/20/23 | Pass |  |
| TC2 | 11/20/23 | Pass |  |
| TC3 | 11/20/23 | Pass |  |
| TC4 | 11/20/23 | Pass |  |
| TC5 | 11/20/23 | Pass |  |
| TC6 | 11/20/23 | Pass |  |
| TC7 | 11/20/23 | Pass |  |
| TC8 | 11/20/23 | Pass |  |
| TC9 | 11/20/23 | Pass |  |
| TC10 | 11/20/23 | Pass |  |
| TC11 | 11/20/23 | Pass |  |
| TC12 | 11/20/23 | Pass |  |
| TC13 | 11/20/23 | Pass |  |
| TC14 | 11/20/23 | Pass |  |
| TC15 | 11/20/23 | Pass |  |
| TC16 | 11/20/23 | Pass |  |
| TC17 | 11/20/23 | Pass |  |
| TC18 | 11/20/23 | Pass |  |
| TC19 | 11/20/23 | Pass |  |
| TC20 | 11/20/23 | Pass |  |
| TC21 | 11/20/23 | Pass |  |
| TC22 | 11/20/23 | Pass |  |
| TC23 | 11/20/23 | Pass |  |
| TC24 | 11/20/23 | Pass |  |
| TC25 | 11/20/23 | Pass |  |
| TC26 | 11/20/23 | Pass |  |
| TC27 | 11/20/23 | Pass |  |
| TC28 | 11/20/23 | Pass |  |
| TC29 | 11/20/23 | Pass |  |
| TC30 | 11/20/23 | Pass |  |
| TC31 | 11/20/23 | Pass |  |
| TC32 | 11/20/23 | Pass |  |
| TC33 | 11/20/23 | Pass |  |
| TC34 | 11/20/23 | Pass |  |
| TC35 | 11/20/23 | Pass |  |
| TC36 | 11/20/23 | Pass |  |
| TC37 | 11/20/23 | Pass |  |
| TC38 | 11/20/23 | Pass |  |
| TC39 | 11/20/23 | Pass |  |
| TC40 | 11/20/23 | Pass |  |
| TC41 | 11/20/23 | Pass |  |
| TC42 | 11/20/23 | Pass |  |
| TC43 | 11/20/23 | Pass |  |
| TC44 | 11/20/23 | Pass |  |
| TC45 | 11/20/23 | Pass |  |
| TC46 | 11/20/23 | Pass |  |
| TC47 | 11/20/23 | Pass |  |
| TC48 | 11/20/23 | Pass |  |
| TC49 | 11/20/23 | Pass |  |
| TC50 | 11/20/23 | Pass |  |
| TC51 | 11/20/23 | Pass |  |
| TC52 | 11/20/23 | Pass |  |
| TC53 | 11/20/23 | Pass |  |
| TC54 | 11/20/23 | Pass |  |
| TC55 | 11/20/23 | Pass |  |
| TC56 | 11/20/23 | Pass |  |
| TC57 | 11/20/23 | Pass |  |
| TC58 | 11/20/23 | Pass |  |
| TC59 | 11/20/23 | Pass |  |

## **5.6 Test Schedule**

The testing schedule for the web app comprises various stages aimed at ensuring its quality and functionality. Continuous unit testing is scheduled throughout the development phase, with each component or module undergoing testing within one day of its development. All unit tests must be completed by November 19, 2023. Following unit testing, integration testing will commence, taking place within two days of completing the associated unit testing. The deadline for completing all integration testing is set for November 20, 2023. After the integration testing, the system testing will start. This will take place on November 21, 2023.

## **5.7 Test Environment**

The test environment will be an already set up environment where the user is already loaded onto the home page of the application. This will be done on a computer with a mouse and keyboard. In addition, they will also need a stopwatch to determine the timing of pages. This will be necessary to do all the tests provided.

