**Software Requirements Specification**

**Medical Billing System**

**Table of Contents**

1.0. Purpose of the system 2

1.1. Introduction 2

1.2. Scope 2

2.0 Overall description 3

2.1 Product Perspective 3

2.2 Product Functions 3

2.3 User Classes and Characteristics 4

2.4 Operating Environment 4

2.5 Design and Implementation Constraints 4

2.6 User Documentation 5

2.7 Assumptions and Dependencies 5

3.0. Analysis model for our project 6

4. Design 6

4.1 Introduction to UML 6

4.2 System design 7

4.3 Use case Diagram 8

4.4 Use case Diagram for MBS 9

4.5 Detailed Use cases 11

4.5.1. Usecase1: functionalities of Doctor’s office: 11

4.5 .2 Usecase2: functionalities of Claim’s Team: 11

4.5.3 Usecase3: functionalities of Insurance Companies: 11

4.5.4 Usecase4: functionalities of Billing Team: 11

4.5.5 Usecase5: functionalities of Management/Executive Team: 11

4.6 System Evolution 12

5. Other Nonfunctional Requirements 13

5.1     Performance Requirements 13

5.2 Safety Requirements 13

5.3 Security Requirements 13

5.4 Software Quality Attributes 14

5.5 Business Rules 15

6 Testing 15

6.1 Testing Overview 15

6.2 Performance Testing 16

6.3 Security Testing 16

6.3.1 Lessons Learned 16

6.4 Compatibility Testing 17

6.5 Test Case Scenarios 17

# 1.0. Purpose of the system

## 1.1. Introduction

The purpose of this document is to present to stakeholders a detailed description of the Medical Billing System software. 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.

## 1.2. Scope

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.

# 2.0 Overall description

## 2.1 Product Perspective

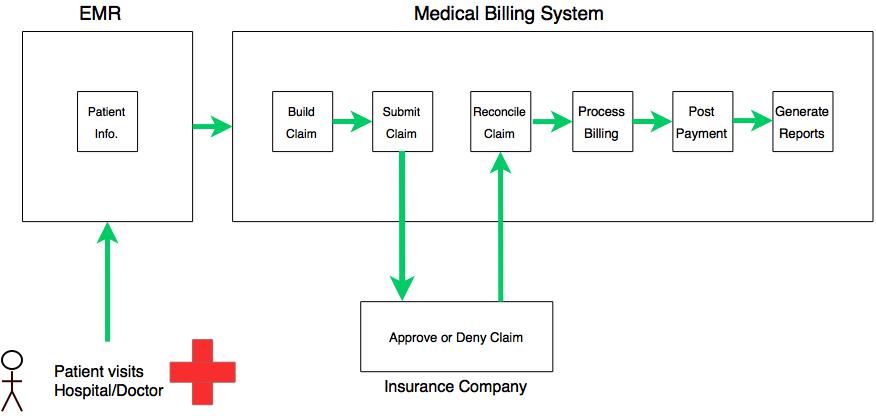
The Medical Billing Software replaces the outdated practice of manual data entry, file running, and paper filing when a doctor’s office files a claim for services performed to an insurance company. The software will interface with the EMR sub-system. The EMR software is already present in most hospitals or medical institutions, and is responsible for processing patient visit information. The Medical Billing Software (BMS) performs these tasks electronically, enhancing production, simplifying data and claims tracking, as well populating patient claims quickly and efficiently, as seen in Figure 1 below.

Figure 1: Diagram of Medical Billing System

## 2.2 Product Functions

* EMR Feed
* Build Medical Claim
* Submit Medical Claim
* Medical Claim Reconciliation
* Billing Consolidation
* Post Medical Claim Payment
* Reporting

## 2.3 User Classes and Characteristics

The MBS is designed to automate much of the billing and claims process as possible, with minimal interaction as possible from the three user classes.

The claims team’s use of the software will be purely data entry. Efficient typing skills are a major requirement in this user class.

The billing staff will be the sole users of the software and must be adept with technical skills and must maintain a privileged level of security and must have a high level of education or much experience.

