Bee Aware Test Plan - Beekeeping

1. Defining the Requirements

1.1 Functional Test Requirements

Apiaries Module

- Create new apiary
- Edit existing apiary details
- Delete an apiary
- View list of all apiaries
- Search/filter apiaries

These functions cover the CRUD operations for the Apiaries module. Testing these ensures that beekeepers can effectively manage their apiary locations.

Hives Module

- Add new hive to an apiary
- Update hive details (number of frames, queen excluder status, beetle trap status)
- Remove hive from an apiary
- View list of hives in an apiary
- Navigate through hives using Previous/Next buttons

These functions are crucial for hive management. The specific details like frame count and queen excluder status are important for beekeepers to track hive health and productivity.

Inspections Module

- Create new inspection record for a hive
- Edit inspection details
- Delete an inspection record
- View inspection history for a specific hive

Regular inspections are vital in beekeeping. This module allows beekeepers to maintain detailed records of hive conditions over time.

1.2 Technical Design Requirements

- Test responsive layout using Bootstrap's grid system
- Verify correct rendering of module-specific icons in the side navigation
- Test functionality of the collapsible sub-navbar for the Beekeeping module
- Ensure proper routing between Home, Apiaries, Hives, and Inspections pages

The technical design focuses on the user interface elements specific to the application. The side navigation with icons and the collapsible sub-navbar are unique features that need thorough testing.

1.3 Integration Requirements

- Test data flow between Apiaries and Hives modules
- Verify that selecting an apiary correctly filters the associated hives
- Ensure that inspection records are correctly associated with specific hives

These tests verify that the different modules of the application work together, providing a cohesive experience for beekeepers managing multiple apiaries and hives.

2. Planning the Tests

2.1 Resource Identification

- Testing Team: T086
- Development Environment: Visual Studio 2022
- Browsers: Chrome, Firefox, Safari, Edge (latest versions)

2.2 Test Plan Flexibility

- Use Git for version control of the test plan and test scripts
- Implement a branching strategy for developing new test cases

Version control is crucial for maintaining the test plan.

3. Executing the Test Plan

3.1 Testing Approach

- Unit Testing: Focus on controller methods in ApiariesController.cs, HivesController.cs, and InspectionsController.cs
- Integration Testing: Test interactions between controllers and views (e.g., data passing to Index.cshtml files)
- UI Testing: Automate testing of the responsive design and module navigation. Manual testing will also be conducted.

This approach ensures thorough testing at all levels of the application, from individual components to the entire system.

3.2 Detailed Test Scenarios

Apiaries Module Test

1. Navigate to the Apiaries page
2. Click "ADD NEW" button
3. Fill in apiary details
4. Submit form
5. Verify new apiary appears in the list
6. Click "MODIFY" on an existing apiary
7. Change details and save
8. Verify changes are reflected

This scenario tests the core functionality of the Apiaries module, ensuring that beekeepers can effectively manage their apiary locations.

Hives Module Test

1. Go to the Hives page for a specific apiary
2. Verify the hive list loads correctly
3. Use Previous/Next buttons to navigate through hives
4. Click "EDIT" next to frame count
5. Update the number of frames

6. Verify the change is saved and displayed correctly

This test focuses on the unique features of the Hives module, including the navigation between hives and the inline editing of hive details.

Inspections Module Test

1. Select a hive from the Hives page
2. Navigate to the Inspections tab
3. Add a new inspection record
4. Verify the record appears in the inspection history
5. Edit the inspection details
6. Confirm changes are saved correctly

This scenario ensures that beekeepers can maintain accurate records of their hive inspections, a critical aspect of effective beekeeping.

4. Analyzing the Results

4.1 Performance Metrics

- Page load times for Apiaries, Hives, and Inspections pages
- Response time for adding/editing records
- System behavior with simulated data for 100+ apiaries and 1000+ hives

These metrics will help identify any performance bottlenecks in the application, ensuring it can handle the data volume of larger beekeeping operations.

4.2 Reporting

- Use a shared document (e.g., Google Sheets) to log test results
- Include columns for test case ID, description, expected result, actual result, and status
- Generate weekly summary reports highlighting key issues and progress

This structured approach to reporting ensures that all team members have visibility into the testing progress and can quickly identify areas needing attention.

5. Types of Tests

5.1 Unit Testing

- Test CRUD operations in ApiariesController, HivesController, and InspectionsController
- Verify correct handling of edge cases (e.g., deleting a hive with existing inspection records)

5.2 Integration Testing

- Test data flow between controllers and views
- Verify that changes in one module (e.g., deleting an apiary) correctly update related modules

5.3 UI Testing

- Verify responsive design across different screen sizes

5.4 Security Testing

- Verify that users can't access or modify data they shouldn't have access to

5.5 API Testing

- Verify that users and developers can access endpoints. This is especially relevant when a mobile team is hired.

5.6 Load/Stress Testing

- Verify that the website can handle immense traffic. We will use an automated script to stress test.

These tests cover the critical aspects of the application, ensuring functionality, usability, and security across all modules.

6. Test Documentation

6.1 Test Case Example

Test Case ID: TC001 Name: Create New Apiary Objective: Verify that a new apiary can be successfully created Steps:

1. Navigate to Apiaries page
2. Click "ADD NEW" button

3. Enter apiary name: "Test Apiary"
4. Enter location: "Test Location"
5. Click Submit Expected Result: New apiary "Test Apiary" appears in the apiary list

This detailed test case format ensures consistency in testing and makes it easy for any team member to execute the test and understand the expected outcome.