
Software Requirements Specification

Document Guidelines for

Vineyard Tavern

1. Introduction

1.1 Purpose of the Document:

This SRS document will define the requirements and specifications for our Vineyard Tavern Web Application, a restaurant ordering and management system. It will describe the system's functionality, entities, design structure, and interaction for both customers and the restaurant Admin. The document will serve as a guide for developers, stakeholders, and the instructor overseeing the CSET 120 project.

1.2 Project Overview:

Vineyard Tavern is a web-based restaurant management system that allows customers to browse a menu, place orders, and receive receipts. A restaurant's Admin can log in to add or remove menu items.

1.3 Use Cases:

Overview:

A restaurant management system where there is one admin and multiple customers. The admin can view customer orders, while customers can place orders and view available options.

Audience:

Development team – They will use the SRS document to implement system functionality, design interface, as well as the data base.

Stakeholders – They will base the functionality and usage of the business off the SRS document.

Administration (IT team) – This team will use the SRS document to know what the idea of each functionality is and the order of each step.

Beta testers – This will be used for them to know what the result of the functionality should be that they test.

Actors:

Guest Account – Visitor viewing the menu; unable to add products to the cart; unable to use cart feature

User Account – any registered account

- **Customer** – A registered account that is logged into a pre-made account.
- **Business Admin**– An account that acts as an admin allowing extra functionalities

Manager account – An account that can access the incoming orders from their side.

High-Level Use Cases:

- Browse menu
- Customer adds / remove an item to their cart
- Customer purchases items from cart
- Customer access their purchased order / cancel orders
- Register / log-in
- Admin adds / removes an item to the menu
- Admin viewing incoming orders

2. Project Scope

2.1 Define Scope:

The system includes:

- Menu display and ordering by customers
- Cart management (add/remove items)
- Checkout and payment simulation (cash or card)
- Receipt generation with order summary
- Admin portal to add/delete menu items

Exclusions:

- No real payment processing

2.2 Goals and Objectives:

- Provide a simple, intuitive web interface for customers to order food.
- Allow the admin to easily update menu items.
- Deliver a fast, smooth user experience using HTML, CSS, and JavaScript.

2.3 Constraints:

- All data will be hard coded in JavaScript
- Admin user with predefined credentials
- Limited to web browser functionality (no mobile app)

3. Requirements

3.1Functional Requirements:

Id	Requirement	Description
FR1	Customer sign-up/login	Customers can enter a name and access the menu
FR2	Menu Display	Show all available food items categorized by sections.
FR3	Order placement	Customers can select/deselect items, view subtotals, and proceed to checkout.
FR4	Cart Management	Add/remove menu items dynamically
FR5	Payment	Simulate payment with “Cash” or “Card” option and optional tip field
FR6	Receipt Generation	Show order summary, customer name, total price, and time of order
FR7	Admin Login	Admin can log in using a hardcoded username/password

FR8	Menu Management	Admin can add or delete menu items.
FR9	Extra Features	Two additional features (e.g., “Order time Estimator” and “Customer Feedback form”)

3.2 Non-Functional Requirements

- Performance: All actions should be carried out within 3 seconds. Ex: page loadinging.
- Usability: The interface should use intuitive buttons, readable fonts, and a modern restaurant theme.
- Reliability: data must be maintained regularly throughout the session.
- Security: Admin access protected by certified credentials.
- Maintainability: Source code structured for readability and security.

3.3. Use Case Scenarios:

Use Case 1: Browse Menu

Primary Actor(s): Guest, Customer, Manager

Precondition: User has opened the application.

Trigger: User opens menu page.

Main Success Scenario:

1. User opens website.
2. User navigates to Menu.
3. System displays menu items with details.
4. User browses or searches.

Exceptions:

- Menu fails to load → error.
- Search item not found → “Item not Found.”

Postconditions: User views menu.

Special Requirements: Responsive layout, images, readable formatting.

Use Case 2: Add / Remove Item to/from Cart

Primary Actor: Customer

Preconditions:

Add → User viewing catalog.
Remove → Cart has items.

Triggers: Add or Remove button.

Add – Main Flow:

1. User selects item.
2. Clicks Add.
3. System shows details.
4. User confirms.
5. Item added.

Remove – Main Flow:

1. User opens cart.
2. Selects item.
3. Removes item.
4. Cart updates.

Exceptions:

- Cart fails to load.
- Add fails.
- Removing with empty cart.

Postconditions: Cart reflects changes.

Special Requirements: Remove button always visible.

Use Case 3: Purchase Items from Cart

Primary Actor: Customer

Preconditions: Cart has items; location saved.

Trigger: User clicks Purchase.

Main Flow:

1. User opens cart.
2. Clicks Purchase.
3. Selects payment.
4. Enters info.
5. System validates.
6. Purchase succeeds.
7. Receipt shown.
8. Order stored.

Exceptions:

- Auto-fill for saved payment.
- Invalid payment info.

Postconditions: Order processed.

Special Requirements: Encryption, proper formatting, no full card storage.

Use Case 4: View / Cancel Past Orders

Primary Actor: Customer

Preconditions: User has past orders.