## **5.8 Test Entry and Exit Criteria**

**Test Entry and Exit Criteria for Pizza Connection - MERN Stack**

As a team responsible for testing the Pizza Connection web application, our collective obligations include overseeing various aspects of the application's performance. The following outlines the conditions for when a tester can commence testing and when they can conclude their testing activities:

**Test Entry Criteria:**

* **Feature Implementation:** All requisite features must be fully implemented, equipped with the necessary functionalities, before initiating the testing phase.
* **Test Environment Setup:** The test environment, encompassing hardware, software, and network configurations, should be established and ready for testing.
* **Test Data Availability:** Adequate and relevant test data should be accessible to execute the test cases effectively.
* **Testing Resources:** Allocate essential testing resources, such as test tools, test infrastructure, and proficient personnel, ensuring their readiness for deployment.

**Test Exit Criteria:**

* **Defect Resolution:** All identified defects must be addressed and resolved satisfactorily before concluding the testing phase.
* **Quality and Acceptance Criteria:** The system must meet the defined quality and acceptance criteria, aligning seamlessly with the project's objectives.
* **Feature Functionality:** Every feature within the application should exhibit proper functionality and undergo comprehensive testing.
* **Performance, Security, and Compliance:** Validate all performance, security, and compliance requirements, ensuring strict adherence to the specified standards.

## **5.9 Test Pass and Fail Criteria**

**Test Pass and Fail Criteria for Pizza Connection - MERN Stack**

Defining meticulous pass and fail criteria is imperative, aligning with the specific requirements and objectives of the Pizza Connection web application. Prior to the commencement of testing, it is essential to establish and mutually agree upon these criteria to ensure a comprehensive understanding of anticipated outcomes. The following delineates the detailed criteria for test success and failure:

**Test Pass Criteria:**

**Test Case Execution: Successful** execution of all test cases within the suite without encountering critical failures is mandatory.

**Defect Resolution:**

Resolution and closure of all high-priority and medium-priority defects are prerequisites for test success.

**Functional Requirements:**

The system must fully adhere to all specified functional requirements and user acceptance criteria.

**Performance and Scalability:**

Performance and scalability tests must surpass the defined targets and adhere to predetermined thresholds.

**Security:**

Successful completion of security tests without any vulnerabilities or breaches is imperative.

**Integration Testing:**

All integration points and interfaces with external systems should undergo successful testing.

**Test Fail Criteria:**

**Critical Functionalities:**

Test failure occurs if critical functionalities of the Pizza Connection Web Application do not operate as intended, impeding the progression of test cases.

**Performance Issues:**

Encounter of performance issues, including slow response times, system crashes, or high resource utilization, leads to test failure.

**Critical Defects:**

Identification of critical defects that obstruct the application's proper functioning is grounds for test failure.

## **5.10 Test Suspension and Resumption Criteria**

**Test Suspension and Resumption Criteria for Pizza Connection - MERN Stack**

To ensure a strategic approach to testing Pizza Connection, we have defined criteria indicating when to temporarily suspend or resume testing. These criteria play a vital role in maintaining testing efficiency while adeptly managing resources.

**Test Suspension Criteria:**

* **Critical Issues or Defects:** Testing is suspended when significant problems are identified that render the system unusable or pose risks to its stability, security, or functionality.
* **Insufficient Test Data:** If there is an inadequate supply of data needed for proper test case execution, we opt to pause testing until the necessary data is made available.
* **Issues or Dependencies Hinder Progress:** Testing is put on hold if issues or dependencies arise that impede further test execution. For instance, the unavailability or malfunctioning of a required environment or component hampers the continuity of testing.

**Test Resumption Criteria:**

* **Resolution of Critical Defects:** Once critical defects are addressed and resolved, testing can resume to validate the fixes and ensure their effectiveness.
* **Availability of Previously Unavailable Resources:** Testing is resumed if previously unavailable resources, such as test environments or data, become accessible again. This allows testing to proceed using the restored resources.
* **Restored and Stable Test Environment:** Testing recommences when the test environment is restored to a stable and functional state, including the resolution of any dependencies or issues.

## **5.11 Test Design and Execution**

**Test Design and Execution for Pizza Connection**

A systematic approach to the test design and execution process is crucial for verifying and validating the Pizza Connection web application. This process aims to confirm alignment with requirements, flawless functionality, optimal performance, and a positive user experience. Here's a detailed breakdown of the steps involved:

**Feature and Functionality Identification:** We meticulously identify unique features of Pizza Connection that require testing. These are then broken down into distinct scenarios covering various use cases and potential edge cases.

