Master Test Plan

**[Medical Billing Application]**

Document Version: [Revision 1]

Application Version: Beta1

Table of Contents

[1. Introduction 3](#_Toc476292744)

[1.1. Purpose & Scope 3](#_Toc476292745)

[2. Test Strategy 4](#_Toc476292746)

[2.1. Test Selection 4](#_Toc476292747)

[3. Test Plan Metrics 6](#_Toc476292748)

[3.1. Functionality Testing Per Week 6](#_Toc476292749)

[3.2. Aggregated Functionalities 7](#_Toc476292750)

[4. Exit Criteria 7](#_Toc476292751)

[5. Testing Members 8](#_Toc476292752)

[6. Testing Equipment 9](#_Toc476292753)

[7. Test Cases 10](#_Toc476292754)

[7.1. Client to Server Network connectivity 10](#_Toc476292755)

[7.2. Client to Server Login 10](#_Toc476292756)

[7.3. Client View – Build Claim 11](#_Toc476292757)

[7.4. Client View – Submit Claim 12](#_Toc476292758)

[7.5. Client View – Claim Reconciliation 13](#_Toc476292759)

[7.6. Usability 13](#_Toc476292760)

[7.7. High Traffic Reliability 14](#_Toc476292761)

[7.8. 14](#_Toc476292762)

[8. Defect Tracking 15](#_Toc476292763)

[8.1. Functionality Defect Tracking 15](#_Toc476292764)

[8.1.1. Defects Documented by Tester 15](#_Toc476292765)

[8.2. Requirements Phase Defect Recording 16](#_Toc476292766)

[8.3. Logical Design Phase Defect Recording 16](#_Toc476292767)

[8.4. Physical Design Phase Defect Recording 17](#_Toc476292768)

[8.5. Program Unit Design Phase Defect Recording 18](#_Toc476292769)

[8.6. Coding Phase Defect Recording 18](#_Toc476292770)

[9. Interview Agenda 19](#_Toc476292771)

[10. Test Schedule 19](#_Toc476292772)

[10.1. Project Milestone 20](#_Toc476292773)

[10.2. Test Schedules 20](#_Toc476292774)

# Introduction

This software system will be a Medical Billing System (MBS) to be used by a local hospital. The MBS will be designed to submit, process and or reconcile claims made by users or patients. Alongside these actions the MBS will also process billings and payments as well as update user account balances and issue reports on request.

The software will be a web-based application to be utilized by the hospital’s billing team in conjunction with the pre-existing Electronic Medical Record (EMR) system. The purpose of the software is to speed up the insurance adjudication process. This will increase a faster turnaround and a greater stream of revenue.

## Purpose & Scope

The purpose of this document is to present to stakeholders a detailed description of the Medical Billing System test plan. It will explain the purpose and features of the system, the system’s interfaces, along with the constraints and requirements for operation and response to system users.

The test plan will cover the testing of the interaction between the GUI (Client) and the different Server interfaces. In addition, the testing plan will cover the response of the different functionalities the applications will provide to the user.

The areas that will be tested will be:

* Client-to-Server Network connectivity
* Client-to-Server Login
* Client view – Build Claim
* Client view – Submit Claim
* Client view – Payment Reconciliation
* Usability
* High Traffic Reliability

# Test Strategy

## Test Selection

Testing will take place throughout the development process of the application. Encouraging the development team to meet throughout the project for weekly walkthroughs and review of the code once they have been assigned to a unit of the project. This document will focus on these types of tests:

Need to test Functionality (Correctness, Reliability, Usability, Integrity), Engineering (Efficiency, Testability, Documentation, Structure), and Adaptability (Flexibility, Reusability, Maintainability).