Trigger: Click Display Past Orders.

Main Flow:

1. User clicks Past Orders.
2. System loads order page.
3. Reads database.
4. Displays order history.
5. User reviews.

Exceptions:

- No past orders.
- Database inaccessible.

Cancel Order – Alternate Flow:

1. User selects cancellable order.
2. System checks status.
3. Confirms.
4. Updates database.
5. Shows cancellation success.

View Details – Alternate Flow:

1. User selects order.
2. Details shown.

Special Requirements: Max 3-sec load, clear statuses, scrollable UI.

Use Case 5: Register / Log-In

Primary Actor: Guest

Precondition: User not logged in.

Trigger: Register or Login clicked.

Register – Main Flow:

1. Guest clicks login / register button.
2. System sends the user to the login / register page
3. Guest Clicks register button
4. Guest Enters info.
5. System Checks if the system has the account
6. System creates accounts.

Login – Main Flow:

1. User clicks Login.
2. Enters credentials.
3. System validates.
4. User logs in.

Exceptions: Email exists.

Postconditions: User logged in.

Special Requirements: Secure storage, password rules.

Use Case 6: Admin Adds / Removes Menu Items

Primary Actor: Admin

Preconditions:

Add: Admin logged in; has Add permissions; required item info ready.
Remove: Admin logged in; has Remove permissions; item exists.

Triggers:

Add → Click "Add Item."
Remove → Click "Remove Item."

Postconditions:

Add: New items are now in the menu

Remove: Item removed; menu updated.

Main Success Scenario – Add Item:

1. Admin opens Menu Management.
2. Selects Add Item.
3. System loads form.
4. Admin inputs item details.
5. Admin submits.
6. System validates fields.
7. System checks for duplicates.
8. generated unique ID
9. Item stored.
10. Confirmation displayed.

Main Success Scenario – Remove Item:

1. Admin opens Menu Management.
2. Selects Remove Item.
3. Item list appears.
4. Admin selects item.
5. Clicks Remove.
6. System confirms.

7. Admin approves.
8. System removes item.
9. Confirmation displayed.
10. Item no longer appears.

Exceptions – Add:

- Duplicate or similar item detected (warning + option to cancel/continue).
- Missing required fields.
- Invalid data format.
- Image upload failure.
- Database/storage unavailable.
- Insufficient permissions.
- Concurrent modification detected.

Exceptions – Remove:

- Item not found.
- Item locked (part of active order).
- Admin cancels confirmation.
- Database error.
- Insufficient permissions.
- Concurrent modification.

Special Requirements:

- Unique ID for every item.
- Duplicate detection must include fuzzy matching.
- Add/Remove complete within 3 seconds.
- Validation must highlight invalid fields.
- All failures logged.
- Removal must verify no active dependencies.
- Clear separation of Add and Remove UI.

Use Case 7: Admin Viewing Incoming Orders

Primary Actor: Admin

Precondition: System has orders; admin logged in.

Trigger: Admin clicks View Incoming Orders.

Main Flow:

1. Admin opens Dashboard.
2. Selects Incoming Orders.
3. System retrieves active orders.
4. Orders displayed with: ID, customer, items, total, timestamp, status.
5. Admin reviews.

Exceptions:

- No orders → message.
- Database failure.
- Corrupted order data.

Postconditions: Admin sees real-time order list.

4. System Design

4.1 Entities and relationships

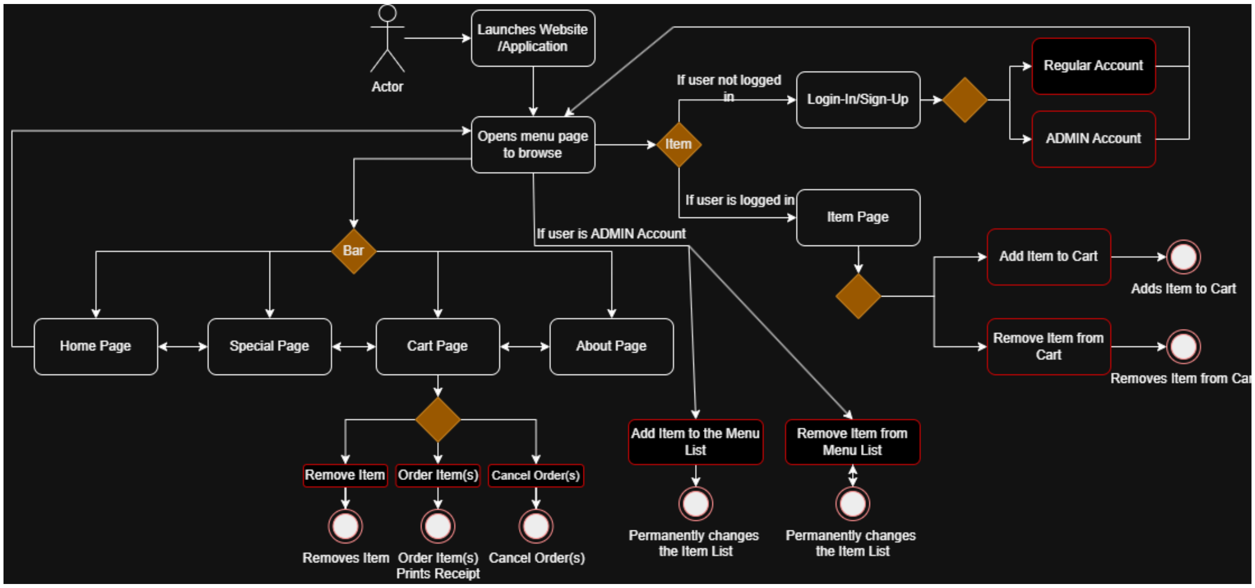
Entity	Attributes	Methods
Admin	ID, Name, username, password	addItem(), deleteItem(), viewOrders().
Customer	Id, Name, OrderList	placeOrder(), viewOrder(),
Order	OrderID, CustomerName, itemList, totalPrice, timeStamp	updateStatus(), printReceipt()
Menu Item	itemID, Name, Price, category.	viewDetails().

