Bee Balanced
Software Testing Plan
Version 1.0

Team Members:
Meaghan Freund
Patricia Madrid
Javier Arribas Gonzales
Gannon Rowlan

Mentor:
Scott LaRocca

Table of Contents

1 Introduction

Bee Balanced is a personalized web application designed to help adolescents build lifelong healthy habits by improving their physical, mental, and social well-being. Through daily health surveys, gamified tracking features like badges and virtual pets, and personalized wellness feedback, the platform aims to keep users engaged and motivated. The system is built using a modern tech stack including React.js, Node.js, Express, MySQL, and AWS, with a strong focus on usability, scalability, and data privacy.

In software engineering, testing plays a crucial role in ensuring the quality, functionality, and usability of an application. It allows developers to validate that the system behaves as expected, uncover bugs before deployment, and ensure that changes do not introduce new issues. A well-structured testing strategy improves reliability, enhances user experience, and ensures long-term maintainability.

For Bee Balanced, our testing plan includes Unit Testing, Integration Testing, and Usability/End-user Testing. Unit testing will focus on validating individual functions like user authentication, data handling, and survey logic using tools such as Mocha, Chai, and Supertest. Integration testing will verify that components like the backend, database, and frontend communicate correctly. Usability testing will ensure that teens, our target audience , can interact with the platform easily and intuitively, using expert reviews, small-group studies, and feedback loops.

The rationale for this specific testing approach stems from the nature of our web application: it's highly interactive, user-centered, and data-driven. Because our target users are adolescents who may quickly disengage with poorly functioning or unintuitive tools, extra care is placed on usability testing. Likewise, since our website handles sensitive wellness data, integration testing is essential to ensure correct and secure data flow between modules. Finally, unit testing helps us maintain code quality and reduce errors at the source. The following sections outline each testing type in more detail, including the tools, techniques, and rationale behind the specific test cases designed for Bee Balanced.

2 Unit Testing

Unit testing helps us make sure that the different components of Bee Balanced work correctly on their own before they are combined into the full system. Our goal is to test important functions like user login, handling user data, and giving health recommendations. This helps catch errors early and makes the app easier to maintain. Bee Balanced is built using Node.js and MySQL for the backend, with express, HTML and CSS for the frontend. To test our code, we will use Mocha/Chai for the backend and frontend, and we will use Supertest for server and API testing.

2.1 Units to be Tested

Unit testing will check different parts of the app to confirm they work properly. Here are the key areas:

2.1.1 Authentication (Logging In and Signing Up): The authentication system lets users sign up, log in, and stay logged in securely. Tests will check:

- Users can create accounts.
- Logins work with correct credentials and fail with wrong ones.
- Passwords are stored and handled securely.
- Sessions and tokens work properly.

2.1.2 User Profiles: Users can create and manage their profiles, including personal health data. Tests will check:

- Profiles can be created, edited, viewed, and deleted.
- Invalid input is rejected.
- Changes to profiles are correctly saved.

2.1.3 Surveys: Users fill out surveys that help generate health recommendations. Tests will check:

- Survey responses are saved correctly.
- The point system for questions is accurate.
- Users can view past survey results.

2.1.4 Feedback: The system gives users personalized health suggestions. Tests will check:

- Recommendations match user data.
- New users without much data get general suggestions.
- Errors don’t break the recommendation system.

2.1.5 API Endpoints: The app uses APIs to send and receive data. Tests will check:

- APIs return the correct data.
- Errors are handled properly.
- Responses follow the expected format.

2.2 How We Are Testing

To make sure our tests actually catch problems, we're going to check if the app works under normal conditions, as well as in situations where things could go wrong. This means testing with both expected and unexpected inputs to see if the system responds correctly and doesn't crash.

2.2.1 Authentication Testing

We will test the signup and login features by creating new accounts and making sure they get stored properly. If someone tries to sign up with an email that's already taken, the system should block it. We'll also test logging in with both correct and incorrect passwords to make sure only real users get in. Another important thing to check is how session tokens work—users should stay logged in securely, but if a token is expired or invalid, the system should prevent access.

2.2.2 User Profiles

User profiles are a big part of the app, so we need to make sure they can be created, updated, viewed, and deleted without any problems. We'll test if the system saves changes properly when a user updates their profile. If someone tries to enter invalid information—like leaving out required fields or entering text where numbers should go—the system should catch that and reject it. We'll also test deleting profiles to make sure all user data is removed correctly.

2.2.3 Surveys

Since surveys are key to personalizing the app experience, we'll run tests to make sure responses are saved properly in the database and linked to the right user. If a user tries to submit an incomplete survey, the system should stop them from doing so. We'll also test whether past survey results are stored and displayed correctly when users go back to check them.

2.2.4 Feedback

The app gives health recommendations based on what users enter in their profiles and surveys, so we need to make sure those suggestions make sense. If a new user doesn't have much data yet, they should get general recommendations. If a user has completed surveys, their

recommendations should be personalized based on their responses. We'll also introduce missing or inconsistent data to see if the system can still function correctly without breaking.