The Management team is responsible for overseeing operations and ensuring the system and staff runs smoothly while using the software to gain access to system reports of claim metrics, success and failure rates, and access authorizations.

## 2.4 Operating Environment

As a web-based application, the Internet is needed as a platform. Hardware to run the system is not needed locally as SaaS may be implemented via Amazon Web Services (AWS) or other cloud based services.

User Class teams may use thin clients, PCs, or any flavor of computing platform to access the web application, provided a link to the Internet.

All patient information and records will be stored along side the system in the cloud with redundant backup solutions to ensure data integrity.

## 2.5 Design and Implementation Constraints

The development team will must face the following limitations and constraints:

Cloud based services such as IaaS or SaaS are not hosted locally, rather they are hosted on the provider’s servers. Development and Management teams must ensure the terms of service are appropriate for optimal runtime. Choosing the location of servers based on geographic deployment is important. Choosing cloud based services with servers closest to the area of deployment may be helpful in he event some urgent maintenance is needed, rather than if the servers were in a more adverse area or time zone where on-site support may not be as responsive.

Other constraints may involve data security, and staff access and termination policies be implemented. The software handles confidential patient information that may have legal ramifications if such information were obtained by dubious means. Access monitoring should be heavily incorporated throughout the software.

Application constraints may also be an issue. Users must be taught to use an Internet browser with the least vulnerability i.e. Firefox, Google chrome, ad-block plus plugins.

The development team will serve as the support team and must maintain the software after deployment. Adequate access to support must be provided, and easily accessible to the user(s).

## 2.6 User Documentation

User documentation shall be provided electronically along with the software. Users will fine an attached user manual, tutorials database and on-line help in the form of a Help tab from within the software. The MBS development team will service as supplemental support in the event the provided documentation components are not sufficient.

## 2.7 Assumptions and Dependencies

Assumed tools and dependents to run the software successfully are listed:

* Basic computer PC/Mac/Linux capable of sustaining a stable Virtual Machine
* Active internet connection
* Access to an internet browser
* Active ERD feed system

All tools and systems must be present and accessible to run the software successfully. If one or more components are not accessible the software will fail to run.

# 3.0. Analysis model for our project

Software process is a framework for the tasks that are required to build high-quality software. Software engineers and their managers adapt the process to their needs and then follow it. As it provides stability, one can control the software development. Processes that you adopt depend on the software you’re building.

# 4. Design

System design is the process of applying various techniques and principles of defining a system in sufficient detail to permit its physical realization. Software design is the kernel of the software engineering process. Once the software requirements have been analyzed and specified, the design is the first activity.

## 4.1 Introduction to UML

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + User Model View
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + Structural model view
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* Behavioral Model View

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* Implementation Model View

In this the structural and behavioral as parts of the system are represented as they are to be built.

* Environmental Model View

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

* UML Analysis modeling, this focuses on the user model and structural model views of the system.
* UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

## 4.2 System design

**Module Description:**

**1.EMR**

This is an external module required for data entry into the MBS. This system is in place in most hospitals and medical institutions as method of checking in patients and tracking patient information and medical records. This system is serviced either in-house or by a third-party.

**2.** **Build Medical Claim**

This module is used by a member of or solely by the Claims team. Duties include data entry based on medical records and patient data received by the EMR module during or after a medical service is performed or a patient visit is completed. An electronic medical claim is produced from this module.

**3**. **Submit Medical Claim**

This module is an extension of the previous. After all data has been confirmed and the electronic medical claim is validated, it is then ready to be submitted via the Electronic Data Interchange (EDI), which is the electronic transmission of medical claims. These claims are to be submitted to the patient’s Insurance Company.

**4. Reconcile Medical Claim**

Upon return of the medical claim the Billing staff must reconcile the claim based on acceptance or rejection from the Insurance Company. If accepted, the process proceeds. If rejected, Billing staff must reconcile service charges with patient and doctor’s office to proceed.