Relationships:

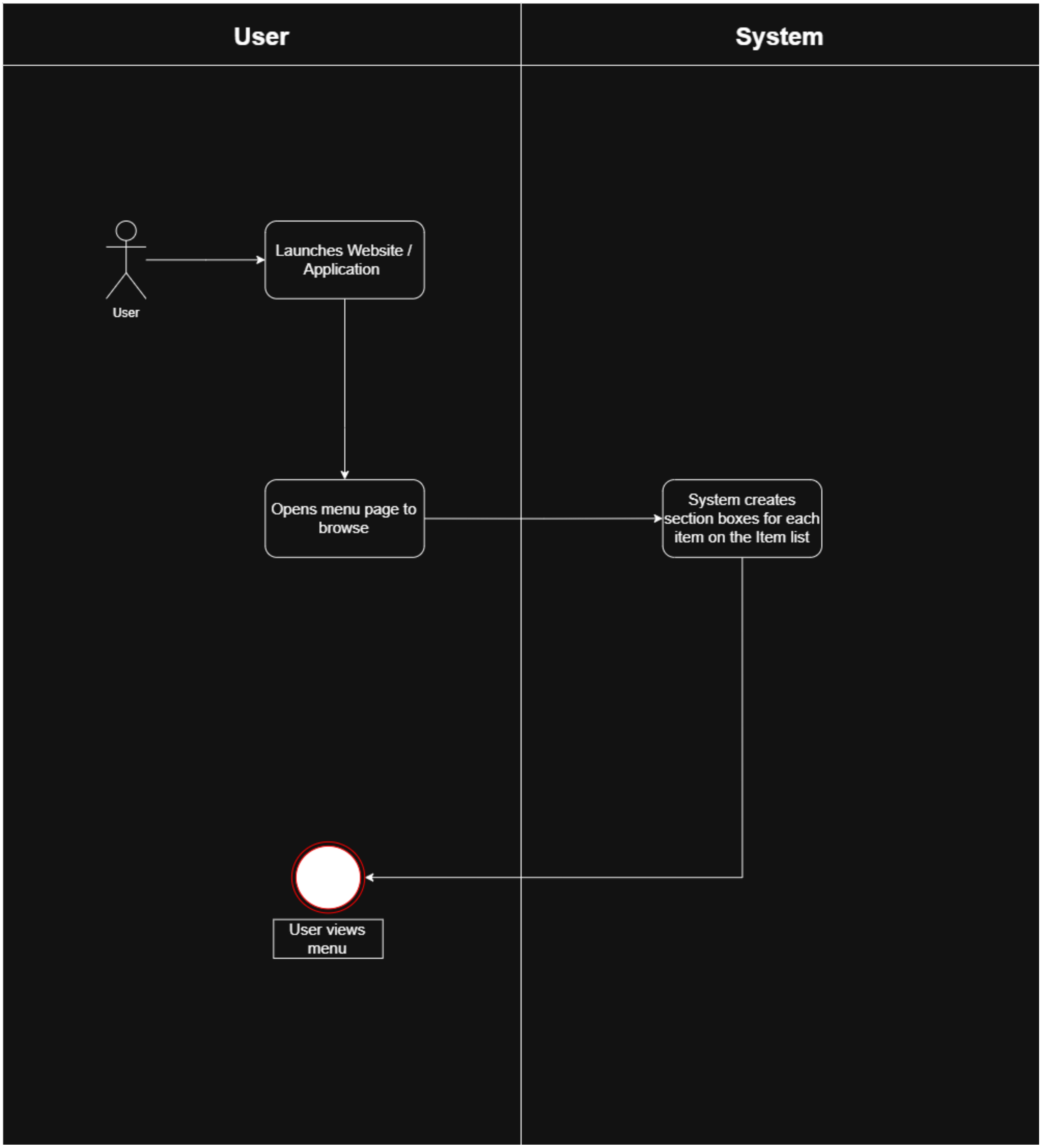
- One Admin oversees multiple Customers
- Each customer can place multiple orders
- Each order contains multiple items from the menu.