| **Test** | **Applicable?** | **Rationale for omitting test level** |
| --- | --- | --- |
| Acceptance | *Yes*  *No* | *No omitting Acceptance test. The first acceptance test will be performed at the end of the first spiral* |
| Integration Testing | *Yes*  *No* | *No omitting the Integration Testing. The integration testing will take place weekly to ensure every database and graphical user interface is working in accordance to the Functional Requirements.* |
| Graphic User Interface | *Yes*  *No* | *No omitting GUI testing. Graphic user interface will be performed at the end of the first spiral* |
| Regression | *Yes*  *No* | *No omitting Regression testing. Regression testing will be performed after the results from the testing.* |
| Stress Testing | *Yes*  *No* | *No omitting Stress Testing. The system will be stress test before users test the application.* |
| Unit Testing | *Yes*  *No* | *No omitting Unit Testing. Unit testing insert, delete, update, and select statements into the different Database Interface entries.* |

* 1. **Test Characteristics**

This section establishes the level of commitment for each test. Defined by the different iterations, entry criteria, and exit criteria.

The iterations will outline the degree of completion for each test.

***Iteration 1:*** *A comprehensive iteration that includes the execution of all the test cases.*

***Iteration 2:*** *Includes all the test cases that failed during iteration 1 and have been corrected.*

***Iteration 3:*** *The final iteration and should result in no or minimal defects being identified.*

Entry criteria will showcase the type of data available to complete the test. Also, what initiated the test and if a proper environment is available.

Exit criteria will define the outcome of the test. The level of testing and its status. Results will be documented for the next implementation spiral.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Level** | **Owner** | **Planned Iterations** | **Entry Criteria** | **Exit Criteria** |
| Unit Testing | User | 3 | Completed and reviewed test cases / test scripts. | All Items in scope tested. First implementation spiral only requires defect reports and documentation. |
| Integration Testing |  | 3 | Completed and reviewed test cases / test scripts. | All Items in scope tested. First implementation spiral only requires defect reports and documentation. |
| GUI | User | 2 | Access rights for the testers were established. | All Items in scope tested. First implementation spiral only requires defect reports and documentation. |
| Performance Testing | User | 1 | Testing environment established. Adequate test data available. | All Items in scope tested. First implementation spiral only requires defect reports and documentation. |

* 1. **Test Deliverables**

The items that will be delivered upon completion of testing are:

* Testing result Metrics.
* Common bug and troubleshooting results.
* Performance results.
* Customer satisfaction

# Interview Agenda

|  |  |
| --- | --- |
| I. | Introductions |
| II. | Project Overview |
| III. | Project Objectives |
| IV. | Project Status |
| V. | Project Plans |
| VI. | Development Methodology |
| VII. | High-Level Requirements |
| VIII. | Project Risks Issues |
| IX. | Summary |

# Test Schedule

The following project milestone breakdown is the format that should be used to track the progress of a project. It is imperative that the test schedule is presented to all the team members that will take part in the application development from the very beginning of the project. This will ensure a successful implementation project.

### Project Milestone

Overview of the project due dates. Ideally there should be a gap of between 15 days between each project milestone. This is only a model of the proper documentation to track the different due dates within the testing and development of the application.

|  |  |
| --- | --- |
| Project Milestone | Due Date |
| Sponsorship approval |  |
| First approval available |  |
| Project test plan |  |
| Test development complete |  |
| Test execution begins |  |
| Final spiral test summary report published |  |
| System ship date |  |

### Test Schedules

Detail breakdown of the test due dates. There should be a gap of between 7 to 10 days between each subsection.

