Test Plan Document

Overview

This document outlines the test functions used to verify each view in the Panda Express Point of Sale System. Each view includes exactly 5 test functions covering normal operations and edge cases.

1. Login View (`/login`)

Acceptance Criteria

- Users can log in with a valid PIN (minimum 4 digits)
- Users can log in with Google OAuth
- Invalid PINs display error messages
- PIN input is limited to 6 digits
- Loading states are displayed during authentication
- OAuth errors are properly handled and displayed

Test Functions

1. **test_valid_pin_login()**
    - **Purpose**: Verify successful login with valid PIN
    - **Steps**: Enter valid 4-6 digit PIN, click "Login with PIN"
    - **Expected**: Redirect to `/employee/kitchen` , session established
    - **Edge Case**: PIN with leading zeros (e.g., "0123")
2. **test_invalid_pin_login()**
    - **Purpose**: Verify error handling for invalid PIN
    - **Steps**: Enter invalid PIN, attempt login
    - **Expected**: Error message "Invalid PIN" displayed, no redirect
    - **Edge Case**: Empty PIN, special characters, non-numeric input
3. **test_pin_length_validation()**
    - **Purpose**: Verify PIN length constraints
    - **Steps**: Enter PINs of length 1-3, 4-6, and >6 digits

- **Expected**: Login button disabled for <4 digits, max 6 digits accepted
- **Edge Case**: Rapid digit entry, copy-paste of long strings

4. **test_google_oauth_login()**
   - **Purpose**: Verify Google OAuth authentication flow
   - **Steps**: Click "Sign in with Google", complete OAuth flow
   - **Expected**: Successful redirect to kitchen page for valid employee accounts
   - **Edge Case**: Unregistered Google account, OAuth callback errors
5. **test_oauth_error_handling()**
   - **Purpose**: Verify OAuth error messages are displayed correctly
   - **Steps**: Attempt login with unregistered account, access denied scenarios
   - **Expected**: Appropriate error messages for AccessDenied, OAuthAccountNotLinked, Callback errors
   - **Edge Case**: Network failures during OAuth, expired tokens

2. Customer Home View (`/home`)

Acceptance Criteria

- All menu category cards are displayed
- Navigation links work correctly
- Loading skeleton is shown during data fetch
- Redirects to login if no session
- Accessibility features are applied

Test Functions

1. **test_category_cards_display()**
   - **Purpose**: Verify all menu category cards are rendered
   - **Steps**: Navigate to `/home` with valid session
   - **Expected**: 5 category cards displayed (Build Your Own, Appetizers, Drinks, Entrees, Sides)
   - **Edge Case**: Empty menu data, missing images
2. **test_category_navigation()**
   - **Purpose**: Verify category links navigate correctly
   - **Steps**: Click each category card
   - **Expected**: Correct navigation to `/home/{category}` routes

- **Edge Case**: Invalid category hrefs, broken links

3. **test_loading_state()**
   - **Purpose**: Verify loading skeleton displays during data fetch
   - **Steps**: Navigate to page, observe loading state
   - **Expected**: Skeleton components shown, then replaced with content
   - **Edge Case**: Slow network, timeout scenarios
4. **test_unauthenticated_redirect()**
   - **Purpose**: Verify redirect to login when no session
   - **Steps**: Access `/home` without valid session
   - **Expected**: "Redirecting to login..." message, redirect to login page
   - **Edge Case**: Expired session, invalid session tokens
5. **test_accessibility_styles()**
   - **Purpose**: Verify accessibility text classes are applied
   - **Steps**: Check text rendering with accessibility settings
   - **Expected**: Text size and styles adjust based on accessibility preferences
   - **Edge Case**: Extreme text size settings, missing accessibility context

3. Build Your Own View (`/home/build`)

Acceptance Criteria

- Meal type cards are displayed (excluding Drink and A La Carte)
- Cards link to correct meal builder pages
- Loading state displays during fetch
- Filters out Party meal types