**Objectives Definition:** Clear articulation of testing objectives is undertaken. This includes the validation of functionality, assessment of usability, evaluation of performance, and scrutiny of security aspects.

**Test Environment Setup:** Proper setup of the testing environment is a priority. This involves actions such as installing necessary tools for interacting with the local server, configuring dependencies, and ensuring readiness for testing both React and MongoDB components within the application. The setup process is crucial for establishing an environment that mirrors real-world conditions, facilitating effective testing of the Pizza Connection web application.

## **5.12 Test Data & Defect Management**

Before executing the Pizza Connection application, having the right test data is crucial. This helps ensure that the application functions correctly and responds appropriately to various commands. The test data encompasses valid and invalid inputs, boundary values, and any specific data necessary for testing. It is essential to generate or collect the required test data and organize it systematically. This might involve creating test databases or spreadsheets. Despite the usage of a MongoDB database within the MERN Stack, effective management of this database is critical throughout the testing process. This includes updating, modifying, or resetting data as needed for different testing phases or iterations. Special attention should be given to handling sensitive or confidential data, ensuring compliance with privacy regulations, and protecting user information for both students and faculty.

In the test execution phase, carefully observe and identify any deviations or issues from the expected behavior. Document these deviations as defects or bugs. Create detailed defect reports for each identified issue, outlining information such as steps to reproduce, severity, priority, and any supporting attachments or screenshots. Collaborate with the development team to investigate and resolve the reported defects. Perform retesting to verify that the issues have been effectively addressed. Once the defects are fixed and verified, mark them as closed or resolved in the defect tracking system, and then resume testing.

Effectively managing test data and defects ensures a thorough testing of the Pizza Connection web application within the MERN Stack, facilitating the identification and resolution of issues and an overall enhancement in the application's quality.

## **5.13 Risk Analysis**

In assessing the test plan document for the Pizza Connection web app project, certain risk factors have been identified. Here is a breakdown of these risks in the context of technical, security, performance, and project management aspects:

**Technical Risks:**

**Compatibility issues:** The risk of potential compatibility issues is acknowledged across different browsers, devices, and screen resolutions. Mitigation involves extensive cross-browser and cross-device testing, ensuring seamless performance in a variety of environments.

**Integration Challenge**: Integrating React, MongoDB, and Bootstrap can create challenges. To deal with this, a fine approach to integration testing will be adopted, smooth functionality and interoperability will be verified.

**Security risks:**

**Data breaches and unauthorized access:** Security risks include the possibility of data breaches and unauthorized access to sensitive information. Mitigation measures involve the implementation of security protocols such as encryption, authentication, and access control. There will be regular security checks and vulnerability assessments.

**MongoDB Security System:** There is a risk associated with misconfiguring or inadequate MongoDB security system. This risk will be addressed.

**Performance risks:**

Slow page load times and scalability challenges: Potential performance issues, such as slow page load times and scalability challenges, will be addressed through strategic measures. This includes performance testing, code optimization, caching process implementation, and adherence to best practices for efficient rendering and data collection. Load testing is essential to identify and address performance barriers by ensuring horizontal scalability with MongoDB.

**Project Management Risks:**

**Resource availability constraints:** Risks related to resource availability, including skilled developers, testers, or infrastructure, will be mitigated through effective resource allocation, contingency planning, and transparent communication with stakeholders.

**Schedule Delays:** Project management risks include the possibility of schedule delays affecting the overall delivery timeline. To address this, a realistic timeline will be established, progress will be closely monitored, potential delays will be actively identified, and mitigation measures such as additional resources or adjusted opportunities will be implemented immediately.

In terms of practical questions and evaluating the data created, testers and developers will receive admin privileges within the Pizza Connection MongoDB project. This advanced status gives them the ability to verify relevant collections, documents, and storage files. Such authorized access ensures a comprehensive exploration of the system's data structure, thereby increasing the depth and breadth of our testing methodology within the Pizza Connection – MERN stack – to the MongoDB environment.

## **5.14 Roles and Responsibilities**

**Test Planner: Kanta Islam (Documentation Lead)**

**Responsibilities: Testing the unit test**

**Test Plan Development:** Lead the creation of the overall test plan document, encompassing the testing strategy, objectives, and scope.

