
**THIS PROJECT REPORT HAS BEEN WRITTEN AND SUBMITTED AS PARTIAL FULLFILLMENT FOR THE BACHELOR OF INFORMATION TECHNOLOGY**

**KCA UNIVERSITY**

**FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY**

# SYSTEM TEST PLAN: DISEASE PREDICTION SYSTEM

**Submitted by:**

OMONDI SANTOS OKELLO

19/05036

**Supervised by: MR COLLINS ONDIEK**

**BSc. INFORMATION TECHNOLOGY**

# TABLE OF CONTENT

# CHAPTER 1: INTRODUCTION

## 1.1 Introduction

The primary objective for a test plan is to produce documentation that describes how the tester will verify that the system works as intended. The document should describe what needs to be tested, how it will be tested, and who's responsible for doing so. It contains the test strategy, objectives, schedule, estimation, deliverables, and resources required to perform testing for a software product. Test Plan helps to determine the effort needed to validate the quality of the application under test and serves as a blueprint to conduct software testing activities as a defined process, which is minutely monitored and controlled by the test team.

## 1.2 Goals and objectives

The goals and objectives of this test plan is to measure testing progress and verify that testing activity is consistent with project objectives.

Goals and objectives are grouped into the following categories:

**Functional correctness.**

Validation that the application correctly supports required Hive design application processes and transactions. List all the smart health application processes that the

application is required to support. Also list any standards for which there is required compliance. Some of the objectives under this category includes the following

- Ability to identify a duplicate user that is about to be created by the system.
- Ability to make accurate disease prediction-based symptoms submitted by patient.
- Ability to find the correct doctor based on the doctor's profile that is searched
- Ability for system users to give their feedback on how they fill about the system's experience

**Authorization.**

Verification that actions and data are available only to those users with correct authorization. Here the goals and objectives are to identify key authorization requirements that must be satisfied, including access to functionality and data. In detail they include the following

- Ability to sign in only users that are registered by the system.
- Ability to register new users and determine their access levels.

**Service level.**

This is the verification that the system will support the required service levels of the business. This includes system availability, load, and responsiveness.
**Usability.**

This the validation that the application meets required levels of usability. This include the required training level needed for users to be able to use the system effectively.

### 1.3 Statement of scope

The scope of this test plan is to define what areas the disease prediction system is supposed to be tested what functionalities to mainly focus on, what types of bugs customers are interested in and what features of the system that will not be tested by any means

#### I. Features

Each test contains at least one feature, each feature describes an area of a product, it describes a process of the disease prediction system, particular to a functionality to be tested. The features to be tested here include

- Navigation

This is to test the level of easiness a user has when using the system, how easy he can allocate links and knowing which area of the system they are at a particular time.

- Landing page
  This is the page where user is redirected to when they visit the website. This is to check whether it sufficiently communicates what the system is all about and its uniqueness for users to distinguish the system from other systems.

- Registration and login
  This entails the authentication functionality of the system. This is to check whether if the functionality works accordingly.

- Disease symptoms page
  This is the major functionality of the system. Here the ability of the system to be able to progressively suggest symptoms that are related to previously selected symptoms is tested to ensure it is mostly accurate.

- Doctors page
  This is to check whether doctors' information is accurate based on the doctor's profile and that they can perform their roles.

- Admin Page
  This is a page by which an admin can render add and deletion of patient and doctor accounts

## II. Bugs and severities

The test plan focusses on certain bugs that users of the system may be able to notice while using the system. These include

- Functionality
  This is to check if there is a bug in the functionality of the system. If a a certain function doesn't perform as expected the severity of the bug is dimmed to be critical and needs immediate attention.

- Content
  This is to check whether the system includes all the content that is needed. If there are some content that is missing the severity of the bug is dimmed high. It needs a quick attention but not necessarily an immediate one.

- Visual

This is to check whether the system UI has been created appropriately and that the user experience is great. The latter, its severity is considered as low and doesn't need an immediate attention.

