**COVER PAGE**

A logo of a university

Description automatically generated

EEC 521/CIS 534: Software Engineering

Professor: Dr. Yongjian Fu

|  |  |  |
| --- | --- | --- |
| **Team Members** | | |
| Team Member 1 | LIKITH REKAPALLI | CSU ID: 2845684 |
| Team Member 2 | KRISHNA SAI SALLA | CSU ID: 2887720 |

**SOFTWARE DESIGN SPECIFICATION**

**1.0 Introduction**

This document details the software design for the Health Care Hospital System, including architectural designs, data management, system components, and user interface design. It offers a comprehensive guide for the implementation of the system based on the waterfall model. The design reflects the structure needed to support the functionalities, constraints, and performance requirements outlined in the software requirements specification.

**1.1 Goals and Objectives**

The Health Care Hospital System aims to streamline hospital administrative operations by offering functionalities for patient registration, room management, staff assignment, and invoice generation. The system should provide a user-friendly, responsive interface for hospital staff and administrators, and be scalable for potential future enhancements.

**1.2 Statement of Scope**

This software manages hospital operations, including patient records, staff assignments, room tracking, and billing. It processes data input through forms and provides outputs in the form of reports and organized data tables (e.g., patient lists, staff rosters, room status, invoices). The software does not currently support real-time processing or third-party integrations.

**1.3 Software Context**

The system is intended for small to medium-sized hospitals seeking a simple, localized solution for managing administrative tasks. It can be deployed on a local server (using WAMP/XAMPP) and accessed via a web browser (Chrome, Firefox, IE, etc.).

**1.4 Major Constraints**

* No deployment or completion of unfinished features, as the project remains partially developed.
* No debugging or feature enhancement is planned in this phase.
* Limited scalability for large hospital networks due to its standalone nature.
* Limited system performance, as the database and interface will only be optimized for single-server, small-to-medium scale operations.

**2.0 Data Design**

**2.1 Data Structures**

The data design is focused on relational databases, using MySQL to manage hospital operations efficiently. Key tables include:

* Patient Table: Manages patient-related information.
* Attributes: patient\_id, first\_name, last\_name, age, gender, address, phone\_number, medical\_history.
* Staff Table: Stores staff-related information.
* Attributes: staff\_id, name, role, department, contact\_information.
* Room Table: Stores information about hospital rooms.
* Attributes: room\_id, room\_number, room\_type, doctor\_assigned, availability\_status, cleaning\_schedule.
* Invoice Table: Contains information about patient invoices.
* Attributes: invoice\_id, patient\_id, charges, date\_of\_invoice, doctor\_fee, medicine\_fee, hospital\_fee, total\_fee.
* Admitted Patients Table: Manages records of admitted patients and their rooms.
* Attributes: admission\_id, patient\_id, room\_id, admission\_date, discharge\_date.

**2.2 Database Description**

**Database Type**: Relational database using MySQL.

**Entity Relationships**:

* **Patients and Invoices**: A patient\_id links the patient to multiple invoices.
* **Rooms and Patients**: A room\_id links a patient to their assigned room.
* **Staff and Patients**: Staff can be linked to multiple patients through their staff\_id.

Relationships are maintained through foreign keys for data integrity, e.g., patient\_id in the Invoice table references the Patient table to establish a patient-invoice relationship.

**3.0 Architectural and Component-Level Design**

**3.1 Architecture Diagrams**

The architecture follows the Model-View-Controller (MVC) pattern, ensuring separation of concerns between the user interface, data, and business logic.

* Model Layer: Contains all the logic related to data manipulation, database queries, and the handling of patient, staff, room, and invoice information.
* View Layer: Contains HTML, CSS, and JavaScript code to generate user interfaces. Bootstrap is used for styling and creating responsive designs, while AJAX is used for dynamic content updates without page reloads.
* Controller Layer: Acts as an intermediary between the user (view) and data (model). PHP handles requests, processes business logic, and returns appropriate views.

**3.2 Description for Components**

**3.2.1 Patient Management Component**

* **Interface Description**: Allows administrators and staff to input and manage patient data. Includes search, view, edit, and delete functionalities.
* **Data Inputs**: first\_name, last\_name, age, gender, address, medical\_history.
* **Output**: Patient records displayed as searchable tables or modifiable forms.
* **Dynamic Models**: Activity diagram depicting patient registration and information editing.

**3.2.2 Staff Management Component**

* **Interface Description**: Admin users can add and modify staff details, including department assignments and contact information.
* **Data Inputs**: staff\_name, role, contact\_info, department.
* **Output**: Staff lists and individual staff profiles.
* **Dynamic Models**: Sequential diagrams depicting staff assignments to patients.

**3.2.3 Room Management Component**

* **Interface Description**: Staff can update room availability, assign rooms to patients, and track cleaning schedules.
* **Data Inputs**: room\_number, availability\_status, doctor\_assigned, cleaning\_time.
* **Output**: Room lists with filters for availability.
* **Static Models**: ERD diagram showing the relationship between rooms and admitted patients.

**3.2.4 Invoice Management Component**

* **Interface Description**: Generates patient invoices and allows administrators to view and manage financial records.
* **Data Inputs**: patient\_id, doctor\_charge, medicine\_charge, hospital\_fee.
* **Output**: Display of individual invoices and total costs for patients.
* **Dynamic Models**: Activity diagram depicting invoice creation, payment tracking, and status updates.

**3.3 External Interface Description**

Currently, the system does not interact with any external software. Future integration with external systems (such as EHR or payment gateways) may be possible but is out of scope for this version of the project.

**4.0 User Interface Design**

**4.1 Description of the User Interface**

The user interface is designed using Bootstrap, making it responsive and intuitive. Key pages include:

* **Admin Dashboard**: Displays key hospital metrics, patient counts, room availability, and financial data.
* **Patient Registration Page**: Provides form-based input for new patient details, with options to search, edit, or delete existing records.
* **Staff Management Page**: Lists all staff members with search and edit functionalities.
* **Invoice Management Page**: Displays patient invoices in tabular format with filters for date and payment status.

The UI uses AJAX for real-time data updates without full-page reloads.

**4.2 Interface Design Rules**

* **Consistency**: Consistent layouts for all pages, including uniform button placement and form field order.
* **Feedback**: Visual feedback for user actions (e.g., successful form submission, validation errors).
* **Accessibility**: Design adheres to basic accessibility guidelines, including appropriate color contrast and readable font sizes.
* **Responsive Design**: Layout adapts for desktop, tablet, and mobile views.

**5.0 Restrictions, Limitations, and Constraints**

* **Performance Limitations**: The system is optimized for single-server deployment and may not scale efficiently for large hospital networks.
* **No Real-Time Features**: The system does not provide real-time updates; data is fetched upon request.
* **Future Integration**: The current version lacks integration with third-party systems such as payment gateways or EHR systems.

**6.0 Appendices**

**6.1 Requirements Traceability Matrix**

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Component** | **Data Structures** |
| Patient Registration | Patient Management | Patient Table |
| Staff Management | Staff Management | Staff Table |
| Room Availability | Room Management | Room Table |
| Invoice Management | Invoice Management | Invoice Table |

**6.2 Implementation Issues**

* **Database Query Optimization**: Reducing the time for data retrieval, especially in high-traffic scenarios.
* **Error Handling**: Robust handling of errors like missing data and database connection issues.
* **Security**: Implementation of user authentication and role-based access control to protect sensitive data.