High Level Design

Hospital Management System

Revision Number: 1.0
Last date of revision: 1/2/23

C2312
Quest Global

TEAM 04

Change Record

Revision	**Date**	**Author**	**Changes**

Contents

1. **Introduction**

1.1. Why this High Level Design Document?

The purpose of this High Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

1.2. Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

1.3. Definitions

- HMS – Hospital Management System
- Tomcat SQL Server – A database management system.
- JDBC – A possible Java-based interface between IP Tables and the Database.
- JSP – The language that will be used for displaying user history and administrative functionality.
- Tomcat – a free, open-source implementation of Java Servlet and Java Server Pages technologies developed under the Jakarta project at the Apache Software Foundation.
- Apache - An open source Web server.
- ER – Entity Relation Diagram
- CBQ –Class-Based Queuing. Limits bandwidth at the IP/port level.
- Kernel – Core of an operating system, a kernel manages the machine's hardware resources (including the processor and the memory), and provides and controls the way any other software component can access these resources.
- Gateway –Bridges the gap between the internet and a local network.

1.4. Overview

The HLD will:

- present all of the design aspects and define them in detail
- describe the user interface being implemented
- describe the hardware and software interfaces
- describe the performance requirements
- include design features and the architecture of the project
- list and describe the non-functional attributes like:
    - security
    - reliability
    - maintainability
    - portability
    - reusability
    - application compatibility
    - resource utilization
    - serviceability

2. **General Description**

2.1. Product Perspective

Hospitals interact with a lot of people in a day and there are various activities involved in day to day operations of hospitals, for example booking of appointments, managing doctor schedules, managing patient diagnoses, managing medical histories of patients, etc. The aim of this project is to show how data related to these tasks can be made easier to manage using databases.

2.2. Tools used

1. Jude, a Java based UML design program, is used to generate all of the diagrams used in analysis and design phases of the project.
2. The project will have a relational database backend that is SQL based. The actual software used is PostgreSQL.
3. Interfacing with the database to display information on the user's web browser will be done using JSP. It can connect to the database and parse it into viewable HTML code.
4. Tomcat compiles JSP pages into servlets to be displayed through Apache.

BACK END

Framework	Spring MVC
ORM Tool	Hibernate
Database	MySQL
Build Tool	Maven
Language	Java

➢ FRONT END

HTML
CSS
Java
Script
Bootstrap
JSP

2.3. General Constraints

1. The Hospital Management System must be user friendly and as automated as possible. Administrators should not be required to do anything besides the initial setup,and users should not be required to know any of the workings. The “Hospital Management System” should run on all Internet Browser and all processors which supports the Internet Browser.

2.4. Assumptions

Our project Hospital Management system includes registration of patients, storing their details into the system, and also booking their appointments with doctors. Our software has the facility to give a unique id for every patient and stores the details of every patient and the staff automatically. Receptionist can assign the doctor for the patient, and Make a bill.

3. Design Details

3.1. Main Design Features

The main design features include five major parts: the architecture, the user interface design, external interface, the database, process relation, and automation. In order to make these designs easier to understand, the design has been illustrated in attached diagrams (ER, Use Case, and Screen Shots).

➢ Admin

The super user, admin class represents complete authority over the system an admin can

1. Admin can register Doctors, Pharmacists and Receptionists
2. View the number of users.
3. Admin can update his own profile.
4. Admin can update and delete the users.
5. Admin can view his dashboard which displays all the data entered and generated by the system.
6. Admin can see bill report for particular patient.

➢ Doctor

1. Can view his/her Patients and their appointments.
2. Can update Patient treatment status.
3. Can assign medicines to Patients based on their diagnosis
4. Adds the fee amount for the particular patient.