**5. Consolidate Billing**

Payment amount will be generated in this module based on full medical claim or partial medical claim data. Such factors that will adjust the results of this module include: Full patient responsibility, CONALL payment adjustment, Doctor’s office write-off etc.

**6. Post Payments**

This module is responsible for posting payments for claims, processing payments, along with tracking and updating remaining payments and balance.

**7. Generate Reports**

The final module is responsible for producing reports based on claims filing success rate and failure rate. Metrics of software use and details of successes and failures are also generated and provided to Management and Executive teams for review.

## 4.3 Use case Diagram

Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view. Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

## 4.4 Use case Diagram for MBS

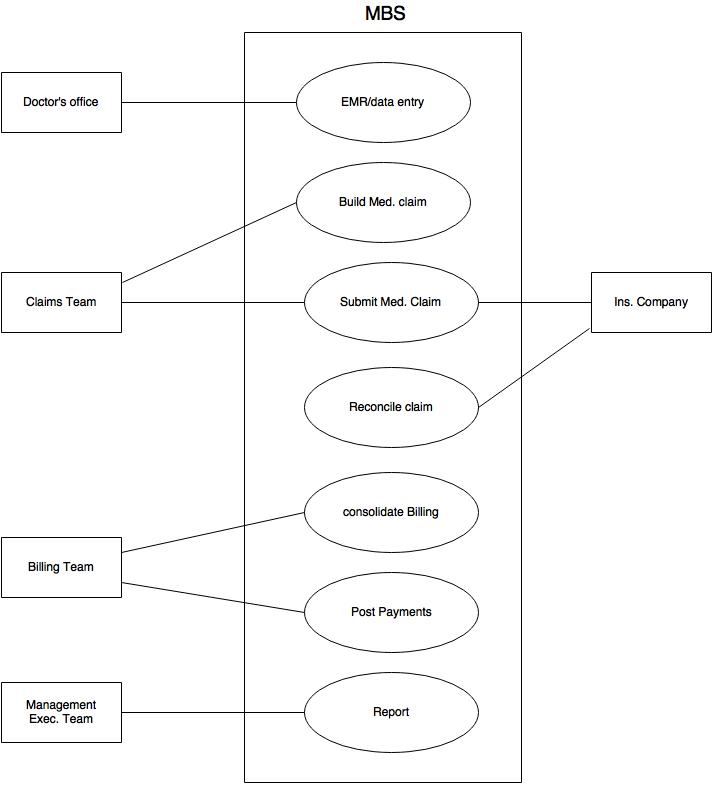


Figure 2: UML Use Case Diagram

## 4.5 Detailed Use cases

### 4.5.1. Usecase1: functionalities of Doctor’s office:

When a patient visits the Doctor’s office, the staff will compile a medical record for the patient. This record will be submitted to the ERD in the event medical services are needed.

### 4.5.2. Usecase2: functionalities of Claim’s Team:

The Claims team of the hospital or medical institution where services will be provided must receive patient records via the ERD feed and enter them into the MBS to generate a medical claim then submit them to Insurance companies for approval or denial.

### 4.5.3. Usecase3: functionalities of Insurance Companies:

The Insurance companies will reject or approve claims based on its internal processing system of patient insurance policies and awards.

### 4.5.4. Usecase4: functionalities of Billing Team:

Upon receipt of the claim form the Insurance company, the Billing team will either process or escalate the claim via the MBS and the system will then post payment amount for services based on Insurance awards or other cost effect.

### 4.5.5. Usecase5: functionalities of Management/Executive Team:

The MBS will generate reports daily or routinely based on Management and Executive teams specifications. Reports are for internal use and review of the system and product services.

## 4.6 System Evolution

The system must undergo maintenance to adapt to new technologies and changes in the technology world. Having a cloud-based system is ideal for ease of migration and adaptation or implementation into any medical institution. In this project the system can be deployed and implemented with the only systemic requirement being a data stream for patient medical record data; with the assumption Internet is active and computers are provided.

The project is currently web-based without intention of moving to mobile apps. This is justified in that mobile apps reside on mobile devices that are easily susceptible to theft or data siphoning. Perhaps when mobile security has come far along the system can be ported over to a mobile platform.

