Test Documentation( Oct 19 Start 6:30 PM)

Team 11

Personalized Recovery Coach App

**Pawan Kumar Gupta**

**Arnav Singh**

**Victor Zhou**

**Thomas El Hindi**

**William Caron**

**Alexandre Robidoux**

**Course Instructor:**

**Dr. Wahab Hamou-Lhadj**

What we need in this document:  
  
1. Manual Testing

Manual testing is a **software testing process** in which a tester executes test cases **without using automation tools**. Instead, the tester manually performs actions—such as clicking buttons, entering inputs, or navigating through workflows—to verify that the software behaves according to the specified requirements.

The primary purpose of manual testing is to ensure that the application is **bug-free, stable, and user-friendly** before release. Unlike automation, which depends on scripts and tools, manual testing leverages **human intuition, creativity, and observation**, making it especially effective for detecting usability issues and unexpected defects.

**Quick Facts:**

* Manual testing is performed at various levels—**unit, integration, system, and acceptance testing**.
* It is commonly used for **exploratory testing, ad-hoc testing, and UI/UX validation**, where human judgment is essential.
* No programming knowledge is required, making it accessible to beginner testers.
* Every application must undergo some level of manual testing before automation can be applied.

In short, manual testing is the **foundation of quality assurance**, ensuring that both functional and non-functional requirements are validated from an **end-user’s perspective**.

* **User Experience (UX) Validation:** Automated tools can check if a button is clickable, but only a human can determine if it’s intuitively placed, aesthetically pleasing, or provides adequate feedback. **Over 70% of projects cite “User Experience” as a primary reason for manual testing.**
* **Exploratory and Ad-hoc Testing:** This unscripted testing relies on tester skill and creativity to find edge-case defects that scripted tests would miss.
* **Cost-Effectiveness for Short Projects:** For short-lifecycle projects or those with frequently changing UIs, the investment in writing and maintaining automated scripts can be greater than the value they provide. Manual testing offers a flexible and immediate solution.

**Key objectives include:**

1. **Bug Identification** – Detecting functional, logical, or design defects before release.
2. **Requirement Validation** – Verifying that the application meets both functional and non-functional specifications.
3. **User Experience Assurance** – Testing from an end-user perspective to confirm that workflows are intuitive and error-free.
4. **Regression Confidence** – Retesting fixed defects to ensure new changes do not break existing functionality.
5. **Quality Delivery** – Delivering a stable, reliable, and bug-free product to customers.

**2.Unit Testing( to be done for every feature)**

1. [Unit Testing](https://www.guru99.com/unit-testing-guide.html) in manual testing focuses on verifying individual components, methods, or small code units separately. Each unit is tested for correctness by supplying inputs and reviewing the results closely. Developers or testers manually validate results before integration. While automation is common here, manual unit testing is still applied during initial builds, prototypes, or quick debugging activities.

**3. System Testing(only to be done when project is complete)**

1. [System Testing](https://www.guru99.com/system-testing.html) in manual testing examines the complete, integrated application as a whole. Testers simulate real-world user actions to confirm that all combined modules work seamlessly together. This testing ensures usability, performance, and business logic correctness. Manual system testing is critical for uncovering integration side effects and verifying that the software delivers exactly what was specified in the requirements.

**4. Integration Testing only to be done when different tasks are complete and added to the test code)**

1. [Integration Testing](https://www.guru99.com/integration-testing.html) in manual testing validates the interaction between multiple modules or systems. Testers manually pass data across interfaces, monitor workflows, and ensure accurate communication between components. This process helps uncover mismatches, incompatible formats, or broken data flows. Manual integration testing is highly useful when APIs, third-party services, or database connectivity need validation in controlled test cycles.

**5. Acceptance Testing**

1. [Acceptance Testing](https://www.guru99.com/user-acceptance-testing.html) in manual testing confirms whether the entire application meets business expectations and customer requirements. End-users, stakeholders, or testers verify real-life scenarios like workflows, usability, and reliability before product release. Types include Alpha Testing (internal) and Beta Testing (external). Manual acceptance ensures the product offers a satisfactory experience and is ready for actual market use.

Manual Testing Plan (so far):  
  
**Test Scenario:** Sprint 1 objectives for user complete: User navigation of home page, user registration, heartbeat display when triggered.

**TEST PLANNING:** The home page is complete – meaning has all buttons mentioned in UI1 in sprint 1.

2. All buttons work properly and take user to the desired page (the page functionality may not be accurate but have some indication of what it does).

3. Pop ups show up when user entres wrong input or not connected to Bluetooth.

4. User can navigate the app properly – ensured by buttons having appropriate size, colour, name, page linked to them.

*UI1-1.1 1. Start workout button which takes user to a new page that lets user enter exercises planned for the workout 2. Check progress button which takes user to a page to check either recent progress or monthly progress 3. Check AQI button which activates the product to check the AQI of the environment. 4. Bluetooth connection status button which brings a pop-up if not connected 5. settings button which takes user to the settings page 6. check heartrate button*

*UI1-1.2 The dedicated page for displaying heart rate and saved metrics*

*UI1-1.3 Create a settings button which includes many options including but not limited to user info button which shows user their info by taking them to a different page, change user info button which lets user change their info*

*UI2-2.2 Pop ups happen and give error messages when :*

*1. Not connected via bluetooth.*

*2. User enters invalid info as input*

*3. User leaves a blank space (does not enter input where needed).*

**2. Test Case Designing** – Create detailed test cases, e.g.:

* Valid Registration:
  + Fill all mandatory fields with correct data.
  + Saves data given.
  + Connects to Bluetooth.
  + Disconnects Bluetooth
* Invalid Registration:
  + Leave fields blank or enter invalid email.
  + Tries to use sensor without connecting to Bluetooth
  + Tries to use app without logging in(optional)

**3. Test Execution** – The tester manually performs each step on the website. For instance, after clicking “Settings icon”, the expected result is the settings page where user can change info, add info, register, sign in, sign out(optional).

**4. Defect Reporting** – If any test fails to deliver expected results(write in detail what they are), the defect is logged with screenshots.

**5. Re-Testing & Closure** – Once developers fix issues, the failed cases are re-executed to verify resolution.