Test Functions

1. **test_meal_type_display()**
   - **Purpose**: Verify meal type cards are displayed correctly
   - **Steps**: Navigate to `/home/build`
   - **Expected**: Meal type cards shown (Bowl, Plate, Bigger Plate), excluding Drink and A La Carte
   - **Edge Case**: No meal types available, missing images
2. **test_meal_type_filtering()**

- **Purpose**: Verify filtering of excluded meal types
- **Steps**: Check rendered meal types
- **Expected**: Drink, A La Carte, and Party types filtered out
- **Edge Case**: Meal types with null/undefined typeName

3. **test_meal_type_navigation()**
   - **Purpose**: Verify navigation to meal builder pages
   - **Steps**: Click meal type card
   - **Expected**: Navigate to `/home/build/{mealType}` with URL encoding
   - **Edge Case**: Meal types with special characters, spaces in names
4. **test_loading_state()**
   - **Purpose**: Verify loading skeleton during data fetch
   - **Steps**: Observe page during initial load
   - **Expected**: 4 skeleton cards displayed, then replaced with meal types
   - **Edge Case**: API failure, partial data load
5. **test_empty_meal_types()**
   - **Purpose**: Verify handling when no meal types available
   - **Steps**: Access page with empty meal types array
   - **Expected**: No cards displayed, no errors thrown
   - **Edge Case**: API returns null, undefined, or empty array

4. Build Meal Selection View (`/home/build/[meal]`)

Acceptance Criteria

- Progress indicator updates correctly
- Recipe selection works for entrees, sides, and drinks
- Auto-advance between selections
- Quantity selection and price calculation
- Add to cart functionality
- Reset functionality clears selections

Test Functions

1. **test_progress_indicator()**
   - **Purpose**: Verify progress bar updates as selections are made

- **Steps**: Select entrees, sides, drinks sequentially
- **Expected**: Progress percentage increases, visual bar updates
- **Edge Case**: Selecting items out of order, deselecting items

2. **test_recipe_selection()**
   - **Purpose**: Verify recipe selection and storage
   - **Steps**: Click recipes for each required type
   - **Expected**: Selected recipes stored in mealSelections state, visual feedback shown
   - **Edge Case**: Clicking same recipe twice, rapid clicking
3. **test_auto_advance()**
   - **Purpose**: Verify automatic progression between selection types
   - **Steps**: Complete entree selections
   - **Expected**: Automatically advances to sides, then drinks
   - **Edge Case**: Meal type with 0 sides/drinks, skipping optional items
4. **test_quantity_selection()**
   - **Purpose**: Verify quantity input and price calculation
   - **Steps**: Change quantity, verify total price
   - **Expected**: Quantity updates, total = mealType.price * quantity
   - **Edge Case**: Negative quantities, zero, very large numbers, decimal values
5. **test_add_to_cart()**
   - **Purpose**: Verify meal is added to cart correctly
   - **Steps**: Complete selections, set quantity, click "Add to Cart"
   - **Expected**: Meal added with correct structure, toast notification, redirect to build page
   - **Edge Case**: Incomplete selections, network failure during add

5. Category Views (`/home/entree` , `/home/side` , `/home/appetizer` , `/home/drink`)

Acceptance Criteria

- Recipes filtered by type are displayed
- Recipe selection opens quantity dialog
- Quantity can be adjusted
- Items can be added to cart
- Loading states handled correctly