2.2.5 API Endpoints

The app relies on APIs to send and receive data, so we'll test if the APIs return the right information when given correct requests. If a request is wrong or missing something important, the system should send an error message instead of just failing silently. We'll also check that the responses follow the expected format so that everything works smoothly between the frontend and backend.

3 Integration Testing

Integration testing itself is a form of testing the interactions and data exchanges between major components, ensuring the correct information is passed through module to module. In short, the goal of integration testing is to validate whether parameters and return values are accurate to what the user is expecting, guaranteeing a reliable application that provides predictable results. For the Bee Balanced project, the overall approach procedure in determining what to test was considering each component of the application, such as the user account, the surveys, and the progress tracking and how they exchange data between each other. In order to test that the correct data is being transferred between each module, the team will conduct integration testing through database checks and unit tests to ensure that the expected results are produced when moving through each component.

3.1 Modules to be Tested

The architecture of Bee Balanced as an application is very interconnected, weaving every module together to create a cohesive structure. To accurately complete the integration testing phase of software development, the following major relationships between components need to be assessed:

3.1.1 Sign Up to Login: Users are able to create a new account and can immediately log in using that account information.

3.1.2 Login to Home Page: After a user logs into their account, their personalized data is presented in the home page.

3.1.3 Saving Survey Results: A user can take a daily survey and expect to have the accurate results stored to their account.

3.1.4 Progress Page: An accurate representation of the user's results history is displayed on the home page.

3.1.5 Relevant Feedback: Feedback that is applicable to the user is displayed, taking the results and history of the surveys into account.

3.1.6 Survey to Games Transaction: Once a user takes one survey, they can expect to have a mini game unlocked and ready to play.

3.2 How We Are Testing

3.2.1 Sign Up to Login

For the sign up to log in process, we will need to ensure that a user is able to sign up with a specified email and password and can successfully log in using the exact same information. To test this, we will create new accounts and monitor the database to match the newest user addition to the account we entered. Next, we will move to the login page and test incorrect emails and passwords that are similar to the newly created account, establishing a secure system that requires exact authentication. Finally, we will properly sign into the correct account, verifying a correct transition from log in to home page, where we will conduct our next tests.

3.2.2 Login to Home Page

Following the creation of a new account and logging in, we need to certify that the account that is entered holds the expected existing information. We will need at least two accounts to experiment with, one that has a history with actual information and one that has been newly created with zero expected held data. First, we will test with the new account that we created, examining the home page to ensure that there are default values with no current data, checking the account page to confirm that the login credentials are accurate, and looking through the database and verifying that there is no history in the survey table. Moving forward, we will use an account that has been filled with survey results, checking the same measures that were conducted with the empty account. Instead, we will expect a filled home page with progress and held results from previous surveys. As a team, we need to guarantee that the history for a user is what they are expecting to see to provide a complete experience with the website.

3.2.3 Saving Survey Results

After ensuring that the previous entries are to be expected, we will need to confirm that new entries are saved and returned to the user's account. Any account can be used to test here, but for consistency, we will use the fresh account we have created previously. Each of the surveys will be completed to test this transaction, making sure that all are properly integrated with the system. Following completion of each survey, we will match the recorded results with the database, certifying that the correct results have been stored. The next operation for testing will be confirmed with the progress page phase of integration testing, as monitoring survey results heavily ties into the progress section.

3.2.4 Progress Page

Succeeding the completion of the survey results, we will verify that the progression page is updated with the new entry of survey data. After matching the results with the database, we will ensure that the data is being passed to the progress graphics. Visually, we will calculate the new average for the survey point and confirm that the new added point is equal to the expected average.

3.2.5 Relevant Feedback

Following the progress page is the feedback section, which takes into account the existing progress history and translates this into a written report on how to improve any relevant areas of health. We will need to track the most recent entry of survey data and note which health habits have a lower average score in comparison to the others. Then, we will need to test the database and verify that the lowest points have been recorded and saved to be reported to the user. Finally, we will check each health category of feedback, which are overall, mental, and physical, and match if the expected habits are being reported on and give relevant advice to the user. The goal of Bee Balanced is to help users improve upon these results, making this section imperative to test accurate data exchange.

3.2.6 Survey to Games Transaction

Finally, we need to ensure that the survey is connected to the games section, allowing users to play a mini game once they complete a survey. We will test this by checking the accessibility of the mini games before taking a survey, checking if they are still locked to the user. Next, we will complete a survey and check if the status of the game has changed to be unlocked for the user. We will test this for each survey, confirming that each type of survey completed allows the user to access a mini game as expected.

4 Usability/End-user Testing

Usability testing is a critical phase in software development that evaluates how easily and effectively end-users can interact with an application. The primary goal of usability testing is to ensure that the software system provides a smooth and intuitive user experience, minimizing confusion and frustration while maximizing user satisfaction. Usability testing assesses the clarity, efficiency, and accessibility of the system's interface and workflow, identifying potential areas for improvement before full deployment.