# 5. Other Nonfunctional Requirements

## 5.1     Performance Requirements

The software application will be a web based tool; as a result, limited requirements will be needed. Other than, taking into consideration the volume of data transmitted between servers. This will create a lag time that will need to be taken into consideration between all the different servers and processes that interact with each other. These should be controlled by taking into consideration the loading and processing times of data. Other performance requirements that should be taken into consideration but not limited to are:

* The local machine must meet internet software/browsing expectations.
* Virtual Machine must meet transactional data volumes.
* Database management tool should have processing features to manage the transactional data volume.

## 5.2 Safety Requirements

There should always be safety requirements implemented to minimize the loss of data. Backup measures should be put into work for any real time transactions, logged communications, and database entries. Some other safety requirements that should be taken into consideration but not limited to are:

* Backup of Database
* Backup Virtual Machine
* When computer freezes and there is no activity on the software for a certain amount of time, the application should save the work and disconnect.

## 5.3 Security Requirements

The specifications below will protect the application and servers from accidental or malicious activity, modifications, destruction or disclosure of all integral data.  The details requirements to be follow are but not limited to:

* All input parameters must be validated.  Please follow all coding standards
  + <https://www.securecoding.cert.org/confluence/display/seccode/SEI+CERT+Coding+Standards>
  + This will eliminate breaches to our system using SQL injections and cross-site scripting attacks. (http://www.infosec.gov.hk/english/technical/files/web\_app.pdf)
* Keep sensitive session values on the server to prevent client-side modification
  + strong cryptographic techniques should be used to protect the confidentiality and integrity of the data (infosec.gov, 12)
* Encrypt pages containing sensitive information and prevent caching
  + Pages containing sensitive information should be encrypted with proper algorithms and keys such as SSL and TLS during transmission (infosec.gov, 12)
* Session Management
  + A session ID should be long, complicated, contain random numbers that are unpredictable, and it should be changed frequently during a session to reduce the duration that a session ID remains valid (infosec.gov, 13)
* Access Restriction
  + Ensure that the end-user account only has specific privileges to access those functions that they are authorized to access, restricting access to the backend database, or to run SQL commands, and OS commands (infosec.gov, 13)

## 5.4 Software Quality Attributes

The medical billing software is a guaranteed product to be user friendly with a variety of tools to view the life of the medical claim.  The user interface is a self-explanatory and users with little training should not have any difficulties navigating through application.  The main tasks that the software was designed to perform can easily be accessed through the main menu.  Other attributes are but not limited to:

* The web application is designed to operate using any Internet browser.
* All changes and updates to the interface will be seamless.
* The software will have access to all facility profiles using the most up to date and secure methods.
* Data exchanges are designed to operate in secure networks.

## 5.5 Business Rules

Access and operations to the different user interfaces in the MBS must be controlled and restricted. This will ensure credibility across the different medical billing operations. Below is a list of different structures but are not limited to.

* Claims building should be specific duties and restricted to the data entry/claim building team.
* Billing and claim reconciliation duties should be restricted to only the billing team.
* All reporting and administrative duties will be restricted to the billing management and executive team.

# 6 Testing

## 6.1 Testing Overview

Several testing strategies should be taken into consideration when adding new features or finding bugs in the MBS. Below is a list of testing strategies that should be taken into consideration but not limited to:

* Basic Principles for all software testing.

o   Spiral Model testing should be practiced, testing should begin “in the small” and progress toward testing “in the large”.

* Unit Testing
* Integration Testing
* Validation Testing
* System Testing

o   All tests should be traceable to customer requirements.

o   Tests should be planned long before testing begins.

o   Exhaustive testing is not possible.