Test Functions

1. **test_recipe_filtering()**
   - **Purpose**: Verify only recipes of correct type are displayed
   - **Steps**: Navigate to each category page
   - **Expected**: Only recipes matching category type shown
   - **Edge Case**: Recipes with null type, recipes with invalid types
2. **test_recipe_selection_dialog()**
   - **Purpose**: Verify selection dialog appears on recipe click
   - **Steps**: Click a recipe card
   - **Expected**: Bottom dialog appears with recipe name, price, quantity input
   - **Edge Case**: Clicking multiple recipes rapidly, clicking during loading
3. **test_quantity_adjustment()**
   - **Purpose**: Verify quantity can be increased/decreased
   - **Steps**: Change quantity in input field
   - **Expected**: Quantity updates, total price recalculates, minimum is 1
   - **Edge Case**: Negative values, zero, non-numeric input, very large numbers
4. **test_add_individual_item_to_cart()**
   - **Purpose**: Verify individual items added correctly
   - **Steps**: Select recipe, set quantity, click "Add to Cart"
   - **Expected**: Item added with correct recipeId, name, type, quantity, price; toast shown
   - **Edge Case**: Adding same item multiple times, network failure
5. **test_cancel_selection()**
   - **Purpose**: Verify cancel button closes dialog
   - **Steps**: Open selection dialog, click Cancel
   - **Expected**: Dialog closes, selectedRecipe reset, quantity reset to 1
   - **Edge Case**: Cancel during add operation, rapid cancel clicks

6. Menu View (`/menu`)

Acceptance Criteria

- Meal plans section displays correctly
- Entrees, sides, and drinks sections render
- Prices formatted correctly

- Images display or show fallback
- Responsive grid layout works

Test Functions

1. **test_meal_plans_display()**
   - **Purpose**: Verify meal plan cards display with correct information
   - **Steps**: Navigate to `/menu`
   - **Expected**: Meal plan cards show name, price, entrees/sides/drinks counts
   - **Edge Case**: Missing meal type data, null prices
2. **test_recipe_sections()**
   - **Purpose**: Verify recipes organized by type in correct sections
   - **Steps**: Check Entrees, Sides, and Drinks sections
   - **Expected**: Recipes filtered and displayed in respective sections
   - **Edge Case**: Empty sections, recipes with missing types
3. **test_price_formatting()**
   - **Purpose**: Verify prices display with 2 decimal places
   - **Steps**: Check price displays throughout page
   - **Expected**: All prices formatted as $X.XX
   - **Edge Case**: Zero prices, very large prices, negative prices
4. **test_image_rendering()**
   - **Purpose**: Verify recipe images display or fallback shown
   - **Steps**: Check recipe cards with and without images
   - **Expected**: Images display when available, placeholder when missing
   - **Edge Case**: Broken image URLs, very large images, missing image paths
5. **test_loading_state()**
   - **Purpose**: Verify loading skeleton during data fetch
   - **Steps**: Navigate to menu page
   - **Expected**: Skeleton components shown, replaced with content
   - **Edge Case**: Slow API response, partial data load

7. Chat View (`/home/chat`)

Acceptance Criteria

- Messages display in chat interface
- User input sends messages to API
- Bot responses displayed correctly
- Function calls (add/remove from cart) work
- Text-to-speech toggle functions
- Conversation history limited to 15 messages

Test Functions

1. **test_message_sending()**
   - **Purpose**: Verify user messages are sent and displayed
   - **Steps**: Type message, click send or press Enter
   - **Expected**: Message appears in chat, sent to `/api/chat` endpoint
   - **Edge Case**: Empty messages, very long messages, special characters
2. **test_bot_response_display()**
   - **Purpose**: Verify bot responses are shown correctly
   - **Steps**: Send message, wait for response
   - **Expected**: Bot message appears with markdown formatting
   - **Edge Case**: API errors, timeout, malformed responses
3. **test_add_to_cart_function_call()**
   - **Purpose**: Verify function calls add items to cart
   - **Steps**: Request item addition via chat
   - **Expected**: Function call detected, item added to cart, confirmation shown
   - **Edge Case**: Invalid item names, out-of-stock items, duplicate additions
4. **test_remove_from_cart_function_call()**
   - **Purpose**: Verify function calls remove items from cart
   - **Steps**: Request item removal via chat
   - **Expected**: Function call detected, item removed, confirmation shown
   - **Edge Case**: Removing non-existent items, removing more than available
5. **test_text_to_speech_toggle()**
   - **Purpose**: Verify TTS can be enabled/disabled
   - **Steps**: Toggle TTS button
   - **Expected**: TTS enabled/disabled, bot messages spoken when enabled