**Scope Definition:** Identify and define the testing scope, objectives, and deliverables, ensuring alignment with project requirements.

**Methodology and Approach:** Establish the testing approach, techniques, and methodologies for effective test planning.

**Resource Identification:** Identify and allocate necessary testing resources, collaborating with relevant stakeholders.

**Requirement Gathering:** Collaborate with stakeholders to gather comprehensive testing requirements.

**Progress Monitoring:** Regularly monitor and track testing progress against the plan, identifying potential risks and issues.

**Documentation Oversight:** Review and approve test documentation and reports to ensure adherence to quality standards.

#### **Test Executor: Matthew (Team Lead)**

**Responsibilities: Testing the system test**

**Test Case Execution:** Lead the execution of test cases, scripts, and procedures as outlined in the test plan.

**Testing Types:** Oversee the execution of distinct types of testing to ensure thorough product evaluation.

**Results Documentation:** Supervise the documentation and reporting of test results, emphasizing clarity and accuracy.

**Defect Management:** Collaborate with the Defect Manager to identify, report, and resolve defects promptly.

**Test Improvement:** Lead initiatives for test case reviews, contributing to test scenario design and creation.

**Preparation Assistance:** Contribute to test data preparation and test environment setup.

#### **Defect Manager: Mehedi Zihad (Technical Lead)**

**Responsibilities: Setting up testing environments for each of the testers**

**Defect Management Process:** Establish and maintain an effective defect management process and tools.

**Defect Lifecycle:** Track and manage the lifecycle of reported defects, prioritizing based on severity and impact.

**Collaboration:** Collaborate with the development team to investigate and resolve defects efficiently.

**Verification and Validation:** Verify and validate defect fixes, ensuring quality resolutions.

**Communication:** Communicate defect status and trends to stakeholders, providing comprehensive metrics and reports.

#### **Reporting Role: William Esparza (UI Lead)**

**Responsibilities: Testing the integration test**

**Testing Progress Reports:** Lead the gathering and consolidation of testing progress, results, and metrics.

**Test Reports Preparation:** Prepare detailed test reports, including test summary reports and defect reports.

**Stakeholder Communication:** Communicate testing progress, status, and metrics effectively to relevant stakeholders.

**Issue Identification:** Identify and escalate critical issues and risks promptly.

**Executive-Level Support:** Provide essential support to the test manager in preparing executive-level reports and presentations.

This distribution of roles and responsibilities ensures a structured and collaborative approach to the testing process, with each team member contributing their expertise to achieve the project's testing objectives. Adjustments can be made based on the project's specific needs and requirements.

# **6. Testing Types**

## **6.1 Functional Testing**

Approach:

When it comes to testing the functional aspects of our application, we have adopted a thoughtful strategy. We are using a dedicated testing database that is loaded with predetermined data and results. This ensures that our testing is not only consistent but also thorough. The development team is taking charge of these testing activities, following a carefully planned testing outline. This outline gives them step-by-step instructions for running each functional test case, including what inputs to use and what results to expect. We are focusing on testing key features of the application, such as: Signup, Log In, Forgot Password, Reset Password, Contact us, Pizza Menu, Add to Cart, Order Replacement, Payments, Clock In, Clock Out, and Downloading Pay Period Details. Employee Management, Sale Tracking, Inventory Management, and Timesheet.

Items to be tested:

We have specific test cases (TC1-TC56) lined up for examination to make sure everything works as intended. Each test case comes with a detailed set of steps to be followed during testing. Importantly, these test cases are carried out independently, focusing solely on testing functionality. For instance, the "adding employee" test case follows its own set of steps, separate from the steps for the "user login" test case. This approach ensures a thorough evaluation of each functional aspect of the application.

## **6.2 Non-Functional Testing**

**Approach:**

When it comes to checking the non-functional aspects of our application, we take a detailed approach. We carefully test each functional requirement for performance, reliability, availability, security, and maintainability. Our goal is to ensure that the application not only works as planned but operates at the level it is supposed to, showing the right information, and functioning as intended.