* The content model for the WebApp is reviewed to uncover errors prior to any testing.
* Interface model is reviewed to ensure that all use cases can be accommodated.
* Design model for the WebApp is reviewed to uncover navigation errors before any testing.
* User interface is tested to uncover errors in presentation and/or navigation mechanics.
* Each functional component is unit tested.
* Navigation throughout the architecture is tested.
* The WebApp is implemented in a variety of different environmental configurations and is tested for compatibility with each configuration (e.g. Firefox, Chrome, Internet Explorer, etc.)
* Security tests are conducted in an attempt to exploit vulnerabilities in the WebApp or within its environment.
* Performance tests are conducted.
* The WebApp is tested by a controlled and monitored population of end users (Alpha or Beta testing).  The results of their interaction with the system are evaluated for errors.

## 6.2 Performance Testing

Performance testing should be conducted regularly. During an implementation of a new feature and throughout the life of the Medical Billing System. This will ensure that all data exchanges and functions within the application are working to the extent of their capabilities. At this time because of the limited team resources, performance testing will be outsourced to the following third party applications but not limited to:

* BugBuster
* Mentora (Forsythe)
* Visual Studios

## 6.3 Security Testing

Security testing is recommended to be performed on the regular basis. Changes and upgrades between third party software providers and platform browsers are taking place on a regular basis, this should be accompanied with security testing at the same rate of all the changes taking place within the different environments that support the MBS. Because of the limited team resources available, all the security testing should be outsourced to third party software providers. Below is a list of a few softwares that should be used but not limited to:

* Mentora (Forsythe)
* Veracode

### 6.3.1. Lesson’s Learned:

* Advantages of automated web vulnerability scanners
  + Automatically scan a web application for web vulnerabilities
    - Check for web server’s vulnerability
    - Server’s configuration is secure
    - Crawling a web application
    - Checking for generic security issues such as application errors, source code disclosure and more
    - Scan every input and parameter for web vulnerability such as SQL injections, XSS and more
* Network security scanners
  + Secure network services (FTP, API)
* Firewalls
  + Do not analyze web traffic

## 6.4 Compatibility Testing

Throughout the life of the software application, the Medical Billing Software should be tested for compatibility among all the platforms that will support the tool. Since changes and upgrades are always taking place through the life of all the commercial tools that will be used to integrate the MBS. At the same time, we should test for compatibility when new features and bugs are implemented or found in our Medical Billing Software to ensure a better experience for our clients. Taking into consideration the limited team resources, we should outsource the testing to the following third party applications but not limited to:

* Test different browsers
  + CrossBrowserTesting.com
* Different Interfaces
  + Access to different Servers and Databases

## 6.5 Test Case Scenarios (Need to revise to Mirror Master Test Plan)

Below is the breakdown of the Testing Scenario Matrix

* Test Scenario ID – Every test scenario must have a unique ID for tracking simplification.
* Requirement-Reference Document Index – The reference will assist to track back to the SRS document.
* Test Scenario Description – Specifies what to test to achieve a specific object (one liner).
* Positive/Negative – Result of the test scenario.
* Importance – Classification of the importance to specific functionality of the application.  The system used will be a point system (1-5), 5 being most important, 1 being less important.
* No. of Test Cases – Number of instances the test case might end up writing that one test scenario.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Scenario ID | Requirement-Reference Document Index | Test Scenario Description | Positive/Negative | Importance | No. of Test Cases |
| TEST0001 |  | Performance – Loading EMR Data and Processing |  |  |  |
| TEST0002 |  | Performance – Transforming and Exchanging Data with Clearinghouse. |  |  |  |
| TEST0003 |  | Performance – Multiple User’s in Environment |  |  |  |
| TEST0004 |  | Reliability - |  |  |  |
| TEST0005 |  | Security – Third Party Security Application |  |  |  |
| TEST0006 |  | Validation – Login credential test (Number of Instances 3) |  |  |  |
| TEST0007 |  | Validation - Claim Building Test |  |  |  |
| TEST0008 |  | Validation - Claim Submission Test |  |  |  |
| TEST0009 |  | Validation - Claim Reconciliation Test |  |  |  |
| TEST0010 |  | Validation - Medical Claim Payment Posted. |  |  |  |
| TEST0011 |  | Compatibility - CrossBrowserTesting.com |  |  |  |

Figure 3: Test case form