TechArch Solutions
235 Sycamore Street
Santa Clara, CA 95054
+1 408.555.3840

info@techarch.domain
techarch.domain
Company Representative
Eddie J. Stevens

System Design Specification

Design Specifications
for Aurora Health

Prepared for:

Chris Holt

Aurora Health
1412 Horseshoe Lane
Philadelphia, PA 19108
+1 484.555.0368
info@aurora.domain

Version:
Version 3

Date Published:
06.10.2021

Table of Contents

1. General 1
1.1. Purpose 1
1.2. Scope 1
1.3. System Overview 1
1.4. Point of Contact 1
2. System Architecture 2
2.1. Assumptions and Dependencies 2
2.1.1. Assumptions 2
2.1.2. Dependencies 2
2.2. General Constraints 2
2.3. Proposed System Architecture 2
2.3.1. System Architecture Layers 2
2.4. Data Characteristics and Categorization 3
2.6. Logical Database Design 4
2.7. Database Tables 4
2.8. Data Dictionary 5
3. Data Policies 8
3.1. Data Constraints 8
3.2. Data Retention 8
3.2.1. Log Retention 8
3.3. System 8
3.3.1. Software 8
3.3.2. Users 8
3.4. Table Design 9
3.4.1. Common Table Attributes 9
3.4.2. Billing 9
3.4.3. Credit Types 9
3.4.4. Drugs 9
3.4.5. Insurors 9
3.4.6. Inventory 9
3.4.7. Orders 10

3.4.8. Patients 10
3.4.9. Staff 10
3.4.10. Suppliers 10
3.4.11. Transactions 10
4. Deployment and Implementation 11
4.1. Deployment View 11
4.2. Implementation 11
5. Persistent Data Model 12
5.1. Patients 12
5.2. Staff 12
5.3. Insurors 12
5.4. Inventory 13
6. Appendix 14

1. General

1.1. Purpose

The purpose of this document is to provide specific details and requirements of the Aurora Health database design. This document includes an in-depth description of the definition of inputs, outputs, procedures, and interactions of the database product.

1.2. Scope

The scope of this System Design Specification is to outline the data model that will be employed by the system to effectively meet its functional requirements. The scope of this specification encompasses the drug and patient tracking capabilities within the organization. The system will encompass a front-end and back-end architecture designed to securely store data, perform transactions for data processing, and facilitate user input and interactions.

1.3. System Overview

The System Design Specification incorporates MySQL as the chosen database solution to effectively track and manage patient and drug information. The database will successfully handle various aspects including inventories, drug details, and prescriptions. The implementation of MySQL will ensure efficient storage, retrieval, and manipulation of data, enabling seamless tracking and management of pharmaceutical information for patients and drugs within the system.

1.4. Point of Contact

Specific points of contact may be reached throughout the development of the system. Information for these contacts is provided in the table below.

POINTS OF CONTACT

Name:	Melvin K. Sailsbury
Position:	Software Development Manager
Phone Number:	(715) 555-4703
E-mail:	msalisbury@tgri.domain

2. System Architecture

2.1. Assumptions and Dependencies

2.1.1. Assumptions

The System Design Specification is based on certain assumptions to ensure the smooth operation of the system. It is assumed that end users will have access to specific software components for optimum functionality, including a web browser or a mobile application with internet connectivity. The system also assumes that well-defined user roles and access levels are implemented to ensure data security and privacy. Accesses and roles may include administrators with full control over the system, healthcare professionals with privileges to view and update patient and drug information, and patients with limited access to their own medical records. These assumptions will guide the design and development of the system, ensuring that the appropriate software and access mechanisms are in place to meet the requirements of the end users.