|  |  |  |  |
| --- | --- | --- | --- |
| Test Step | Begin Date | End Date | Responsible Staff Member |
| First Spiral  Information Gathering | | | | |
| Prepare for Interview |  |  |  |
| Conduct Interview |  |  |  |
| Summarize findings |  |  |  |
| *Test Planning* | | | | |
| Build Test Plan |  |  |  |
| Define the metric objectives |  |  |  |
| Review/approve plan |  |  |  |
| *Test Case Design* | | | |
| Design function tests |  |  |  |
| Design GUI tests |  |  |  |
| *Define the system/acceptance* | | | |
| Tests |  |  |  |
| Review/approve design |  |  |  |
| *Test development* | | | |
| Develop test scripts |  |  |  |
| Review/approve test development |  |  |  |
| *Test execution/evaluation* | | | | |
| Setup and testing |  |  |  |
| Evaluation |  |  |  |
| *Prepare for the next spiral* | | | |
| Refine the tests |  |  |  |
| Reassess team, procedures, and test environment |  |  |  |
| Publish interim report |  |  |  |
|  |  |  |  |
| *Last spiral* | | | |
| *Test execution/evaluation* | | | |
| Setup and testing |  |  |  |
| Evaluation |  |  |  |
|  |  |  |  |
| *Conduct System Testing* | | | |
| Complete system test plan |  |  |  |
| Complete system test cases |  |  |  |
| Review/approve system tests |  |  |  |
| Execute the system tests |  |  |  |
| *Conduct Acceptance Testing* | | | | |
| Complete acceptance test plan |  |  |  |
| Complete acceptance test cases |  |  |  |
| *Review/approve acceptance* | | | | |
| Test plan |  |  |  |
| Execute the acceptance tests |  |  |  |
| *Summarize/report spiral test results* | | | | |
| Perform data reduction |  |  |  |
| Prepare final test report |  |  |  |
| *Review/Approve the Final* | | | | |
| Test report |  |  |  |

# Test Plan Metrics

## Functionality Testing Per Week

The metric will portray the amount of testing per week and the progress being made.

## Aggregated Functionalities

Series 3 will define the total number of tests for the development of the application. On the other hand, series 1 and 2 will define the different test categories. This below is only a prototype but the end goal is the aggregation of the different series to total Series 3.

# Exit Criteria

Testing will accomplish the following:

1. Find any bugs and reveal methods to troubleshoot the bugs.
2. Functionality meets 85% of user expectations.
3. Performance has met 95% of expectations.
4. Less than 10% of tests have failed.
5. At the end of the spiral, users and developers will create new functionality to be implemented.

# Testing Members

The testing team will ensure all functionalities and quality requirements have been met. The roles of the different test members have been defined below.

|  |  |  |
| --- | --- | --- |
| **Role/ Group** | **Responsibilities** | **Name** |
| *Project Manager* | *Ensures a successful testing environment. The project manager will provide:*   * *Design and Monitoring of the Project* * *Technical guidance.* * *Resource management.* * *Reporting* |  |
| *Billing Team Manager* | *Will provide guidance and business knowledge of the billing workflow. Also, the billing team manager will provide guidance to the test team of the different functionalities to test.* |  |
| *Test Team* | *Executes the tests. Once the test have been executed, the user will ensure to log the results of the test.* |  |
| *QA Team* | *QA Team will perform the system and integration testing.*   * *Ensure the system is tested for performance.* * *Unit testing the code.* * *Integration testing the code.* * *Provide recommendations for any defect discovered.* |  |
| *Database Administrator* | *Database manager will assess the data being stored and extracted from the database. In addition, it will ensure integration of the database is not violated.* |  |
| *Other* | *Suggestions for other users.* |  |

# Testing Equipment

Following equipment will be used for testing:

* Pylot will be used to test the performance and scalability of the application.
* Reliable internet
* Over 8 GB of RAM
* 10/100/1000 Network. Ensuring the Ethernet speeds has the appropriate Mbps download times.

# Test Cases

### Client to Server Network connectivity

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Type | Case Number | Name | Description | Expected Results | Phase | Pass/Fail |
| Integration | 1 | Client not within Network | Verify that the application is only accessible within the network. | The client should be denied access if not within the network. |  |  |
| Integration | 2 | Client within Network | Allow client to access the application when within Network | Client will be notified that they accessed the network successfully. |  |  |
| Integration | 3 | Browser Compatibility | Testing the application is successfully accessed using Chrome | There should be not issues accessing the application using Chrome. |  |  |

### Client to Server Login

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Type | Case Number | Name | Description | Expected Results | Phase | Pass/Fail |
| Integration | 4 | Client Logged in Successfully | Client has been found successfully in the database. | Client is informed that they are confirmed to access the application. |  |  |
| Integration | 5 | Client could not log in successfully | Client is not found in the Database | Client is informed that they do not have access to the Database. |  |  |