4.2 UML class Diagram

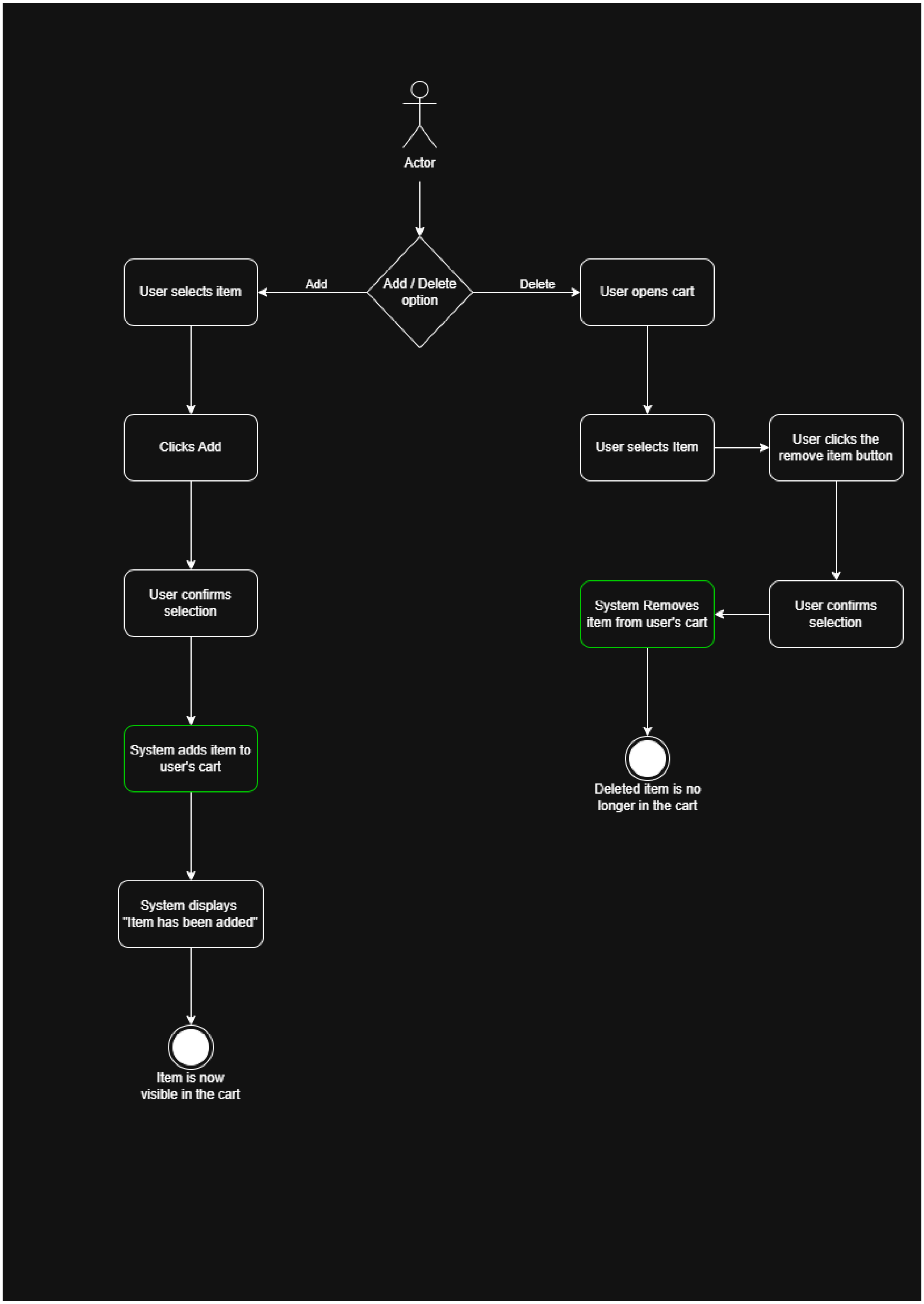
UML:


Actor
Launches Website /Application
Opens menu page to browse
Item
If user not logged in
Login-In/Sign-Up
Regular Account
ADMIN Account
If user is logged in
Item Page
Add Item to Cart
Adds Item to Cart
Remove Item from Cart
Removes Item from Car
If user is ADMIN Account
Bar
Home Page
Special Page
Cart Page
About Page
Remove Item
Order Item(s)
Cancel Order(s)
Removes Item
Order Item(s) Prints Receipt
Cancel Order(s)
Add Item to the Menu List
Permanently changes the Item List
Remove Item from Menu List
Permanently changes the Item List