This methodical testing approach means we are looking at every aspect of how the application should perform. We want to make sure it does not just meet the basic requirements but goes above and beyond, delivering accurate results and behaving exactly as intended.

We will examine the performance of each functional test case, evaluating both correct and incorrect data entry, to ensure a swift response time of 3 seconds or less post data entry. This performance standard is consistently applicable to all pages and components.

We will check how dependable each functional test is by looking at how often it is available and how well it performs. The tests will also look at how reliable the data is that comes out, following the steps we have planned and what we expect to happen. It is important to know that our team can only really check how well the functional parts of the app work. Problems with the MongoDB database is something we cannot control, but we picked MongoDB because it is usually reliable, so we are not too worried. For example, when we test showing the team members, we are making sure it reliably shows the right information and the list works well most of the time, aiming for it to work 99.0% of the time.

We make sure everything is secure when we test each part of the app. We check how data is kept safe by using encryption and making sure everything works well. We test MongoDB, which is set up to be secure, and we also test how data is kept safe as it moves between various parts of the app. We follow the rules for keeping user data safe, whether the information is right or wrong. Especially when users put in information or log in, we are incredibly careful. We watch how the data moves around to make sure it has been kept safe by being turned into code that is hard to understand.

Keeping the application easy to manage is an important part of testing each function. We have designed the parts and pages to be easy to maintain, creating a testing environment that is like a building with blocks. This helps the development team add, change, or take away parts for testing in different situations and find mistakes in pages and parts easily. Every test looks at how easy it is to maintain that part, taking away and/or changing parts of the main function to see if it is easy to work with.

We will do each function test on different web browsers like Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. We will also try the browsers on different computer systems to make sure it works well and can be used in many different situations. Each test should show the results we expect in every situation we try. Also, we will test each function on Android/iOS web browser apps. Everyone on the team will do full tests while using different web browsers and systems. The results from our tests should be the same no matter which browser we use.

## **6.3 Unit Testing**

Approach

For unit testing the functional aspects of this application, our approach involves checking each component, including buttons, sorting dropdowns, and any user interaction on our site. Every unit test will be manually performed by each development team member during the testing phase. We will test the consistency of data before and after each unit test for components involving inputs. Each user input within these components will be individually tested for functionality. Buttons and other inputs will be tested in isolation, away from other data inputs as much as possible, before moving on to integration testing.

Items to be tested:

The following test cases, TC1-TC36 all, include buttons or dropdown buttons that users click to input data or interact with the site. Each of these buttons needs to be individually tested, away from any other input or data. Additionally, these test cases involve checkboxes that should be tested one by one, specifically by the development team. The focus is on testing the functionality of each of these types of buttons independently.

## **6.4 Integration Testing**

Approach:

Our method for checking how various parts of the app work together involves testing each piece, page, and function in various orders. This helps us make sure the data stays correct and complete as it moves through each part of the app. Different team members will do the tests in different orders to thoroughly examine all the parts. We test all the parts together and in their final forms, making changes only if absolutely necessary. If we need to remove or modify a part for any reason, we'll mark it for further review. This way, we ensure a comprehensive evaluation of how everything integrates seamlessly.

Items to be tested:

We will conduct integration testing for the following test cases, labeled TC37-TC56, to ensure we get the expected results. Each test case will be examined together, following any specific order we have laid out. We will make sure each page is fully tested before moving on to the next one. For our test cases, the pages are already connected and work together, so we do not need to test the integration of individual elements like login, valid/invalid inputs, or the link to Pizza Connection before checking the whole page. This ensures a thorough evaluation of how everything fits together seamlessly.

## **6.5 System Testing**

Approach:

Our method for testing the entire functional system of this application involves checking each component and page in complete scenarios that cover multiple pages and components. We'll only test the components and pages in their final forms once integration testing is done. Multiple team members will test error handling, data validation, security, and how data is shown across the whole application. This ensures a thorough system test for each functional test case.

Items to be tested:

We will perform system testing for the following test cases, labeled TC57-TC59, to make sure we get the expected results. Each test case will be fully examined, starting from login, and moving to any other page or component, in any order the tester decides. The system will be tested by going through all test cases one after another, starting with a valid login. This way, we'll ensure a comprehensive evaluation of the entire system's functionality.