2.1.2. Dependencies

To access and utilize the database efficiently, end users are required to have certain prerequisites. Firstly, a computer system with compatible hardware and software specifications is necessary. This includes a reliable operating system such as Windows, macOS, or Linux, along with sufficient processing power and storage capacity. Additionally, a stable internet connection is required to enable seamless communication between the user's device and the database server. Furthermore, appropriate permissions and login credentials will be provided to authorized users to ensure secure access to the database. These prerequisites are essential to establish a functional user environment, enabling end users to effectively interact with the database and utilize its features and functionalities.

2.2. General Constraints

To optimize the use of the system, the system will use standardized software among user workstations with default references and libraries. A balance between back-end and front-end operations should be maintained to make the most effective use of system operation.

2.3. Proposed System Architecture

The system architecture lays out the conceptualized design for the system. The system architecture is divided into distinct areas to improve overall functionality. Functionality is separated at boundaries that segment the application into unique areas. This allows for each segment to be responsible only for one or a few functions, thus partitioning the application to make it more efficient and to reduce the impact of system issues to smaller parts of the application.

2.3.1. System Architecture Layers

The system architecture is broken down into several layers that segment the system into the following components:

- Presentation Layer: this layer consists of the graphical front-end UI components from which the user will interface through the web application.
- Business Layer: this layer will oversee the end-to-end operations, encompassing both the front-end and back-end functionalities.
- Data Layer: this layer consists of the back-end data repository from which data is stored, read, and written.