### Client View – Build Claim

The page will be utilized create a 1500 medical claim. During the entry of every field, these will be validated against the ANSI standards built in the application.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Type | Case Number | Name | Description | Expected Results | Phase | Pass/Fail |
| GUI | 6 | Client accessing the Build Claim Function | Ensure the client is able to see the claim building page. | Client will be presented with a page where all the necessary 1500 Claim entries can be made. |  |  |
| GUI | 7 | Submit Claim – Build Claim | The submit claim functionality enables the user to send their claims electronically |  |  |  |
| GUI | 8 | Pend Claim – Build Claim | User will be given the option to save the claim t work on it later. |  |  |  |
| GUI | 9 | Warrior Handoff during battle | Battling warriors should iterate with a one person attack at a time method. |  |  |  |

### Client View – Submit Claim

Claims will be loaded from the EMR system and will be queued from the Build Claim page.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Type | Case Number | Name | Description | Expected Results | Phase | Pass/Fail |
| GUI | 10 | View Claim | Claims in the Database will be accessed using the Submit claims page. | All Claims in the database queue will be displayed in the Submit Claim page. |  |  |
| GUI | 11 | Submit Claim | The submit claim functionality enables the user to send their claims electronically | Page will validate all fields to submit a successful 1500 medical claim to the different payers. |  |  |

### Client View – Claim Reconciliation

The page will be used to view all responses to the medical claims submitted. Here we can view the rejections per payor and the accepted medical claims.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Type | Case Number | Name | Description | Expected Results | Phase | Pass/Fail |
| GUI | 12 | View Accepted Claims | All medical claims that have been accepted for payment will be displayed. | The page will bring up all the medical claims that will receive a payment by the payer. |  |  |
| GUI | 13 | View Rejected Claims | The medical claims that have been rejected will be visible. | Rejected claims in the database will be shown by the page when selected. |  |  |

### Usability

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Type | Case Number | Name | Description | Expected Results | Phase | Pass/Fail |
| GUI/Acceptance Test | 9 | Tab from field to field – Build Claims | Ability of user to Tab through the different fields to build a medical claims | User should be able to tab through the different tabs in order form left to right, top to bottom. |  |  |
| GUI/Acceptance Test | 10 | Submit Claim Shortcut – Build Claims | User has the option to submit their medical claim using shift + enter | Page will inform the user if the claim was submitted successfully or field are missing. |  |  |

### High Traffic Reliability

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test Type | Case Number | Name | Description | Expected Results | Phase | Pass/Fail |
| Unit Testing | 11 | Insert Claim | Ensuring the medical claims are inserted successfully into the Database. | Making sure the Medical claim and all components of the 1500 medical claim live in the Database. |  |  |
| Integration | 12 | Bandwidth Dependency | Test file size transmission between Server/Network | Seconds of the transmission of the file sizes between Server/Networks. |  |  |
| Stress | 13 | Horizontal Scalability | Ensuring the servers are integrated seamlessly when the bandwidth is reaching the max size. | New servers are added to the Application when the volume of users increases. |  |  |

### 

# Defect Tracking

The different defects will be tracked by the different aspects of the Project. All the entries will be evaluated in the next Project Interview.

### Functionality Defect Tracking

Format of the defect tracking per functionality. This is only a model of the proper documentation to track the different defects within the application.

|  |  |  |
| --- | --- | --- |
| *Function* | *Number of Defects* | *Percentage of Total* |
| Build Claim | | |
| Create Claim |  |  |
| Submit Claim |  |  |
| Claim Queue | | |
| Edit Claim |  |  |
| Submit Claim |  |  |
| Billing | | |
| Payment Reconciliation |  |  |
| Post Payment |  |  |

### Defects Documented by Tester

Please ensure the names of the testers match the test names in section 5. This is only a model of the proper documentation to track the different defects per user within the application.

|  |  |  |
| --- | --- | --- |
| *Tester* | *Number of Defects* | *Percent of Total* |
| Sample Name |  |  |
| Sample Name 2 |  |  |
| Sample Name 3 |  |  |