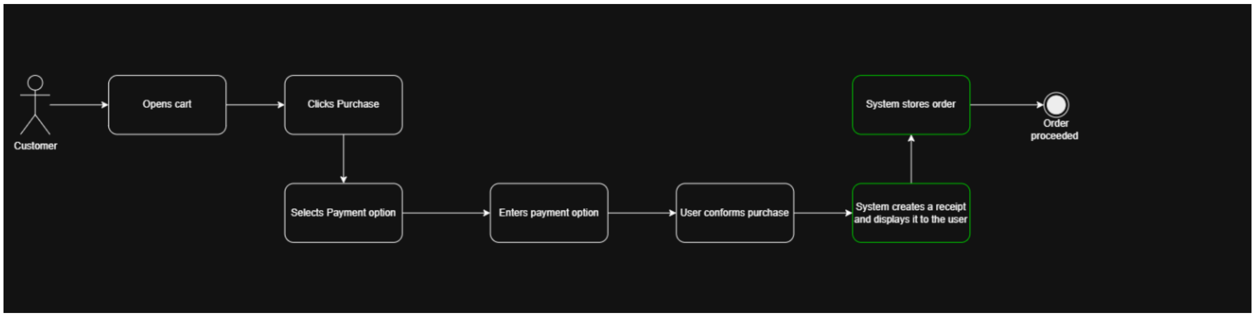
Use 01: Browse menu


User
System
User
Launches Website / Application
Opens menu page to browse
System creates section boxes for each item on the Item list
User views menu

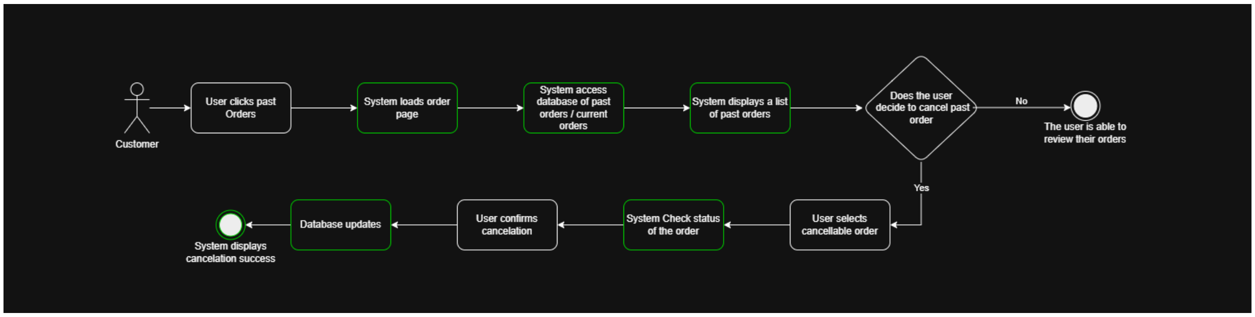
Use 02: Add / Remove item to/from cart


Actor
Add / Delete option
Add
Delete
User selects item
Clicks Add
User confirms selection
System adds item to user's cart
System displays "Item has been added"
Item is now visible in the cart
User opens cart
User selects Item
User clicks the remove item button
User confirms selection
System Removes item from user's cart
Deleted item is no longer in the cart