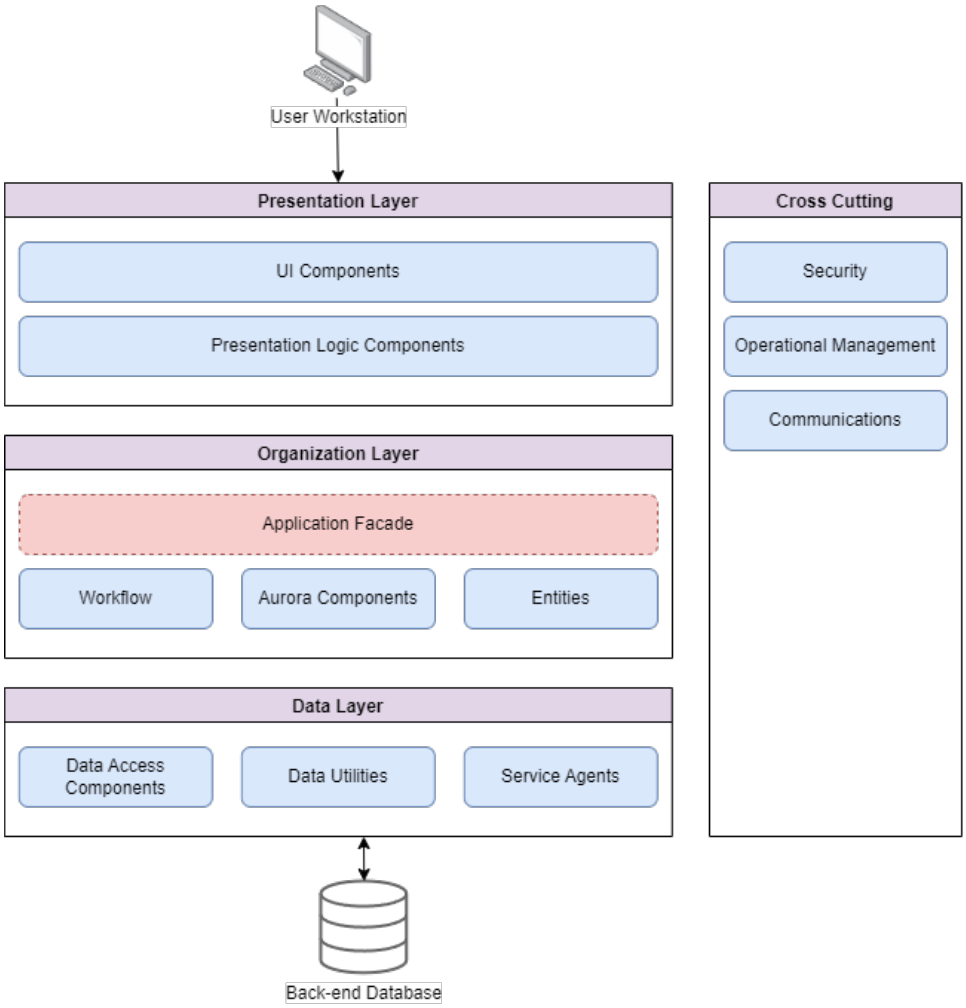

Figure 1: System Architecture

2.4. Data Characteristics and Categorization

The database will utilize a diverse range of data elements, encompassing alphanumeric, alphabetic, numeric, and Boolean data types. The database will be composed of multiple interconnected tables, facilitating efficient organization and management of data. These tables, residing within the back-end database, will establish a seamless link with the front-end application. The back-end database itself will be hosted at the designated location: https://aurora.domain.