- Usability
  This is to check how complex the website is to use a much sophisticated one that makes its usage difficult is dimmed as high severity and needs to be attended to quickly.

**Out of scope**

The out of scope these are the areas the testing will not cover. For disease prediction system, they include the following

- Legal problems, this is so since I am not a legal advisor and severity of a bug is not determined by legal provisions or standards.
- Problems related to browser extensions, ad-blockers, or virus scanners e.g., blocking certain contents or even execution of apps.
- Setup problems in testing.

### 1.4 Major constraints

Every healthcare system right now is revolved around three principles these include Security, Performance, and quality.

**Security**

To ensure security permission testing is required, and this requires that a detailed suite of auto tests to exclude possibility of regression, only problem with this approach is that it requires many preparations before the exact run. And that time needed for testing increases exponentially with increase of testcase coverage. Another issue is that in healthcare protection of patient's data is very crucial and thus there is no access to the production environment especially to real patients' data. So when we should check a fix for some issue that is actual for production environment only and haven't noticed before for test environments, it's a challenge to investigate steps to reproduce or some specific configurations without access to real examples of patients with that issue.

**Performance**

When the development is just started, it's hard to predict all possible 'bottlenecks' in the system that may occur. The cost of such issues can be very high. Performance and load testing should be introduced as soon as possible to make sure that the system behaves as expected for final users. This is the ideal approach, but it is not practically possible since balancing these testing and developments require a lot of work and creativity to be able to predict future performance and how the system will behave.

**Quality**

The costs of mistakes in healthcare are very high, a small error can be very catastrophic. Thereby a system can't be delivered without a full test coverage to ensure all functionalities are tested. A user is more inclined towards a system that is nice looking, simple, working, and fast processing of requests. This gives much work in testing that include manual testing of new features, regression and smoke testing, design, and development of auto tests, supporting existing tests, creation of test documentation for new feature etc.

# CHAPTER 2: TEST PLAN

## 2.1 Software to be tested

The software to be tested here will be a disease prediction system. This system will be web based, that is able to be accessed from any device that has internet connection.

Name of the website:  Hive design healthcare

## 2.2 Testing strategy

Here I outline a high-level description of the approach I will take in the software testing life cycle that clearly shows the testing objectives of the system. This is to guide to define the test coverage and testing scope to clear the picture of the project at any instance, to ensure a test activity is not missed.

### 2.2.1 Unit testing

This is the first and most important level of testing. It starts from the moment a unit of code is written. Every unit is tested for various scenarios to detect and fix bugs during early stages of software development lifecycle. For unit testing the following strategy will be used

i. Creation of test cases and test data
ii. Creation of scripts to run the test cases wherever applicable
iii. Execution of test cases once the code is ready
iv. Fixing of the bugs of present and retesting of the code
v. Repletion of the test cycle until the confidence level is that there are no significant bugs remaining.

Tests that will be performed during unit testing include.

- Testing the user interfaces to ensure that the inputs produce the expected outputs, and that information is properly flowing into the program unit.
- White boxing testing will be used to ensure the functionalities of the system behave as they are expected to.
- Error handling paths to review if errors are handled properly by them or not.
- Independent paths are tested to see that they are properly executing their task and terminating at the end of the program.
- Local data structures are tested to inquiry if the local data within the module is stored properly or not.

### 2.2.2 Integration testing

This is the step where software modules are integrated logically and tested as a group. Here I will be integrating different components that are interrelated with the aim to exposed defects in their interactions. This is because even though unit testing is done independently defects will still exist. Integration testing is done when all modules code is completed and successfully integrated. For this testing I will follow an incremental approach this is because makes it easier to locate a fault, obtain a prototype earlier, and no time is wasted waiting for all modules to be developed like in big bang approach.
To carry out integration testing the following activities will be followed

i. Design the Test Scenarios, test cases, and create scripts. ii. Executing the test Cases followed by recording the defects. iii. Fixing & re-testing the defects. iv. Steps 3 and 4 are repeated until the completion of Integration is successful.