Usability testing typically involves real users interacting with the system under controlled conditions while researchers observe and analyze their experiences. This process helps uncover usability issues, such as confusing navigation, unclear instructions, or inefficient workflows. The results inform design refinements that enhance overall user satisfaction and engagement.

4.1 Considerations for Usability Testing

Bee Balanced is a survey-based web application designed to help users track and improve their well-being. Given the nature of our user base, which are individuals looking for health insights who may not be tech-savvy, it is crucial to ensure that the interface is highly intuitive and responsive. The usability testing plan for Bee Balanced considers the following factors:

4.1.1 User Demographics

The application will cater to a broad audience, including users with varying levels of technical proficiency. Testing should include users with minimal technical background to ensure accessibility.

4.1.2 Survey Engagement

Since the core functionality revolves around users completing surveys, it is essential to verify that the process is straightforward, visually appealing, and engaging.

4.1.3 Data Interpretation

Users should be able to easily comprehend their results and insights, including visual data representations using Plotly.js.

4.1.4 Interactivity and Gamification

The virtual pet and mini-games integrated into Bee Balanced should enhance user engagement rather than serve as a distraction or cause confusion.

Based on these considerations, our usability testing will focus on identifying friction points in the user journey, refining the interface, and ensuring a user-friendly experience.

4.2 Usability Testing Plan

The usability testing plan consists of three main components: expert reviews, user studies, and acceptance testing. These methods will provide both qualitative and quantitative insights into the effectiveness of Bee Balanced user interface and workflow.

4.2.1 Expert Reviews

Objective: Gather feedback from usability experts to identify potential design and interaction issues before testing with real users.

- **Who:** Usability specialists, UI/UX designers, and accessibility experts.

- **What:** Experts will review the Bee Balanced interface for clarity, ease of navigation, and adherence to usability best practices.
- **How Often:** Conducted once before user testing, with follow-up evaluations if major changes are made.
- **Recording & Analysis:** Findings will be documented, categorized, and prioritized for implementation based on severity and impact.

4.2.2 User Studies

Objective: Observe real users as they interact with Bee Balanced to identify usability bottlenecks and areas of confusion.

- **Who:** A diverse group of 15-20 users reflecting our target audience.
- **What:** Users will complete a set of tasks, such as signing up, completing a survey, reviewing results, and interacting with the gamified elements.
- **How Often:** Conducted in two rounds—an initial study before major refinements and a follow-up study after implementing improvements.
- **Recording & Analysis:** Sessions will be recorded (with user consent) and analyzed for navigation errors, completion times, and user feedback. Heatmaps and interaction tracking may be used for additional insights.

4.2.3 Acceptance Testing

Objective: Validate that usability requirements have been met before the official launch.

- Who: A final group of users, including some from prior user studies and new participants.
- What: Users will complete a final usability test to ensure all improvements have addressed initial concerns and no new issues have emerged.
- How Often: One round conducted near the end of development.
- Recording & Analysis: Surveys and structured interviews will collect feedback on ease of use, overall satisfaction, and remaining concerns.

4.3 Timeline for Usability Testing

Phase	Activity	Duration	Expected Completion
Expert Review	UI/UX Analysis	1 week	Week 1
User Study - Round 1	Initial user testing	1 week	Week 2
Refinements	Implement Improvements	1 week	Week 3

User Study - Round 2	Follow-up testing	1 week	Week 4
Acceptance Testing	Final validation	1 week	Week 5

Usability testing is essential for ensuring that Bee Balanced provides a seamless and engaging experience for its users. Through expert reviews, real user studies, and acceptance testing, we will iteratively refine the platform to maximize usability and satisfaction. By carefully analyzing user interactions and feedback, we aim to deliver an intuitive and accessible system that encourages long-term engagement and positive health outcomes.

5 Conclusion

The Bee Balanced software test plan outlines a comprehensive approach to ensuring that our web application functions correctly, integrates seamlessly, and provides a smooth and intuitive experience for our users. Through detailed unit testing, we will verify that individual components such as authentication, user profiles, surveys, and API endpoints work as intended. Our integration testing strategy focuses on confirming that these components interact properly, particularly in the areas of user onboarding, data tracking, and personalized feedback delivery.

In addition, we have prioritized usability testing to ensure that adolescents, our primary user base, can easily navigate and engage with the platform. By conducting expert reviews, user studies, and acceptance testing, we aim to identify and fix pain points in the user experience before full deployment.ç

This well-rounded testing regime is tailored to the nature of Bee Balanced as a health-focused, interactive, and data-driven application. With an emphasis on usability, reliability, and secure data handling, our testing plan is designed to deliver a polished, error-resistant, and engaging product. We are confident that following this plan will result in a high-quality application that meets user needs and supports positive, long-term health habits.