2.6. Logical Database Design

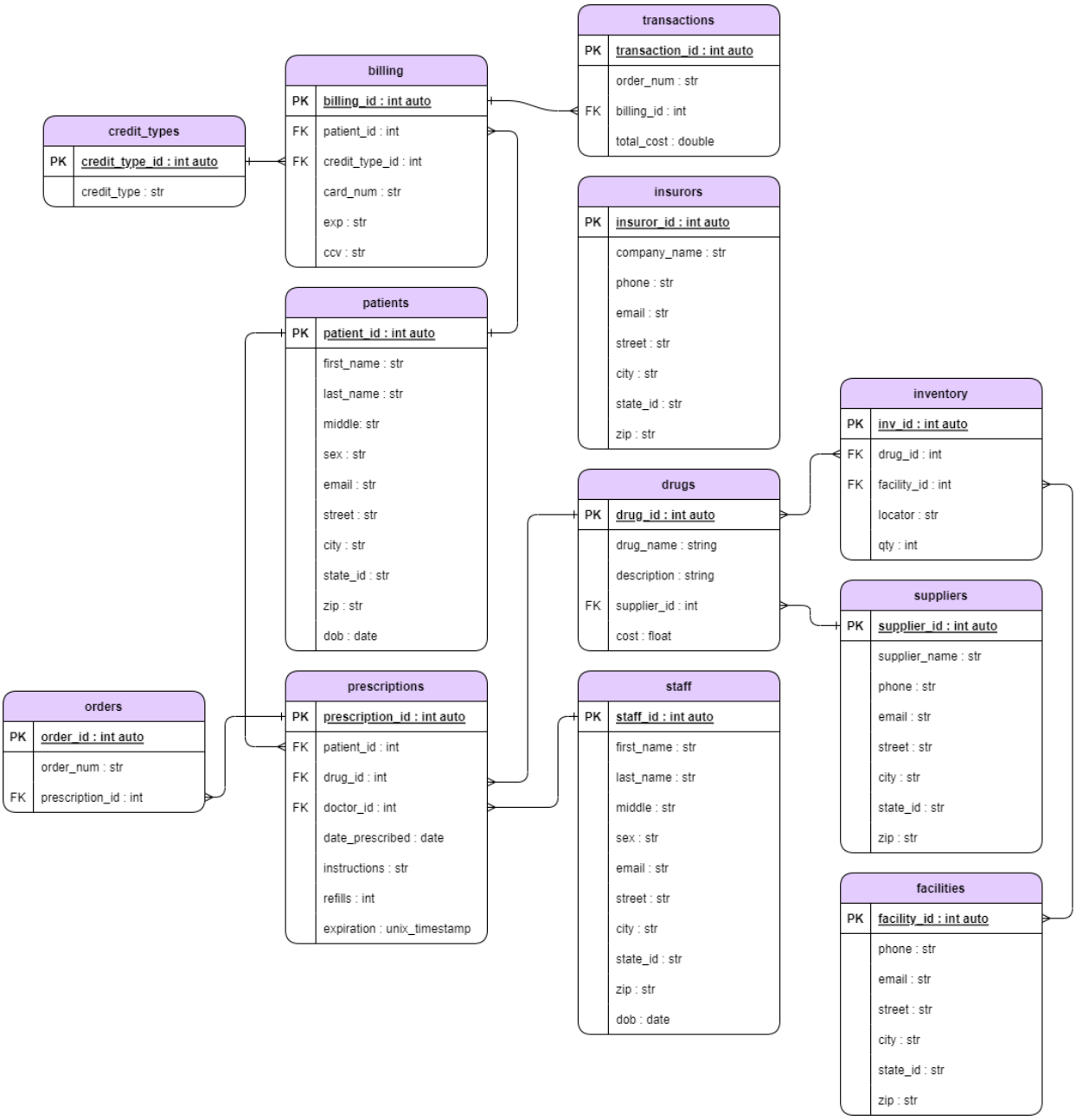

Figure 2: Logical Database Design

2.7. Database Tables

The database will be constructed with the following tables:

- billing (billing_id, patient_id, credit_type_id, card_num, exp, ccv)
- credit_types (credit_type_id, credit_type)

- drugs (drug_id, drug_name, description, supplier_id, cost)
- facilities (facility_id, phone, street, city, state, zip)
- insurors (insuror_id, company_name, phone, email, street, city, state, zip)
- inventory (inv_id, drug_id, facility_id, locator, qty)
- orders (order_id, order_num, prescription_id)
- patients (patient_id, first_name, last_name, middle, sex, email, street, city, state, zip, dob)
- positions (position_id, position_name)
- positions_assigned (position_assigned_id, staff_id, position_id)
- prescriptions (prescription_id, patient_id, drug_id, doctor_id, date_prescribed, instructions, refills, expiration)
- staff (staff_id, first_name, last_name, middle, sex, email, street, city, state, zip, dob)
- suppliers (supplier_id, supplier_name, phone, email, street, city, state, zip)
- transactions (transaction_id, order_num, billing_id, total_cost)

2.8. Data Dictionary

The data dictionary is referenced below and indicates which data elements are static, dynamic input, dynamic output, and which are internally generating.

Attribute Name	Table	Required	Data Type	Default Value	Duplicates Allowed?	Type
billing_id	billing	✓	int(11)	NULL		Static
patient_id	billing	✓	int(11)	NULL	✓	Dynamic
credit_type_id	billing	✓	int(11)	NULL	✓	Dynamic
card_num	billing	✓	varchar(24)	NULL		Dynamic
exp	billing	✓	varchar(8)	NULL	✓	Dynamic
ccv	billing	✓	varchar(4)	NULL	✓	Dynamic
credit_type_id	credit_types	✓	int(11)	NULL		Static
credit_type	credit_types	✓	varchar(24)	NULL		Static
drug_id	drugs	✓	int(11)	NULL		Static
drug_name	drugs	✓	varchar(56)	NULL		Dynamic
description	drugs	✓	varchar(4096)	NULL	✓	Dynamic
supplier_id	drugs	✓	int(11)	NULL	✓	Dynamic
cost	drugs		float	NULL	✓	Dynamic
facility_id	facilities	✓	int(11)	NULL		Static
phone	facilities	✓	varchar(12)	NULL		Dynamic
street	facilities	✓	varchar(64)	NULL	✓	Dynamic
city	facilities	✓	varchar(64)	NULL	✓	Dynamic
state	facilities	✓	varchar(8)	NULL	✓	Dynamic
zip	facilities	✓	varchar(12)	NULL	✓	Dynamic
insuror_id	insurors	✓	int(11)	NULL		Static
company_name	insurors	✓	varchar(64)	NULL		Dynamic
phone	insurors	✓	varchar(12)	NULL		Dynamic
email	insurors	✓	varchar(128)	NULL		Dynamic
street	insurors	✓	varchar(64)	NULL	✓	Dynamic
city	insurors	✓	varchar(64)	NULL	✓	Dynamic
state	insurors	✓	varchar(8)	NULL	✓	Dynamic
zip	insurors	✓	varchar(12)	NULL	✓	Dynamic
inv_id	inventory	✓	int(11)	NULL		Static
drug_id	inventory	✓	int(11)	NULL	✓	Dynamic