### Requirements Phase Defect Recording

The following overview of the defects are recorded after the requirements review. This is only a model of the proper documentation to track the different defects per user within the application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Defect Category* | *Missing* | *Wrong* | *Extra* | *Total* |
| Operating rules (or information) are inadequate or partially missing |  |  |  |  |
| Performance Criteria (or information) are inadequate or partially missing |  |  |  |  |
| Environment information is inadequate or partially missing |  |  |  |  |
| System mission information is inadequate or partially missing |  |  |  |  |
| Requirements are incompatible |  |  |  |  |
| Requirements are incomplete |  |  |  |  |
| Requirements are missing |  |  |  |  |
| Requirements are incomplete |  |  |  |  |
| The accuracy specified does not conform to actual need |  |  |  |  |
| The data environment is inadequately described |  |  |  |  |

### Logical Design Phase Defect Recording

The defects below will be recorded at the conclusion of the logical design review. This is only a model of the proper documentation to track the different defects per user within the application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Defect Category* | *Missing* | *Wrong* | *Extra* | *Total* |
| The data has not been adequately defined |  |  |  |  |
| Entity definition is incomplete |  |  |  |  |
| Entity cardinality is incorrect |  |  |  |  |
| Entity attribute is incomplete |  |  |  |  |
| Normalization is violated |  |  |  |  |
| Incorrect primary key |  |  |  |  |
| Incorrect foreign key |  |  |  |  |
| Incorrect compound key |  |  |  |  |
| Incorrect entity subtype |  |  |  |  |
| The process has not been adequately defined |  |  |  |  |

### Physical Design Phase Defect Recording

The defects below will be based on the static and semantic errors founds. These will be categorized during the physical design phase review. Any defects found, will be presented to the developers that designed the logical phase. This is only a model of the proper documentation to track the different defects per user within the application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Defect Category* | *Missing* | *Wrong* | *Extra* | *Total* |
| Logic or sequencing is erroneous |  |  |  |  |
| Processing is inaccurate |  |  |  |  |
| Routine does not input or output required parameters |  |  |  |  |
| Routine does not accept all data within the allowable range |  |  |  |  |
| Limit and validity checks are made on input data |  |  |  |  |
| Recovery procedures are not implemented or are not adequate |  |  |  |  |
| Required processing is missing or inadequate |  |  |  |  |
| Values are erroneous or ambiguous |  |  |  |  |
| Data storage is erroneous or inadequate |  |  |  |  |
| Variables are missing |  |  |  |  |

### Program Unit Design Phase Defect Recording

Semantic errors involving information and logical control flow should be recorded in the table below. This should cover all the defects discovered during the program unit design phase review. This is only a model of the proper documentation to track the different defects per user within the application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Defect Category* | *Missing* | *Wrong* | *Extra* | *Total* |
| Is the if-then-else construct used incorrectly? |  |  |  |  |
| Is the do-while construct used incorrectly? |  |  |  |  |
| Is the do-until construct used incorrectly? |  |  |  |  |
| Is the case construct used incorrectly? |  |  |  |  |
| Are there infinite loops? |  |  |  |  |
| Is it proper program? |  |  |  |  |
| Are there go-to statements? |  |  |  |  |
| Is the program readable? |  |  |  |  |
| Is the program efficient? |  |  |  |  |
| Does the case construct contain all the conditions? |  |  |  |  |

### Coding Phase Defect Recording

All the defects discovered during the coding phase should be recorded below in order to correct them in the next iteration of the testing phase. These defects will be uncovered during the code review and potential walkthrough. This is only a model of the proper documentation to track the different defects per user within the application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Defect Category* | *Missing* | *Wrong* | *Extra* | *Total* |
| Decision logic or sequencing is erroneous or inadequate |  |  |  |  |
| Arithmetic computations are erroneous or inadequate |  |  |  |  |
| Branching is erroneous |  |  |  |  |
| Branching or other testing is performed incorrectly |  |  |  |  |
| There are undefined loop terminations |  |  |  |  |
| Programming language rules are violated |  |  |  |  |
| Programming standards are violated |  |  |  |  |
| The programmer misinterprets language constructs |  |  |  |  |
| Typographical errors exist |  |  |  |  |
| Main storage allocations errors exist |  |  |  |  |