### 2.2.3 Validation testing

Validation testing is in place to determine if the existing system complies with the system requirements and performs the dedicated functions for which it is designed along with meeting the goals and needs of the organization. In this case it is to test that the disease prediction system complies with requirements and performs its functions correctly in relation to healthcare of its users i.e., business logic. Here is where the critical functionalities of the system are tested for my case the critical functionalities of my system include correct prediction of illness, security to ensure no data is compromised, availability of the system to users, usability to ensure all level of individuals can navigate the system. Validation process will entail the following steps

i. Gather healthcare requirements for validation testing from the end user.
ii. Prepare the business plan and send it for the approval to the onsite/stakeholders involved i.e the partnering hospital for my project.
iii. On approval, begin to write the necessary test cases and send them for approval.
iv. Once approved I begin to complete testing with the required software, environment and send the deliverables as requested by the client.
v. Upon approval of the deliverables, User acceptance testing is done by the client.
vi. After that, the software goes for production.

### 2.2.4 High order testing

High order testing is done because unit testing and integration testing are not always sufficient to identify and locate all the vulnerabilities of a software and that's why there is a need for further testing i.e., high order tests. They are mostly done by third parties to ensure the test findings match with those of the owners of the software. High Order Test ensures the fulfilment of all sorts of requirement, criterion, at each step of the development, to produce a software product of desired quality. High order testing is of various types that include the following.

- **System Testing**

  Herein the focus lies on the integrated system. The main aim behind System Testing is to completely work out and analyze the software to identify the loopholes that can cause errors.

- **Recovery Testing**

  In Recovery Testing the software is forced to fail under various scenarios and in different ways. The idea is to corroborate that the recovery process occurs as expected and that there are no deviations from it.

- **Security Testing**

  These days security is a very big issue. If a software is open to unauthorized and illegal access, then it is bound to cause concern. To substantiate and authenticate the fact that the software is well secured it is necessary to conduct comprehensive Security Testing.

- **Stress Testing**

  The software is now tested at varying frequencies, with abnormal data inputs and haphazard volume to validate its breaking point. The idea is to judge its complete functionality in the light of unpredictable and inconsistent consumer or user behavior.

- **Performance Testing**

  Finally, the software needs to be checked and tested for its range of performance to know its compatibility and functionality.

- **Function Testing**

  This testing is used to locate out the discrepancies, between the actual working and behavior and what was intended, from the user's point of view.

- **Integration Testing**

  Ensures the proper working of the modules, after getting integrated.

- **Acceptance Testing**

  It is used to verify and validate a software product, against the specified requirements and specifications, to ensure that the product is ready for the use.

**2.3 Testing resources and staffing**

Resources that will be used include

i. Laptop
ii. VisualStudio
iii. PostgreSQL

iv. TestCollab

I will be the sole staff for the software testing.

### 2.4 Test work products

### 2.5 Test record keeping

For test record keeping I will be using TestCollab. It is among the most modern test management tool available for your quality assurance needs. It is effortless to onboard another member since its user interface is very friendly and easy to understand. TestCollab also offers in-app live chat support. Hence one can always get support it also offers the following

- Jira Integration, one can post their bugs in Jira from TestCollab and using their Jira plugin you can also see all your Test Cases, test plans, and reports in Jira as well. • Automatic work assignment for multiple testers so you don't have to manually assign test cases to each tester.
- Reuse the same test case across multiple projects, once the test case is updated in a project, it gets updated in all the projects.
- To-do list with due dates. Hence one can be on track on his testing schedule
- TestCollab offers a free option which makes it cheap and affordable for me as a student.

### 2.6 Test metrics

Software Testing Metrics are the quantitative measures used to estimate the progress, quality, productivity, and health of the software testing process. The goal of software testing metrics is to improve the efficiency and effectiveness in the software testing process and to help make better decisions for further testing process by providing reliable data about the testing process. A Metric defines in quantitative terms the degree to which a system, system component, or process possesses a given attribute. The types of software metrics that will be used on this project include the following

- **Process Metrics**
  It is used to improve the efficiency of the process in the SDLC (Software Development Life Cycle).