Attribute Name	Table	Required	Data Type	Default Value	Duplicates Allowed?	Type
facility_id	inventory	✓	int(11)	NULL	✓	Dynamic
locator	inventory	✓	varchar(24)	NULL	✓	Dynamic
qty	inventory	✓	int(11)	NULL	✓	Dynamic
order_id	orders	✓	int(11)	NULL		Static
order_num	orders	✓	varchar(32)	NULL		Static
prescription_id	orders	✓	int(11)	NULL	✓	Dynamic
prescription_id	prescriptions	✓	int(11)	NULL		Static
patient_id	prescriptions	✓	int(11)	NULL	✓	Dynamic
drug_id	prescriptions	✓	int(11)	NULL	✓	Dynamic
doctor_id	prescriptions	✓	int(11)	NULL	✓	Dynamic
date_prescribed	prescriptions	✓	date	NULL	✓	Dynamic
instructions	prescriptions		varchar(4096)	NULL	✓	Dynamic
refills	prescriptions	✓	int(11)	NULL	✓	Dynamic
expiration	prescriptions	✓	date	NULL	✓	Dynamic
position_id	positions	✓	int(11)	NULL		Static
position_name	positions	✓	varchar(56)	NULL		Static
position_assigned_id	positions_assigned	✓	int(11)	NULL		Static
staff_id	positions_assigned	✓	int(11)	NULL		Dynamic
position_id	positions_assigned	✓	int(11)	NULL		Dynamic
patient_id	patients	✓	int(11)	NULL		Static
first_name	patients	✓	varchar(32)	NULL	✓	Dynamic
last_name	patients	✓	varchar(64)	NULL	✓	Dynamic
middle	patients		varchar(8)	NULL	✓	Dynamic
sex	patients	✓	varchar(8)	NULL	✓	Dynamic
email	patients	✓	varchar(128)	NULL		Dynamic
street	patients	✓	varchar(64)	NULL	✓	Dynamic
city	patients	✓	varchar(64)	NULL	✓	Dynamic
state	patients	✓	varchar(8)	NULL	✓	Dynamic
zip	patients	✓	varchar(12)	NULL	✓	Dynamic
dob	patients	✓	date	NULL	✓	Dynamic
staff_id	staff	✓	int(11)	NULL		Static
first_name	staff	✓	varchar(32)	NULL	✓	Dynamic
last_name	staff	✓	varchar(64)	NULL	✓	Dynamic
middle	staff		varchar(8)	NULL	✓	Dynamic
sex	staff	✓	varchar(8)	NULL	✓	Dynamic
email	staff	✓	varchar(128)	NULL		Dynamic
street	staff	✓	varchar(64)	NULL	✓	Dynamic
city	staff	✓	varchar(64)	NULL	✓	Dynamic
state	staff	✓	varchar(8)	NULL	✓	Dynamic
zip	staff	✓	varchar(12)	NULL	✓	Dynamic
dob	staff	✓	date	NULL	✓	Dynamic
supplier_id	suppliers	✓	int(11)	NULL		Static
supplier_name	suppliers	✓	varchar(64)	NULL		Dynamic
phone	suppliers	✓	varchar(12)	NULL		Dynamic
email	suppliers	✓	varchar(128)	NULL		Dynamic
street	suppliers	✓	varchar(64)	NULL	✓	Dynamic
city	suppliers	✓	varchar(64)	NULL	✓	Dynamic
state	suppliers	✓	varchar(8)	NULL	✓	Dynamic