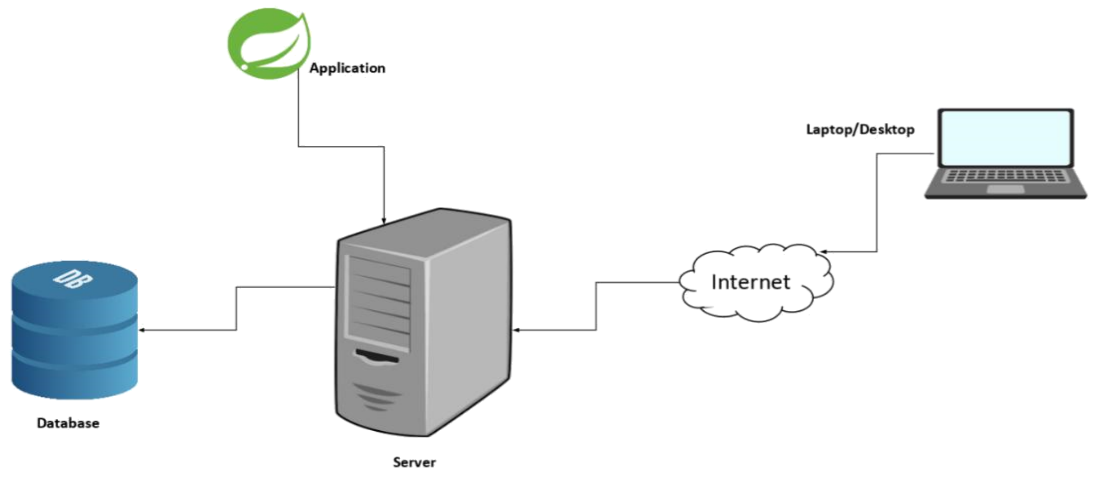
➢ Pharmacist

1. Can add Medicines.
2. Can add Distributors.
3. Can add Medicine Company.

➢ Receptionist

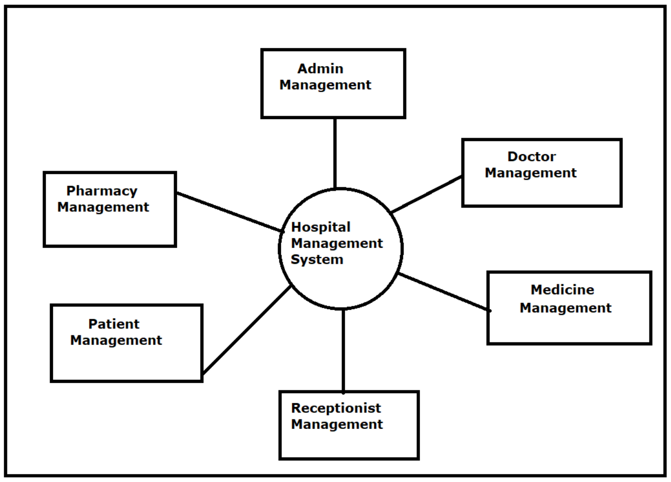
1. Can register Patients.
2. Can book appointments for patients.
3. Can view Appointment status and Treatment status for patients.
4. Can generate and download patient Bill.

3.2. Application Architecture



3.3.1. Web Application Architecture

The front end of the program is a web application. Functionality will vary based user privileges if a patient wants to log in he/she should first sign in to the website. The Doctor just needs to sign in and look into his patients database and accept/reject appointments.



➢ Scenario 1: Mainline Sequence

1. Admin: Enter Username and Password.

2. System:

1. Display the Admin page where admin can see options as admin dashboard and register users.

2. Admin can register doctor, receptionist and pharmacist.

➢ Scenario 2: Mainline Sequence

1. Receptionist: Enter username and Password.

2. System:

1. Register patients.

2. Book Appointments.
3. Generate Bill.

➢ Scenario 3: Mainline Sequence

1. Doctor: Enter username and Password

2. System:

1. View Appointments

2. Treat Patients and change status.

3. Assign Treatment Price.

4. Suggest Medicines.

➢ Scenario 4: Mainline Sequence

1. Pharmacist : Enter username and Password

2. System:

1. Add Medicine Distributors

2. Add Medicine Company.

3. Add Medicines.

3.3.1.Presentation Layer

Information about the patients will be displayed to the doctor. Patients can see is their appoints is confirmed or not by logging into their data base.

3.3.2. Data Access Layer

The database will be maintained. Only patients can see their own appointments and doctors can see their patient details.

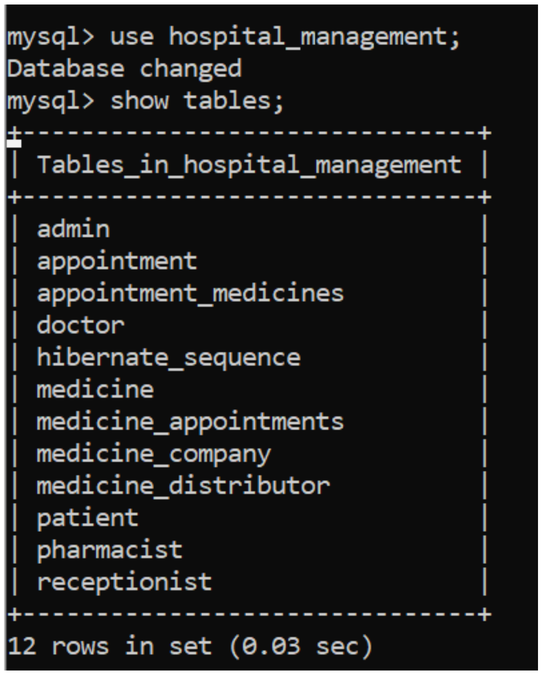
3.3.3. Tools Used

See section 2.2 for tools used in the design of this project.