- **Product Metrics**
  It is used to tackle the quality of the software product.

- **Project Metrics**
  It measures the efficiency of the team working on the project along with the The

life cycle for test metrics is listed below

- **Analysis**
  It is responsible for the identification of metrics as well as the definition.

- **Communicate**
  It helps in explaining the need and significance of metrics to stakeholders and testing team. It educates the testing team about the data points that need to be captured for processing the metric.

- **Evaluation**
  It helps in capturing the needed data. It also verifies the validity of the captured data and calculates the metric value.

- **Reports**
  It develops the report with an effective conclusion. It distributes the reports to the stakeholders, developers, and the testing teams.

**2.7 Testing tools and environment**

We are in an era of automation being used everywhere the increased demand for automation is trending in our software testing industry, as well. And in many software testing communities like uTest, Quora, etc. software testers are urging for various tools that can be helpful in their day-today testing activities Selenium is a good testing framework to perform web application testing across various browsers and platforms like Windows, Mac, and Linux. Selenium helps the testers to write tests in various programming languages like Java, PHP, C#, Python, Groovy, Ruby, and Perl. It offers record and playback features to write tests without learning Selenium IDE. And this makes easily adaptable for usage in my system testing.

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured.

**2.8 Test schedule**

# CHAPTER 3: TEST PROCEDURE

## 3.1 Software to be tested

The software to be tested here will be a disease prediction system. This system will be web based, that is able to be accessed from a mobile device that has internet connection.

Name of the website:  Hive design healthcare

## 3.2. Testing procedure

Unit testing will first be conducted then integration testing, followed by high order testing before the system is released for the production environment.

### 3.2.1. Unit test cases

- **Module user interfaces**

**a. Stubs and/or drivers**

- ✦ User's Authentication
- ✦ User experience

**b. Test cases**

- ✦ User's Authentication: Using verification method to ensure that correct users get a login and deny to others
- ✦ User experience: using user feedback to ensure whether the user experience is good

**c. Purpose of tests**

To ensure that the user interface works as required and gives the output that is required

**d. Expected results**

User can be redirected to the correct page depending on user input on the website page.

- **Functionalities testing a. Stubs and drivers required**
    - ✦ User access to system ✦ Symptom's correctness ✦ User roles
    - ✦ Admin rule

  **b. Test cases**
    - ✦ User access to system: using verification method check if a system user can access services from the website.
    - ✦ Symptom's correctness: using verification method confirm if the system gives the correct disease prediction based on symptoms given
    - ✦ User roles : verify whether users only access roles that are allocated to them only.
    - ✦ Account termination: check whether if a user deletes his or her account that it has been successfully deleted from the system.

  **c. Purpose of test**

  This test is to ensure that a system performs all its functionalities accordingly. **d. Expected results**

  That the system can give positive outcomes that conforms with the systems requirements

- **Error handling path testing**

  **a. Stubs and drivers required**
    - ✦ Error checking

  **b. Test cases**
    - ✦ Error checking: Verify that the system checks for errors and alerts user of the error. Verify that it also keeps a log of the errors

  **c. Purpose of test**
    - ✦ To ensure that a user of the system is made aware of an error when it occurs and that there is a log of these errors to check on which errors are frequently made by user or by the system.

  **d. Expected results**
    - ✦ That a user is alerted when an error occurs and an error that has occurred has been stored

- **Independent paths testing a. Stubs and drivers required**
    - ✦ System tasks

  **b. Test cases**
    - ✦ System tasks: verify that system independent process functions well i.e., login, registration, symptom checking, account deletion, giving feedback etc. can perform their functions from start to finish independently. **c. Purpose of test**
    - ✦ To ensure that all the process in the system is functioning as required.

**d. Expected results**

- ✦ That the system independent process is functioning well according to conformance with requirements.