Attribute Name	Table	Required	Data Type	Default Value	Duplicates Allowed?	Type
zip	suppliers	✓	varchar(12)	NULL	✓	Dynamic
transaction_id	transactions	✓	int(11)	NULL		Static
order_num	transactions	✓	varchar(32)	NULL	✓	Dynamic
billing_id	transactions	✓	int(11)	NULL	✓	Dynamic
total_cost	transactions	✓	double	NULL	✓	Dynamic

3. Data Policies

3.1. Data Constraints

The database implemented in the system adheres to specific data constraints to ensure data integrity and consistency. Data entry may occur through two channels: the web application or the MySQL prompt. Regardless of the entry method, comprehensive data validation procedures will be implemented to restrict and validate user inputs. These validation mechanisms will enforce predefined constraints, such as data type adherence, field length limitations, and adherence to specific formats. By enforcing these constraints, the database will maintain high data quality and accuracy, preventing the introduction of erroneous or inconsistent data into the system.

3.2. Data Retention

The system incorporates a well-defined data retention policy to handle the storage and management of the database. The database will be securely stored within a designated database management system, such as MySQL, ensuring its availability and accessibility. The system will implement suitable backup and recovery mechanisms to prevent data loss and maintain data integrity. As per the data retention policy, the database will retain relevant information for a specified duration as determined by regulatory requirements and the organization's internal guidelines. Regular data archiving and purging processes will be conducted to maintain an optimal database size and optimize system performance. The storage location will adhere to stringent security measures, protecting the database from unauthorized access and ensuring confidentiality and privacy of the stored data.

3.2.1. Log Retention

Logs will be generated in the database application whenever a user commits database transactions as pre-determined by the development team. Database administrators can set the amount of time (in days) before the logs are automatically purged by the system. Alternatively, administrators may purge logs at their own discretion via the Settings feature.

3.3. System

3.3.1. Software

The system will utilize the most up-to-date stable version of MySQL. Staging and testing of software upgrades should be done prior to implementing changes on the production server.

3.3.2. Users

To effectively operate the database, users of the system should possess certain knowledge and skills. They should be familiar with the basic concepts of relational databases and have an understanding of database management systems. Proficiency in using the web application or MySQL prompt, depending on the preferred method of data entry, is essential. Users should also have knowledge of the specific data elements, tables, and relationships within the database schema, enabling them to navigate and perform relevant operations. Additionally, it is crucial for users to understand the data constraints and validation rules implemented in the system, ensuring that data inputs are accurate and compliant. Adequate training and documentation should be provided to users to familiarize themselves with the system's functionalities and ensure efficient operation of the database.

3.4. Table Design

3.4.1. Common Table Attributes

Many of the tables in the database contain common attributes. These attributes are listed below and are omitted from the table-specific sections that follow.

Field	Description
*_id	The *id* fields are artificial keys unique to each individual record. They use the AUTO_INCREMENT data type to generate unique primary keys and foreign keys for each record.
phone	The unique phone number for the patient, staff, facility, or company.
email	The unique email address for the patient, staff, facility, or company.
street	The street address for the patient, staff, facility, or company.
city	The city of the patient, staff, facility, or company.
state	The state of the patient, staff, facility, or company.
zip	The postal code for the patient, staff, facility, or company.

3.4.2. Billing

Field	Description
card_num	The credit card number associated with patients.
exp	The expiration date for the credit card on file for patients.
ccv	The security code associated with the patient's credit card.

3.4.3. Credit Types

Field	Description
credit_type	The type of credit card (e.g., Mastercard, Visa, etc.)

3.4.4. Drugs

Field	Description
drug_name	The brand name of the drug.
description	The vendor's description of the drug.
cost	The cost associated with each dose of the drug.