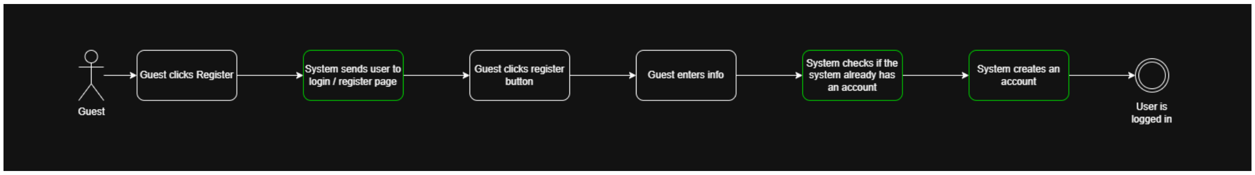
Use 03: Purchase Items from cart



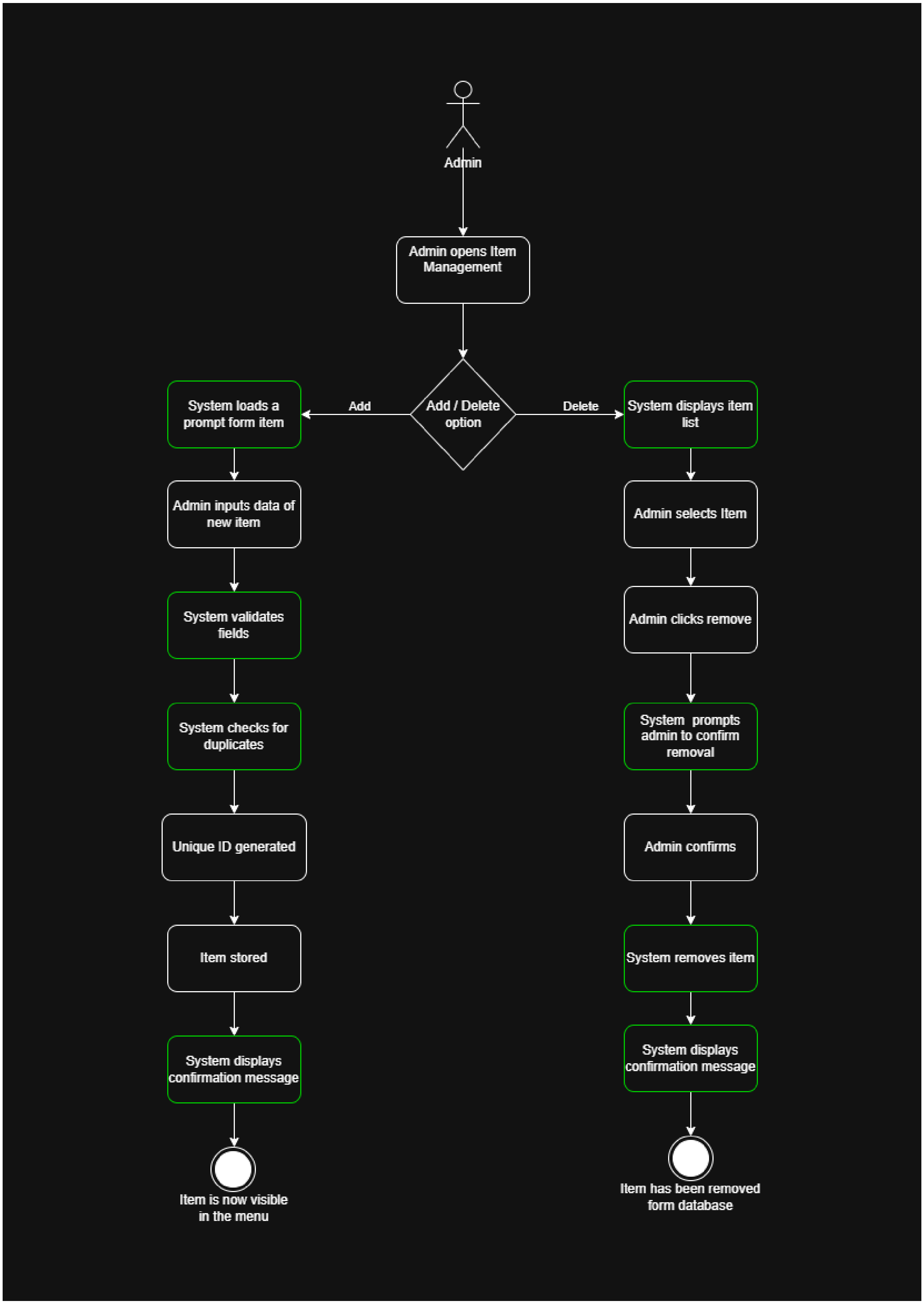
Use 04: View/ Cancel Past Orders


Customer
User clicks past Orders
System loads order page
System access database of past orders / current orders
System displays a list of past orders
Does the user decide to cancel past order
No
The user is able to review their orders
Yes
User selects cancellable order
System Check status of the order
User confirms cancelation
Database updates
System displays cancelation success

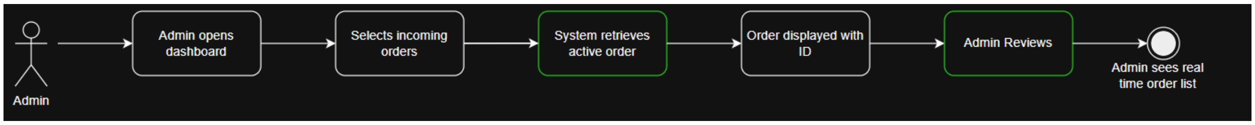
Use 05: Register/ Log-In


Guest
Guest clicks Register
System sends user to login / register page
Guest clicks register button
Guest enters info
System checks if the system already has an account
System creates an account
User is logged in

Use 06: Admin Adds/ Removes Menu Items


Admin
Admin opens Item Management
Add / Delete option
Add
Delete
System loads a prompt form item
Admin inputs data of new item
System validates fields
System checks for duplicates
Unique ID generated
Item stored
System displays confirmation message
Item is now visible in the menu
System displays item list
Admin selects Item
Admin clicks remove
System prompts admin to confirm removal
Admin confirms
System removes item
System displays confirmation message
Item has been removed form database

Use 07: Admin Viewing Incoming Orders



5. System Architecture

5.1 architecture overview

A client web application using:

- HTML/CSS for structure and design
- JavaScript for logic and user interaction
- Arrays to store menu and order data
- JSON to hold local storage that will keep changes throughout iterations

5.2 Component Design

Customer User-Interaction Component:

- Create and manage user accounts (registration & login)
- Browse menu and view item details
- Add or remove items from the cart
- Purchase items using simulated payment methods, and entering information
- View and cancel past orders

Key Interactions:

- **Menu Component:** Retrieves menu items from the storage to browse.
- **Cart Component:** Sends add/remove item requests to the system.
- **Order Processing Component:** Sends purchase requests and receives receipts.
- **Order History Component:** Retrieves previous orders and cancellation options.

2. Guest Component

Responsibilities:

- View menu items (read-only access)
- Limited system access (cannot add to cart or order)

Key Interactions:

- **Menu Component:** Displays menu data without user-specific features.

3. Manager Component

Responsibilities:

- View incoming customer orders
- Monitor order statuses
- Track restaurant activity

Key Interactions:

- **Order Processing Component:** Retrieves real-time active orders.
- **Admin Component:** Shares similar permissions but focused on order view rather than menu modification.