- **Edge Case**: Browser doesn't support TTS, TTS interrupted by new message

8. Checkout View (Cart & Payment)

Acceptance Criteria

- Cart displays all items with quantities
- Subtotal, tax, and total calculated correctly
- Payment method selection works
- Stripe checkout integration functions
- Cash payment creates order
- Email input optional for receipts

Test Functions

1. **test_cart_display()**
   - **Purpose**: Verify cart shows all meals and individual items
   - **Steps**: Add items to cart, view checkout
   - **Expected**: All items displayed with quantities, prices, totals
   - **Edge Case**: Empty cart, very large cart, items with zero quantity
2. **test_price_calculations()**
   - **Purpose**: Verify subtotal, tax, and total calculations
   - **Steps**: Add items, check calculations
   - **Expected**: Subtotal = sum of all items, Tax = subtotal * tax rate, Total = subtotal + tax
   - **Edge Case**: Zero items, negative prices, very large totals, rounding errors
3. **test_payment_method_selection()**
   - **Purpose**: Verify payment method selection (Card vs Cash)
   - **Steps**: Select different payment methods
   - **Expected**: Selected method stored, UI updates accordingly
   - **Edge Case**: No selection, rapid switching, invalid selection
4. **test_stripe_checkout_flow()**
   - **Purpose**: Verify Stripe checkout session creation and redirect
   - **Steps**: Select Card payment, click Pay
   - **Expected**: Order created, Stripe session created, redirect to Stripe checkout
   - **Edge Case**: Stripe API failure, missing Stripe key, invalid order data

5. **test_cash_payment_flow()**
    - **Purpose**: Verify cash payment creates order without Stripe
    - **Steps**: Select Cash payment, click Pay
    - **Expected**: Order created with isCompleted: false, no Stripe redirect
    - **Edge Case**: Order creation failure, missing required fields

9. Cashier Dashboard (`/employee/cashier`)

Acceptance Criteria

- Category selection cards displayed
- Navigation to category pages works
- Home button returns to cashier dashboard
- Layout responsive for cashier interface

Test Functions

1. **test_category_cards_display()**
    - **Purpose**: Verify all category cards render
    - **Steps**: Navigate to cashier dashboard
    - **Expected**: 5 category cards displayed (Build, Appetizer, Drink, Entree, Side)
    - **Edge Case**: Missing categories, layout issues
2. **test_category_navigation()**
    - **Purpose**: Verify navigation to cashier category pages
    - **Steps**: Click each category card
    - **Expected**: Navigate to `/employee/cashier/{category}`
    - **Edge Case**: Invalid routes, broken links
3. **test_home_button()**
    - **Purpose**: Verify Home button returns to dashboard
    - **Steps**: Click Home button from category page
    - **Expected**: Navigate back to `/employee/cashier`
    - **Edge Case**: Multiple rapid clicks, navigation during loading
4. **test_responsive_layout()**
    - **Purpose**: Verify grid adapts to screen size
    - **Steps**: Resize browser window

- **Expected**: Grid columns adjust (3-5 columns based on breakpoints)
- **Edge Case**: Very small screens, tablet orientation

5. **test_cashier_card_rendering()**
    - **Purpose**: Verify CashierCard components render correctly
    - **Steps**: Check card appearance and styling
    - **Expected**: Cards display category names, proper styling
    - **Edge Case**: Long category names, missing styles

10. Kitchen Display View (`/employee/kitchen`)

Acceptance Criteria

- Incomplete orders displayed
- Orders refresh every 5 seconds
- Meal type filter works
- Order completion updates cooked status
- Kitchen drawer shows completed items
- Logout functionality works