3.4.5. Insurors

Field	Description
company_name	The name of the insurance company.

3.4.6. Inventory

Field	Description
locator	The aisle (represented as "A") and the bin (represented as "B") where the drug is stored in the associated facility.
qty	The number of doses of the drug kept on-hand in the facility.

3.4.7. Orders

Field	Description
order_num	A random, 16-alphanumeric string associated with the order.

3.4.8. Patients

Field	Description
first_name	The patient's given name (i.e., first name).
last_name	The patient's surname.
middle	The patient's middle initial.
sex	The patient's sex.
dob	The patient's date of birth.

3.4.9. Staff

Field	Description
first_name	The staff member's given name (i.e., first name).
last_name	The staff member's surname.
middle	The staff member's middle initial.
sex	The staff member's sex.
dob	The staff member's date of birth.

3.4.10. Suppliers

Field	Description
supplier_name	The name of the company that manufactures the associated drug.

3.4.11. Transactions

Field	Description
order_num	The order number associated with the transaction.
total_cost	The total cost to the patient or insuror for the transaction.

4. Deployment and Implementation

4.1. Deployment View

The Deployment View consists of physical nodes and how each node is related during the transaction of information to, from, and within the system.

- End-User terminal – the user's machine that interacts with the system.
- Back-end Database – the MySQL service on the back-end infrastructure
- Front-end application – the web application or MySQL client that interacts with the back-end database.