4. Admin Component

Responsibilities:

- Add new menu items
- Remove existing menu items
- Validate item data before submission (duplicate detection, formatting, missing fields)

Key Interactions:

- **Menu Component:** Sends item add/remove requests.
- **Database Component:** Stores and modifies menu item records.
- **Authentication Component:** Confirms admin permissions and access levels.

5. Menu Component

Responsibilities:

- Store and display menu items (name, description, price, category, image)
- Make the menu responsive (Media Queries)

Key Interactions:

- **Customer Component:** Allows browsing and item selection.
- **Admin Component:** Receives add/remove/modify requests.
- **Database Component:** Retrieves and updates item records.

6. Cart Component

Responsibilities:

- Maintain a temporary list of customer-selected items
- Allow item quantity modification
- Sync cart changes in real-time with the JS temporary storage

Key Interactions:

- **Customer Component:** Issues add/remove item actions.
- **Menu Component:** Supplies item details when added.
- **Order Processing Component:** Sends cart contents for purchase.

7. Order Processing Component

Responsibilities:

- Process payment information (simulated)
- Validate user's payment data
- Generate receipts and log order data
- Update order status as it progresses

Key Interactions:

- **Cart Component:** Accepts final purchase request with items.
- **Customer Component:** Sends receipts for display.
- **Database Component:** Stores order records.
- **Manager Component:** Provides incoming order data.

8. Order History Component

Responsibilities:

- Retrieve past orders from the database
- Provide detailed view of past receipts
- Validate whether an order is cancellable
- Update order status after cancellation

Key Interactions:

- **Order Processing Component:** Provides stored order data.
- **Customer Component:** Displays order history UI.
- **Database Component:** Stores updated order statuses.

9. Account Creation (& Admin Verification) Component

Responsibilities:

- Registering new user accounts
- Validate login credentials (Admin Specifically)

Key Interactions:

- **Customer Component:** Handles all account/user operations.
- **Admin Component:** Verifies admin access.
- **Database Component:** Stores account information.

10. Database (Local Storage) Component

Responsibilities:

- Store menu items, orders, users, receipts
- Maintain data integrity
- Guarantee consistent read/write operations
- Support fuzzy-matching duplicate checks for menu items

Key Interactions:

- **Menu Component:** Stores / displays items.
- **Order Processing Component:** Stores purchases & receipts.
- **Admin Component:** Adds/removes items.
- **Account Component:** Stores user credentials securely.

6. Testing and Validation

Test Case 1 – Verify Menu Items Load Correctly

Test Case ID: TC-01

Functional Requirement ID: FR2 (Menu Display)

Test Title: Verify Menu Loads and Displays Items

Test Description:

Test whether the menu loads successfully on startup and ensures that, if an item fails to load, the system displays an "Item Not Found" placeholder message.

Preconditions:

- Users have access to the internet.
- User has opened the Vineyard Tavern website.

Test Steps:

1. Navigate to the homepage.
2. Click on the "Menu" button.
3. Wait for the menu page to load fully.

Input Data:

- No input required (page load only).

Expected Result:

- Menu loads within 3 seconds.
- Each menu item displays the following:
  - Name
  - Description
  - Price
  - Image

- If any item fails to load, a placeholder “Item Not Found” message or icon appears instead.

Test Case 2 – Add Item to Cart

Requirement ID: FR4 (Cart Management)

Test Title: Add Item to Cart

Test Description:

Verify that the user can successfully add an item to the cart.

Preconditions:

- User is logged in (customer).
- User is viewing the menu catalog.

Test Steps:

1. Click on an item in the menu.
2. Click “Add Item to Cart.”
3. Confirm the add action if prompted.

Input Data:

- Selected menu item.

Expected Result:

- Item appears in the cart.
- Cart totally updates correctly.

Actual Result:
TBA

Status:
Pass / Fail

Test Case 3 – Remove Item from Cart

Requirement ID: FR4 (Cart Management)
Test Title: Remove Item from Cart
Test Description:
Verify that the user can remove an item from the cart correctly.

Preconditions:

- Cart contains at least one item.

Test Steps:

1. Navigate to the cart page.
2. Click "Remove Item" on any item.

Input Data:

- Item selected for removal.

Expected Result:

- Item is removed from the cart.
- Cart total updates.

Actual Result:
TBA

Status:
Pass / Fail

Test Case 4 – Successful Checkout

Requirement ID: FR5 (Payment)
Test Title: Complete Checkout with Valid Payment Information
Test Description:
Verify that the user can successfully purchase items when valid payment data is entered.

Preconditions:

- User is logged in.
- Cart contains at least one item.

Test Steps:

1. Open the cart.
2. Click the “Purchase Items” button.
3. Select a payment method.
4. Enter valid payment details.
5. Click “Submit Payment.”

Input Data:

- Valid card information (ex: 4242 4242 4242 4242).

Expected Result:

- Payment succeeds.
- A receipt is generated and displayed.
- Order is stored in the database.

Actual Result:
TBA

Status:
Pass / Fail

Test Case 5 – Invalid Payment Attempt

Requirement ID: FR5 (Payment)
Test Title: Invalid Payment Information Should Fail
Test Description:
Verify that the system shows an error when invalid payment information is entered.