### 3.2.2 Integration testing

**a. Stubs and drivers required**

- ✦ Login and registration

**b. Test cases and purpose**

- ✦ Login and registration: check whether a user is being redirected to login page if registered and to registered page if not registered
- ✦ This is to ensure a user is on the right page based on their registration status with the system.

**c. Expected results**

- ✦ That a user is redirected to the correct page based on their status on the website

**a. Stubs and drivers required**

- ✦ Symptom's progression

**b. Test cases and purpose**

- ✦ Symptom's progression: verify that a symptom a user selects progresses to a symptom that is related to the previously selected symptom
- ✦ The purpose of this is to ensure that a user symptom is only related to other similar symptoms associated with the illness they may be suffering from

**c. Expected results**

That a symptom selected prompts progression to only related symptom to the illness a patient may be suffering from

### 3.2.3 Validation testing

#### 3.2.3.1 Testing procedure for validation

The validation procedure involves the following steps

- Gather healthcare requirements for validation testing from the end user.
- Prepare the business plan and send it for the approval to the onsite/stakeholders involved i.e the partnering hospital for my project.
- On approval, begin to write the necessary test cases and send them for approval.
- Once approved I begin to complete testing with the required software, environment and send the deliverables as requested by the client.
- Upon approval of the deliverables, User acceptance testing is done by the client.
- After that, the software goes for production.

#### 3.2.3.2 Expected results

- The expected results here is that at the end of the test the acceptable requirements that the system should have been extracted from the user.
- The system has been adjusted to a user acceptable level approved by users.
- The system can be released to production.

#### 3.2.3.3 Pass/Fail criterion for all validation tests

User acceptance is the ultimate criterion for validation testing. If a user accepts the system, the validation testing is considered a pass and it is a failure when a user rejects the system.

### 3.2.4 High-order testing

#### 3.2.4.1 Recovery testing

**a. Test procedure**

- Here the software is forced to fail under various scenarios and in different ways.

**b. Test cases**

- Denial of service attack: this can be one of test cases where the system is forced to fail through an attack that prevents users of the website from getting services by sending too much traffic to the website or corrupting its files with viruses A recovery test can be performed to check how the system will recover. The system should have a back up system incase this scenario occurs. This is to test the capability of the system being able to restore to itself to its initial state before the attack occurred. Back up types i.e., offline back up, and online back up to test which of them is more effective on the restoration of a site to its initial state after it has failed.
- Sudden shutting down of the website while in use: when user is using the website sudden shutting down of the app while it is receiving data by either disconnecting from the internet or switching off the device and then after some time turning it on or restoring the internet connection. The ability of the website to continue receiving data from the point where the connection was disconnected is evaluated.

**c. Purpose**

- To evaluate whether the system meets the requirement that the website can restore to its initial state even after a catastrophic failure has occurred

**d. Pass/fail criteria**

- the pass and fail criteria is determined by the time taken for the website to recover and to the degree at which the state of the system is restored to. Shorter recovery period is dimmed to be more successful than longer ones and systems that restore to a state closer to the initial state before an attack is considered a success and the latter a failure.

#### 3.2.4.2 Security testing

Security testing uncovers vulnerabilities, threats, risks in a software application and prevents malicious attacks from intruders.

**a. Test procedure**

We will check for different types of security testing. These will include the following

- **Vulnerability Scanning**: This is done through automated software to scan a system against known vulnerability signatures.
- **Security Scanning:** It involves identifying network and system weaknesses, and later provides solutions for reducing these risks. This scanning can be performed for both Manual and Automated scanning.
- **Penetration testing**: This kind of testing simulates an attack from a malicious hacker. This testing involves analysis of a particular system to check for potential vulnerabilities to an external hacking attempt.
- **Risk Assessment:** This testing involves analysis of security risks observed in the organization. Risks are classified as Low, Medium and High. This testing recommends controls and measures to reduce the risk.