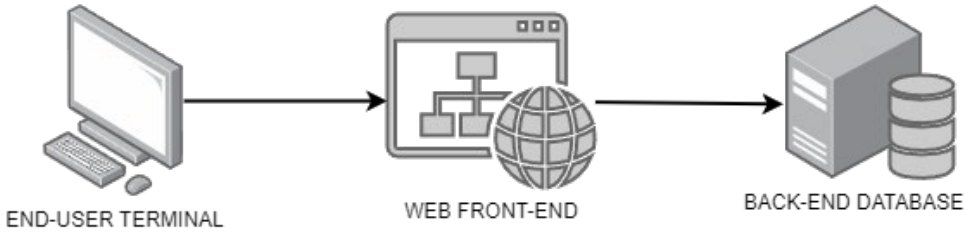

Figure 3: Deployment View

4.2. Implementation

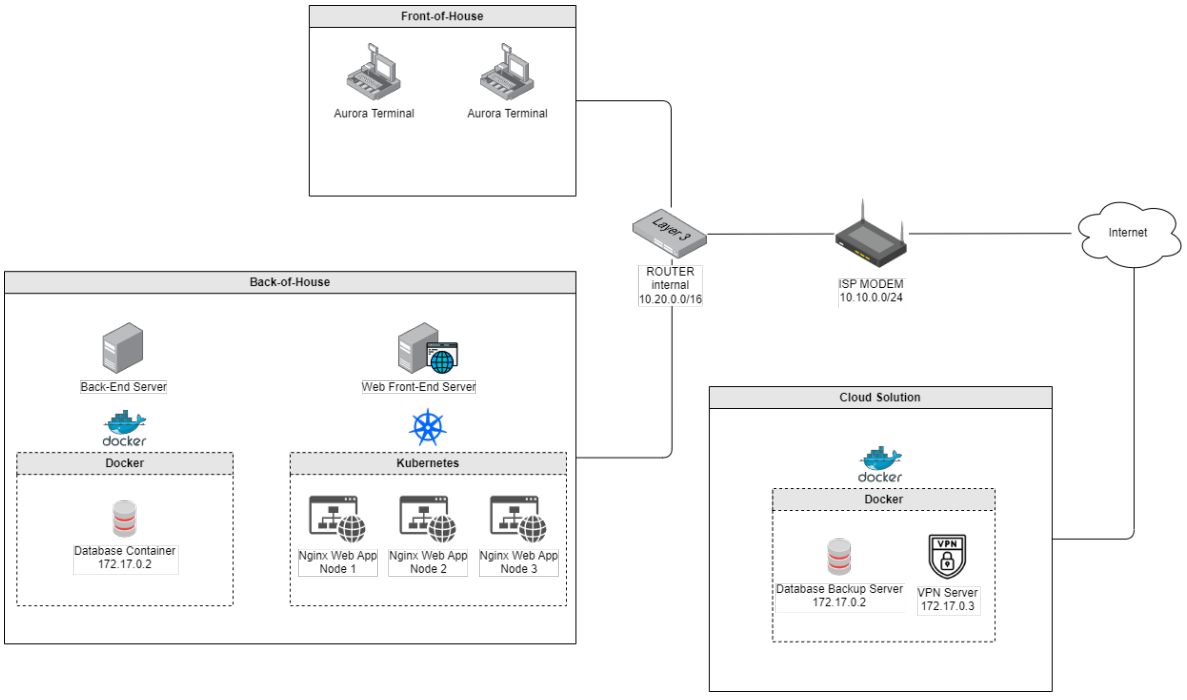

Figure 4: Implementation

5. Persistent Data Model

Below are the SQL statements for the creation of the various tables used within the system. The following are samples of the SQL statements; not every table is documented in this System Design Specification.

5.1. Patients

```
CREATE TABLE IF NOT EXISTS patients (
      patient_id int NOT NULL AUTO_INCREMENT,
      first_name varchar(32) NOT NULL,
      last_name varchar(64) NOT NULL,
      middle varchar(8),
      sex varchar(8) NOT NULL,
      email varchar(128) NOT NULL UNIQUE,
      street varchar(64) NOT NULL,
      city varchar(64) NOT NULL,
      state varchar(8) NOT NULL,
      zip varchar(12) NOT NULL,
      dob date NOT NULL,
      PRIMARY KEY (patient_id)
);
```

5.2. Staff

```
CREATE TABLE IF NOT EXISTS staff (
      staff_id int NOT NULL AUTO_INCREMENT,
      first_name varchar(32) NOT NULL,
      last_name varchar(64) NOT NULL,
      middle varchar(8),
      sex varchar(8) NOT NULL,
      email varchar(128) NOT NULL UNIQUE,
      street varchar(64) NOT NULL,
      city varchar(64) NOT NULL,
      state varchar(8) NOT NULL,
      zip varchar(12) NOT NULL,
      dob date NOT NULL,
      PRIMARY KEY (staff_id)
);
```

5.3. Insurors

```
CREATE TABLE IF NOT EXISTS insurors (
      insuror_id int NOT NULL AUTO_INCREMENT,
      company_name varchar(64) NOT NULL UNIQUE,
      phone varchar(12) NOT NULL UNIQUE,
      email varchar(128) NOT NULL UNIQUE,
      street varchar(64) NOT NULL,
      city varchar(64) NOT NULL,
      state varchar(8) NOT NULL,
      zip varchar(12) NOT NULL,
      PRIMARY KEY (insuror_id)
```

```
);
```

5.4. Inventory

```
CREATE TABLE IF NOT EXISTS inventory (
      inv_id int NOT NULL AUTO_INCREMENT,
      drug_id int NOT NULL,
      facility_id int NOT NULL,
      locator varchar(24) NOT NULL,
      qty int NOT NULL,
      PRIMARY KEY (inv_id),
      CONSTRAINT fk_inventory_drug_id FOREIGN KEY (drug_id) REFERENCES drugs
(drug_id) ON DELETE CASCADE,
      CONSTRAINT fk_inventory_facility_id FOREIGN KEY (facility_id) REFERENCES
facilities (facility_id) ON DELETE CASCADE
);
```

6. Appendix

Figure 1: System Architecture....3
Figure 2: Logical Database Design....4
Figure 3: Deployment View....11
Figure 4: Implementation....11