3.3. Standards

Database – relational
Inputs – entered through text field and stored in database.
Security – username and password are required for access to the system.
Quality – by keeping the interface simple and direct, quality should be kept at amaximum.

3.4. Database design:



```
mysql> use hospital_management;
Database changed
mysql> show tables;
+-------------------------------+
| Tables_in_hospital_management |
+-------------------------------+
| admin                         |
| appointment                   |
| appointment_medicines         |
| doctor                        |
| hibernate_sequence            |
| medicine                      |
| medicine_appointments         |
| medicine_company              |
| medicine_distributor          |
| patient                       |
| pharmacist                    |
| receptionist                  |
+-------------------------------+
12 rows in set (0.03 sec)
```

3.5. Files

All the files related to this project are stored in the database.

3.6. User Interface

The user interface is a very simple plain layout with little to no graphics. It will display information very clearly for the patient and will primarily output information to the patient through HTML pages. Administrative screens are use mainly for input through text fields in HTML pages. Screen shots have been provided to demonstrate the patient and administrative interface.

3.7. Reports

The reports will display the user's average usage up to the last time the system calculated it and their history.

3.8. Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong. An error will be defined as anything that falls outside the normal and intended usage.

3.9. Interfaces

There are three main interfaces for this project. The patients sign in process where the patient has to sign into the system using his information. The second is the patients log in where the patients just need to log in using their id and just take appointment from the doctor. The third is the Doctor Login where the doctor has to log in and check the patients database and confirm their appointments.

3.10. Help

Help will come in the form of all the documentation created prior to coding, which explain the intended uses. Should time allow, detailed instructions will be written on how to create and implement the system with the intentions of publishing as an Open Source solution .

3.11. Performance

Performance is going to be very important for this project. For everything to run smoothly for this project, the gateways will have to be able to update data on the database and refresh the Tables before it is supposed to do so again. This is likely to be the most processor intensive aspect of the project. The gateways will also need to supply requested pages to the users at a reasonable speed. The database server will need to keep up with all database requests and transactions.

3.12. Security

Because security is not the prime focus of this project, only the minimal aspects of security will be implemented. A username and password will be required to log into patient and doctor. For now, all data will be sent in plain text.
For now, there will also be no log of failed attempts of an administrator logging in.

3.13. Software System Attributes

Usability: Software can be used again and again without distortion.
Availability: The system shall be available all the time.
Correctness: Bug free software which fulfills the correct need/requirements of the client.
Maintainability: The ability to maintain, modify information and update fix problems of the system.
Accessibility: Administrator and many other users can access the system but the access level is controlled for each user according to their work scope.

3.14. Maintainability

Very little maintenance should be required for this setup. An initial configuration will be the only system required interaction after system is put together. The only other user maintenance would be any changes to settings after setup, and any specified special cases where user settings or history need to be changed. No Physical maintenance on the system is required. Upgrades of software should have little effect on this project, but may result in downtime.

3.15. Portability

This system should have the ability that, once it is together, the entire system should be able to be physically moved to any location. Code and program portability should be possible between any OS distributions. For everything to work properly, all components should be compiled from source.

3.16. Reusability

The code written and the components used should have the ability to be reused with no problems. Should time allow, and detailed instructions are written on how to create this project, everything will be completely reusable to anyone.

3.17. Application compatibility

The different components for this project will be using Java as an interface between them. Each component will have its own task to perform, and it is the job of the Java code to ensure proper transfer of information. MySQL maintains a good record of the data that is getting stored

3.18. Resource utilization

When any task is performed, it will likely use all the processing power available until that function is finished. This helps the patients and doctors to maintain a good record of data and also helps in confirmation of their appointments.

3.19. Major Classes

There are a total of five major classes: Patient Login, Patient Sign in, Doctor Login. The relationships between these major classes are:

A user can book appointment.
A user can see his history.
A Doctor can see patient details.
A Doctor can confirm for appointments of Patient.