CareMate: Personalized Healthcare Companion

Jason Browder

CS-3332 Software Engineering I

Mr. Reggie Haseltine

3/2/2025

A Software Test Plan (STP) is a crucial document that defines the testing strategy for a software product. It ensures that all functionalities are tested, defects are fixed, and quality standards. This paper lays out a practical Software Test Plan for CareMate, covering unit, integration, system, and acceptance testing. The plan aligns with the Software Requirements Specification (SRS) and emphasizes efficient error handling, structured testing in SCRUM Agile sprints, and a mix of white-box and black-box testing techniques.

Developing high-quality software requires a well-structured testing approach to catch issues early and ensure smooth operation. This test plan outlines how testing will be conducted for CareMate ensuring that the software works as expected and meets user needs. It explains the different testing types, who will be responsible for testing, and how defects will be tracked and resolved.

Unit testing focuses on checking individual parts of the software to make sure they work correctly. Developers will use white box testing techniques to validate logic, conditions, and loops (Pressman & Maxim, 2020). To speed up this process, we will use automated unit testing frameworks such as JUnit for Java or PyTest for Python.

Integration testing ensures that different software components work well together. We will use three main approaches (Pressman & Maxim, 2020):

- **Top-Down Testing**: Testing will start with high-level modules, using stubs for lower-level ones.
- **Bottom-Up Testing**: Testing will begin with lower-level components, using drivers to simulate higher-level ones.

- **Continuous Integration (CI)**: Automated integration tests will run daily using tools like Jenkins or GitHub Actions.

System testing will check the full application to ensure it meets all requirements, including functionality, performance, security, and usability (GeeksforGeeks, 2023). Acceptance testing will confirm that the software meets business and user expectations.

- **Alpha Testing**: Internal team members will test the software before release.
- **Beta Testing**: A small group of real users will try the software in a real-world setting before the final launch.

Testing will be fully integrated into SCRUM sprints to align with the agile development process (Pressman & Maxim, 2020). Each sprint will include:

1. **Sprint Planning**: QA engineers will create test cases based on user stories.
2. **Development & Unit Testing**: Developers will run unit tests and fix any defects.
3. **Integration Testing**: QA engineers will verify that modules work together correctly.
4. **Sprint Review & Acceptance Testing**: Stakeholders will test new features to ensure they meet expectations.
5. **Sprint Retrospective**: The team will document lessons learned and plan improvements for future sprints.

Roles and Responsibilities:

- **Developers**: Write unit tests and fix issues.
- **QA Engineers**: Conduct integration, system, and acceptance testing.

- **Product Owners**: Validate that the software meets business needs.

A Requirements Traceability Matrix (RTM) will map software requirements to corresponding test cases, ensuring complete test coverage (GeeksforGeeks, 2023).

To keep track of software defects, we will use a structured bug report template (Software Testing Material, 2023). Each bug report will include:

1. **Bug ID**
2. **Severity & Priority**
3. **Test Case Reference**
4. **Steps to Reproduce**
5. **Expected vs. Actual Results**
6. **Status (Open, In Progress, Fixed, Closed)**

Bug fixes will follow this process:

- **New**: The bug has been reported.
- **Assigned**: It is assigned to a developer.
- **In Progress**: The developer works on a fix.
- **Resolved**: The fix is implemented.
- **Verified**: QA retests the fix.
- **Closed**: The bug is confirmed as fixed.

White-Box testing looks at the internal workings of the software to ensure correctness. It will be used for:

- Code coverage analysis
- Path testing to ensure all conditions are checked
- Boundary testing for edge cases (Pressman & Maxim, 2020).

Black-Box testing tests the software by focusing on inputs and outputs without examining the internal code. It will be used for:

- Functional validation
- User interface (UI) testing
- System and acceptance testing (GeeksforGeeks, 2023).

A strong Software Test Plan is key to delivering reliable software. By using a combination of unit, integration, system, and acceptance testing, this plan ensures that all functional and non-functional requirements are met. Following SCRUM principles, thorough documentation, and continuous testing will help us build a high-quality product that meets user expectations.

References

GeeksforGeeks. (2023). *Types of software testing.*

https://www.geeksforgeeks.org/types-software-testing/?ref=gcse_outind

Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). New York, NY: McGraw Hill.

Purdue Online Writing Lab. (n.d.). General format. Purdue University.

https://owl.purdue.edu/owl/research_and_citation/apa_style/apa_formatting_and_style_guide/general_format.html

Software Testing Material. (2023). *How to write a good bug report.*

https://www.softwaretestingmaterial.com/bug-report-template/