**b. Test cases**

- A password should be in encrypted format: for this test scenario the test should verify that encryption used for hiding user private information like passwords cannot be decrypted by unauthorized user.
- Application or System should not allow invalid users: This test scenario is to verify that the system is able to detect invalid users from the valid ones and only authenticates valid users into the system
- Check cookies and session time for application: this is to verify that the website is able to remember those who visit the website and keep track of their activities for possible suggestions

**c. Purpose**

- The purpose of Security Tests is to identify all possible loopholes and weaknesses of the software system which might result in a loss of information, revenue, repute at the hands of the employees or outsiders of the Organization **d. Pass/fail criteria**
- A system that is able to protect its data from unauthorized access and as well protect itself from unauthorized access is considered a success and the contrary is a failure

### 3.2.4.3 Stress testing

Stress testing verifies stability & reliability of software application.

**a. Test procedure**

- Flooding the website with too much traffic to observe its reaction **b. Test cases**
- When the website receives sudden high traffic: this is a scenario whereby for some reason the website receives high number of traffics than usual. To verify that the system can accommodate these abnormal traffic spikes with failing. **c. Purpose**

✦ The goal of Stress testing is measuring software on its robustness and error handling capabilities under extremely heavy load conditions and ensuring that software doesn't crash under crunch situations.

**d. Pass/fail criteria**

✦ A system that can withstand all the stress being put at it considered a success and the later a failure.

**3.2.4.4 Performance testing**

Performance testing is used for testing the speed, response time, stability, reliability, scalability, and resource usage of a software application under particular workload.

**a. Test procedure**

Various types of performance testing are conducted to determine the performance of the system for the stakeholders

- ✦ **Load testing –** checks the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
- ✦ **Stress testing –** involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
- ✦ **Endurance testing –** is done to make sure the software can handle the expected load over a long period of time.
- ✦ **Spike testing –** tests the software's reaction to sudden large spikes in the load generated by users.
- ✦ **Volume testing** – Under Volume Testing large no. of. Data is populated in a database and the overall software system's behavior is monitored. The objective is to check software application's performance under varying database volumes. ✦ **Scalability testing** – The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.

**b. Test cases**

- ✦ Speed – this test scenario is to determine whether the application responds quickly
- ✦ Scalability – This test scenario is to determine maximum user load the software application can handle.
- ✦ Stability – Determines if the application is stable under varying loads

**c. Purpose**

✦ The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application. Performance Testing is done to provide stakeholders with information about their application regarding speed,

stability, and scalability. More importantly, Performance Testing uncovers what needs to be improved before the product goes to market.

**d. Pass/fail criteria**

- ✦ A system that satisfies all the types of performance testing conducted is considered a pass and the latter is a failure.

### 3.2.4.5 Alpha/beta testing

**Alpha Testing** is a type of acceptance testing; performed to identify all possible issues and bugs before releasing the final product to the end users. **Beta Testing** is performed by "real users" of the software application in "real environment"

**a. Test procedure**

- ✦ Conducting a alpha test following all the phases of alpha testing and finally conducting the beta testing

**b. Test cases c. Purpose**

- ✦ No matter how many tests you perform, how many bugs you kill, your software is useless if your end-users do not like it. Beta testing helps provide authentic feedback of your software from real users. Alpha testing helps simulate real-time user environment before the software is sent for Beta Testing and helps shape a stable software candidate eligible for Beta Tests.

**d. Pass/fail criteria**

- ✦ A system that is accepted by the users is considered a success.

## 3.3 Testing resources and staffing

Resources that will be used include

i. Laptop. This will be the hardware to be used for the testing purposes
ii. Browser: this is the testing software environment that will be used
iii. Users : these are the personnel's that will be involved in the testing purposes of the website.
iv. TestCollab:: used to manage testing activities and keeping record of test logs
v. Selenium IDE: this is an IDE for that will be used for test automation

I will be the sole staff for the software testing.

## 3.4 Test work products

Test work products include:

- Detection of bugs vulnerabilities and correcting them to make the system better
- specificity of test documentation

- Review testing of the system sing the appropriate techniques and by the appropriate participants and stakeholders
- Establishing and monitoring metrics that monitor the quality of the software tested

.