Test Functions

1. **test_incomplete_orders_display()**
    - **Purpose**: Verify all incomplete orders are shown
    - **Steps**: Navigate to kitchen page
    - **Expected**: All orders with isCompleted: false displayed
    - **Edge Case**: No orders, very large number of orders
2. **test_auto_refresh()**
    - **Purpose**: Verify orders refresh every 5 seconds
    - **Steps**: Observe page, make order changes
    - **Expected**: Data refreshes automatically, new orders appear
    - **Edge Case**: Network failure during refresh, rapid order changes
3. **test_meal_type_filter()**
    - **Purpose**: Verify filtering by meal type works
    - **Steps**: Click filter buttons (All, Bowl, Plate, etc.)
    - **Expected**: Only orders with selected meal type shown, count updates

- **Edge Case**: Orders with multiple meal types, no orders for type

4. **test_order_completion()**
    - **Purpose**: Verify marking orders as complete
    - **Steps**: Click complete button on order
    - **Expected**: Order marked complete, removed from display, cooked record created
    - **Edge Case**: Completing already completed order, network failure
5. **test_kitchen_drawer()**
    - **Purpose**: Verify drawer shows completed items
    - **Steps**: Complete orders, open drawer
    - **Expected**: Completed items listed with timestamps
    - **Edge Case**: Empty drawer, many completed items

11. Manager Dashboard (`/employee/manager`)

Acceptance Criteria

- Dashboard loads with all tabs
- Tab navigation works
- Header and footer display correctly
- Logo renders properly

Test Functions

1. **test_dashboard_load()**
    - **Purpose**: Verify dashboard loads with AdminTabsCard
    - **Steps**: Navigate to manager dashboard
    - **Expected**: Dashboard renders with tabs card, header, footer
    - **Edge Case**: Missing components, layout issues
2. **test_tab_navigation()**
    - **Purpose**: Verify switching between tabs works
    - **Steps**: Click each tab (Reports, Employees, Inventory, Recipes)
    - **Expected**: Correct tab content displayed, active tab highlighted
    - **Edge Case**: Rapid tab switching, tab content loading failures
3. **test_header_display()**
    - **Purpose**: Verify header with logo and title

- **Steps**: Check header section
- **Expected**: Logo image, "Manager Dashboard" title, subtitle displayed
- **Edge Case**: Missing logo file, long titles

4. **test_footer_display()**
    - **Purpose**: Verify footer renders correctly
    - **Steps**: Check footer section
    - **Expected**: Copyright text, version info displayed
    - **Edge Case**: Footer overflow, responsive issues
5. **test_logo_rendering()**
    - **Purpose**: Verify logo image loads
    - **Steps**: Check logo in header
    - **Expected**: Logo image displays from `/Panda Express/round_logo.png`
    - **Edge Case**: Missing image file, broken image path

12. Manager Reports Tab

Acceptance Criteria

- All report types accessible via accordion
- Date inputs work for date-range reports
- Reports generate with correct data
- Error messages display on failure
- Z-report reset functionality works

Test Functions

1. **test_product_usage_report()**
    - **Purpose**: Verify product usage report generation
    - **Steps**: Select date range, click "Generate Report"
    - **Expected**: Table shows inventory usage data for period
    - **Edge Case**: Invalid date range, no data for period, end date before start date
2. **test_sales_by_item_report()**
    - **Purpose**: Verify sales by item report
    - **Steps**: Select date range, generate report
    - **Expected**: Table shows recipe sales with quantities and revenue

- **Edge Case**: Large date ranges, missing sales data, zero sales

3. **test_x_report()**
    - **Purpose**: Verify X-report (hourly sales) generation
    - **Steps**: Click "Generate Report"
    - **Expected**: Table shows hourly net sales for current day
    - **Edge Case**: No sales today, partial day data, timezone issues
4. **test_z_report()**
    - **Purpose**: Verify Z-report (end of day) generation
    - **Steps**: Click "Generate Report"
    - **Expected**: Table shows hourly totals, reset button available
    - **Edge Case**: No data, reset during generation
5. **test_z_report_reset()**
    - **Purpose**: Verify Z-report reset functionality
    - **Steps**: Generate Z-report, click "Reset Z-Report"
    - **Expected**: Confirmation alert, report data reset
    - **Edge Case**: Reset without generating, network failure