Preconditions:

- User is logged in.
- Cart contains at least one item.

Test Steps:

1. Start checkout process.
2. Enter invalid payment details (e.g., wrong CVV or expired card).
3. Submit the payment.

Input Data:

- Invalid payment info.

Expected Result:

- System displays error message.
- Payment is not processed.
- User stays on checkout screen.

Actual Result:
TBA

Status:
Pass / Fail

Test Case 6 – Register New User

Requirement ID: FR1 (Customer Sign-Up/Login)
Test Title: Register Account Successfully
Test Description:
Verify that a new user can register with valid information.

Preconditions:

- User is not logged in.

Test Steps:

1. Click “Register.”
2. Enter valid First Name, Last Name, Email, and Password.
3. Submit the form.

Input Data:

- Valid user credentials.

Expected Result:

- Account is created.
- Confirmation email is sent.
- User is able to log in.

Actual Result:
TBA

Status:
Pass / Fail

Test Case 7 – Login with Correct Credentials

Requirement ID: FR1 (Customer Login)
Test Title: Login Successfully
Test Description:
Verify that a returning user can successfully log in.

Preconditions:

- Account already exists.

Test Steps:

1. Click "Login."
2. Enter valid email/password.
3. Submit login form.

Input Data:

- Valid email & password.

Expected Result:

- Login succeeds.
- User is redirected to the menu.

Actual Result:
TBA

Status:
Pass / Fail

Test Case 8 – Admin Adds Menu Item

Requirement ID: FR8 (Menu Management)
Test Title: Admin Adds New Menu Item

Test Description:
Verify that the admin can successfully add a new menu item.

Preconditions:

- Admin is logged in.

Test Steps:

1. Navigate to Admin Menu.
2. Click “Add Item.”
3. Enter all required fields (name, description, price, category, image).
4. Submit.

Input Data:

- New item info.

Expected Result:

- Item is added to menu.
- A unique ID is assigned.
- Item shows up on the menu after refresh.

Actual Result:
TBA

Status:
Pass / Fail

Test Case 9 – Admin Removes Menu Item

Requirement ID: FR8 (Menu Management)
Test Title: Admin Removes Existing Item
Test Description:
Verify that the admin can delete an existing item from the menu.

Preconditions:

- Admin is logged in.
- Item exists in menu.

Test Steps:

1. Open Admin Menu.
2. Click “Remove Item.”
3. Select the target item.
4. Confirm removal.

Input Data:

- Menu item to remove.

Expected Result:

- Item is removed from the system.
- Menu updates correctly.

Actual Result:
TBA

Status:
Pass / Fail

Test Case 10 – View Past Orders

Requirement ID: FR6 (Receipt Generation)
Test Title: Customer Views Past Orders
Test Description:
Verify that the user can view their past orders.

Preconditions:

- User has at least one past order.

Test Steps:

1. Log in.
2. Click “Past Orders.”
3. Wait for page to load.

Input Data:

- None.

Expected Result:

- List of past orders appears.
- Each order includes date, total, items, and status.

Actual Result:
TBA

Status:
Pass / Fail

Test Case 11 – Cancel an Eligible Order

Requirement ID: FR6 (Order Lookup + Cancellation)
Test Title: Cancel Order Successfully
Test Description:
Verify that the user can cancel an order that is eligible.

Preconditions:

- Order with “Cancellable” status exists.

Test Steps:

1. Go to Past Orders.
2. Select a cancellable order.
3. Click “Cancel Order.”
4. Confirm cancellation.

Input Data:

- Selected order.

Expected Result:

- Order status updates to “Canceled.”
- Confirmation message appears.

Actual Result:
TBA

Status:
Pass / Fail

Test Case 12 – Admin Views Incoming Orders

Requirement ID: FR7 + Admin Use Case 7
Test Title: Admin Views Incoming Orders
Test Description:
Verify that the admin can view all current incoming orders.

Preconditions:

- System has at least one active order.
- Admin is logged in.

Test Steps:

1. Admin opens Dashboard.
2. Clicks "Incoming Orders."
3. Wait for list to load.

Input Data:

- None.

Expected Result:

- All active orders appear with full details (items, customer, total, timestamp).

Actual Result:
TBA

Status:
Pass / Fail

7. Document Summary

The Vineyard Tavern system is designed as a simple, hardcoded restaurant ordering and management tool that allows customers to browse a menu, add or remove items from a cart, place orders, and view past receipts. Its primary goal is to deliver a straightforward web-based interface where all menu items, orders, and user accounts are stored in predefined JavaScript structures rather than a live database. The system also supports an Admin user who can add or remove menu items and view incoming orders, ensuring the menu and order records stay up to date.

Approvals and Sign-Off

Name	Title / Role	Signature	Date
Owen Sayenga	Developer	X___Owen_Sayenga___	11/ 21 / 2025
Juan J. Vasquez	Developer	X___Juan J. Vasquez___	11/ 21 / 2025
Adrian Renteria	Developer	X___Adrian Renteria__	11/ 21 / 2025
Mohammad Ashraful Huq	Stakeholder	X____________________	11/ 21/ 2025