13. Manager Employees Tab

Acceptance Criteria

- Employee list displays all employees
- Add employee dialog opens and saves
- Edit employee updates information
- Delete employee removes from list
- Form validation works
- Employment status toggle functions

Test Functions

1. **test_employee_list_display()**
    - **Purpose**: Verify employee table displays all employees
    - **Steps**: Navigate to Employees tab
    - **Expected**: Table shows name, salary, hours, employed status, role ID
    - **Edge Case**: Empty employee list, very large list, missing data

2. **test_add_employee()**
    - **Purpose**: Verify adding new employee
    - **Steps**: Click "Add Employee", fill form, save
    - **Expected**: Employee created, appears in table, dialog closes
    - **Edge Case**: Missing required fields, invalid data types, duplicate names
3. **test_edit_employee()**
    - **Purpose**: Verify editing existing employee
    - **Steps**: Click Edit, modify fields, save
    - **Expected**: Employee updated, changes reflected in table
    - **Edge Case**: Editing during save, invalid updates, concurrent edits
4. **test_delete_employee()**
    - **Purpose**: Verify employee deletion
    - **Steps**: Click Edit, click Delete, confirm
    - **Expected**: Confirmation dialog, employee removed from table
    - **Edge Case**: Delete without confirmation, delete during edit, network failure
5. **test_employment_status_toggle()**
    - **Purpose**: Verify employment status switch
    - **Steps**: Toggle isEmployed switch
    - **Expected**: Status updates, saved correctly
    - **Edge Case**: Toggle during save, rapid toggling

14. Manager Inventory Tab

Acceptance Criteria

- Inventory list displays all items
- Add/edit inventory items works
- Restock functionality updates stock and creates expense
- Delete inventory removes item
- Form validation prevents invalid data

Test Functions

1. **test_inventory_list_display()**
    - **Purpose**: Verify inventory table displays all items

- **Steps**: Navigate to Inventory tab
- **Expected**: Table shows name, current stock, batch cost, estimated daily use
- **Edge Case**: Empty inventory, very large list, missing fields

2. **test_add_inventory_item()**
   - **Purpose**: Verify adding new inventory item
   - **Steps**: Click "Add New Item", fill form, save
   - **Expected**: Item created, appears in table
   - **Edge Case**: Missing required fields, negative values, invalid data types
3. **test_edit_inventory_item()**
   - **Purpose**: Verify editing inventory item
   - **Steps**: Click Edit, modify fields, save
   - **Expected**: Item updated, changes reflected
   - **Edge Case**: Editing stock during restock, invalid updates
4. **test_restock_functionality()**
   - **Purpose**: Verify restock updates stock and creates expense
   - **Steps**: Click Restock, enter quantity, confirm
   - **Expected**: Stock increased, expense record created, table updated
   - **Edge Case**: Zero quantity, negative quantity, very large quantities, expense creation failure
5. **test_delete_inventory_item()**
   - **Purpose**: Verify inventory item deletion
   - **Steps**: Click Edit, Delete, confirm
   - **Expected**: Item removed from table
   - **Edge Case**: Delete item used in recipes, delete during restock

15. Manager Recipes Tab

Acceptance Criteria

- Recipe list displays all recipes
- Add/edit recipe works
- Ingredient management (add/remove) functions
- Recipe deletion works
- Form validation prevents invalid data
- Premium flag toggle works

Test Functions

1. **test_recipe_list_display()**
   - **Purpose**: Verify recipe table displays all recipes
   - **Steps**: Navigate to Recipes tab
   - **Expected**: Table shows name, type, price, orders/batch, premium status
   - **Edge Case**: Empty list, recipes with null types, missing data
2. **test_add_recipe()**
   - **Purpose**: Verify adding new recipe
   - **Steps**: Click "Create Recipe", fill form, add ingredients, save
   - **Expected**: Recipe created, ingredients saved, appears in table
   - **Edge Case**: Missing required fields, invalid price, duplicate ingredients
3. **test_edit_recipe()**
   - **Purpose**: Verify editing existing recipe
   - **Steps**: Click Edit, modify fields, update ingredients, save
   - **Expected**: Recipe updated, ingredient changes saved
   - **Edge Case**: Editing during save, removing all ingredients
4. **test_add_ingredient()**
   - **Purpose**: Verify adding ingredients to recipe
   - **Steps**: Select inventory item, enter quantity, click Add
   - **Expected**: Ingredient added to list, can't add duplicate
   - **Edge Case**: Duplicate ingredients, zero quantity, invalid inventory item
5. **test_remove_ingredient()**
   - **Purpose**: Verify removing ingredients
   - **Steps**: Click Remove on ingredient
   - **Expected**: Ingredient removed from list, deleted from database if saved
   - **Edge Case**: Removing during save, removing non-existent ingredient

16. Docs View (`/docs`)

Acceptance Criteria

- Documentation page loads
- API and manual documentation links work
- Page renders correctly

Test Functions

1. **test_docs_page_load()**
    - **Purpose**: Verify documentation page loads
    - **Steps**: Navigate to `/docs`
    - **Expected**: Page renders with documentation content
    - **Edge Case**: Missing documentation files, broken links
2. **test_api_docs_link()**
    - **Purpose**: Verify API documentation link works
    - **Steps**: Click API documentation link
    - **Expected**: Navigate to API docs, documentation displays
    - **Edge Case**: Missing API docs, broken links
3. **test_manual_docs_link()**
    - **Purpose**: Verify manual documentation link works
    - **Steps**: Click manual documentation link
    - **Expected**: Navigate to manual docs, content displays
    - **Edge Case**: Missing manual files, broken navigation
4. **test_docs_navigation()**
    - **Purpose**: Verify navigation within documentation
    - **Steps**: Navigate between doc sections
    - **Expected**: Sections load correctly, navigation works
    - **Edge Case**: Deep linking, browser back/forward
5. **test_docs_responsive()**
    - **Purpose**: Verify documentation is responsive
    - **Steps**: Resize browser window
    - **Expected**: Layout adapts to screen size
    - **Edge Case**: Mobile devices, very small screens

Code Coverage Analysis

Expected Code Coverage

Based on this test plan, we would expect **approximately 75-85% code coverage**. We are making sure to test every single major function which in turn also calls the various API routes, event handlers, stat management and utility functions.

We've also done extensive amount of edge case testing. Some types of edge case testing includes empty data, invalid input, network failures, concurrent operations and missing data scenarios.

Some of the more difficult edge cases to check for were related to race conditions where two different browsers were trying to change the same data. Another tough edge case was all of the third-party endpoints since they all had their own nuances.

Verification Approach

The test plan verifies function calls through:

- **Direct Function Calls**: Testing event handlers trigger correct functions
- **API Endpoint Calls**: Verifying HTTP requests to correct endpoints
- **State Changes**: Confirming state updates after function execution
- **UI Updates**: Verifying visual changes indicate function execution
- **Side Effects**: Checking database updates, external API calls, navigation

Regression Testing Effectiveness

This current set up would work really well for regression testing. However, the one thing that we are missing that will make regression testing really effective is in-depth unit testing with mock data. Since our data is being pulled straight from the database the testing software is struggling to get the dynamic data and test the function individually. One thing that we would also need to set up is github action which could run automated tests as soon as a commit or push is made. This type of set up would ensure that we have working software throughout the sprint.