10/12/2024

Meditronix

Meditronix Project Documentation v2.3.1

Created by : Team Possible
Ahmed Rafid | Fariya Ahmed | Rumman Adib

Contents

1 Introduction **3**
- 1.1 The Doctor's Panel . . . 3
  - 1.1.1 Key Features . . . 3
  - 1.1.2 Key Functionalities . . . 4
- 1.2 The shop management panel . . . 4
  - 1.2.1 Key Features . . . 4
  - 1.2.2 Key Functionalities . . . 5
- 1.3 The customer/patient's panel . . . 6
  - 1.3.1 Key Features . . . 6
  - 1.3.2 Key Functionalities . . . 6

2 System Design **8**
- 2.1 Overall System Architecture . . . 8
  - 2.1.1 Context level diagram . . . 8
- 2.2 Chosen Design Patterns and Technologies Used . . . 9
- 2.3 User Interaction With the System . . . 9
  - 2.3.1 Doctor's Menu Interaction . . . 9
  - 2.3.2 Shop Menu Interaction . . . 10
  - 2.3.3 Customer/Patient's Menu Interaction . . . 11

3 Implementation Details **14**
- 3.1 General Class Implementation Details . . . 14
- 3.2 Implementation Details of the Doctor's Panel . . . 16
- 3.3 Implementation Details of the Shop Management Panel . . . 19
- 3.4 Implementation Details of the Customer/Patient's Panel . . . 24

4 GUI Design **27**
- 4.1 Doctor's Panel . . . 27
- 4.2 Shop Management Panel . . . 32
- 4.3 Customer/Patient's Panel . . . 34

5 Testing and Evaluation **36**
- 5.1 Doctor's Panel . . . 36
- 5.2 Shop Management Panel . . . 37
- 5.3 Customer/Patient's Panel . . . 37

6 Entity Relationship Diagram **38**

7 Conclusion **38**

7.1 key Achievements & Successful Implementation of Functionalities 38

7.2 Limitations or Future Improvements 39

7.2.1 Doctor's Panel 39

7.2.2 Shop Management Panel: 39

7.2.3 Customer/Patient's Panel: 40

8 External Library **40**

1 Introduction

A medical management app that regulates medicine distribution. By providing a comprehensive solution for inventory management, prescription handling, and patient care documentation, *Meditronix* aims to bridge the gap between healthcare providers and patients, ensuring timely access to medication and medical records.

The software integrates doctors, patients, and pharmacy owners into a single domain. It restricts the selling of expired medicines and denies selling prescription medicines without a valid prescription. It does so by storing prescriptions onto a secure database. Whenever a patient goes to buy medicines, the person must provide a valid code for a unique prescription that the system validates before allowing the patient to buy those specific medicines according to the exact dosage mentioned, no more, no less. All subroutines and features work together to accomplish this goal.

The software is split into three core components:

1. The doctor's panel
2. The shop management panel
3. The customer/patient's panel

1.1 The Doctor's Panel

1.1.1 Key Features

1. Prescription Input
2. Add or remove multiple medicines
3. Generates Unique Prescription Code
4. Search prescription with patient's username
5. Sortable prescription search table
6. Displaying detailed prescription with the option of downloading the prescription as PDF

1.1.2 Key Functionalities

- Doctors can easily make the prescription by first inputting patient's username and then adding medicine information. For now, a maximum of 15 medicines can be added in a single prescription, but the constraints can be removed.
- While inputting username as well as medicine information, everything suggested in real-time so that doctors don't have to write the full information of patients' as well as medicines'.
- While creating the prescription, doctors can delete any medicine as needed.
- Upon the creation of a prescription providing all the necessary information, a unique prescription code is created and it is notified to the doctor. It can be later copied by pressing the 'Prescription Code' button.
- Doctors can search any previously created prescription by going to the View Prescription menu and then searching the prescription by the patient's username. It will then show all the previous available prescriptions.
- After selecting the needed prescription, it will show the detailed information of the prescription in another window. There is also an option to download the prescription as PDF for any kind of external usage.

1.2 The shop management panel

1.2.1 Key Features

1. Add medicine to inventory
2. Update an existing medicine in inventory
3. Delete an existing medicine in inventory
4. Search for a medicine
5. Update account credentials

6. Change inventory warning parameter

7. Sorting medicines according to name,price,expiry date,validity, dose and quantity

8. User guide and FAQs

1.2.2 Key Functionalities

- Medicines are added and stored to a secure database that can be configured to be either local or online based on client requirements. Medicines ,when added to the inventory are cross checked with the existing inventory to see if the batch of medicine matching the expiry date,dose and name already exists or not. If it exists, the new batch is added to the existing record and the quantity and prices are updated only. If no record exists, the medicine is added as a new record.

- User need only fill the fields marked with * in the Add Panel.

- System allows the user to update a medicine by clicking it on the list then pressing update. While still on the update panel, users can update other medicines by entering their index in the 'Select by index' field. The system uses tool tips to indicate the index while selecting and hovering over a medicine.

- While entering a new medicine, the system also records the exact time it was added automatically.

- Users can search for a specific medicine by pressing the search button. This opens up a new column labeled 'Date added'. Users then can search for a medicine using 4 different search criterion. The system supports sub-string searching as well.

- A very important function of the inventory management system is to check the status of a medicine. This happens on a priority based hierarchy. At first the system checks whether the medicine is past the expiry date, comparing it with the system date. If it is still valid, the system checks whether the medicine is low on stock then whether it is out of stock. Based on these checks, the specific medicine in inventory is highlighted with red if it has expired, with yellow if it is low on stock

based on the 'low stock value' and in purple if it is out of stock i.e 0 quantities available.

- The user can set the amount of quantity at which the low stock status is activated using the settings in the menu bar.
- The help button gives a FAQ pop up that allows the users to get accustomed with the system features.
- The UI is layed out to be as intuitive as possible with all essential features and operations visible to the user out of the box.

1.3 The customer/patient's panel

1.3.1 Key Features

1. Sign up as new patient
2. View previous medication history and prescriptions
3. View previous purchase history and receipts
4. Attain PDF copies of the documents
5. Browse generic inventory without prescription
6. Purchase specialized medicines through prescriptions
7. Update and change cart as per liking
8. View products available on different store branches.

1.3.2 Key Functionalities

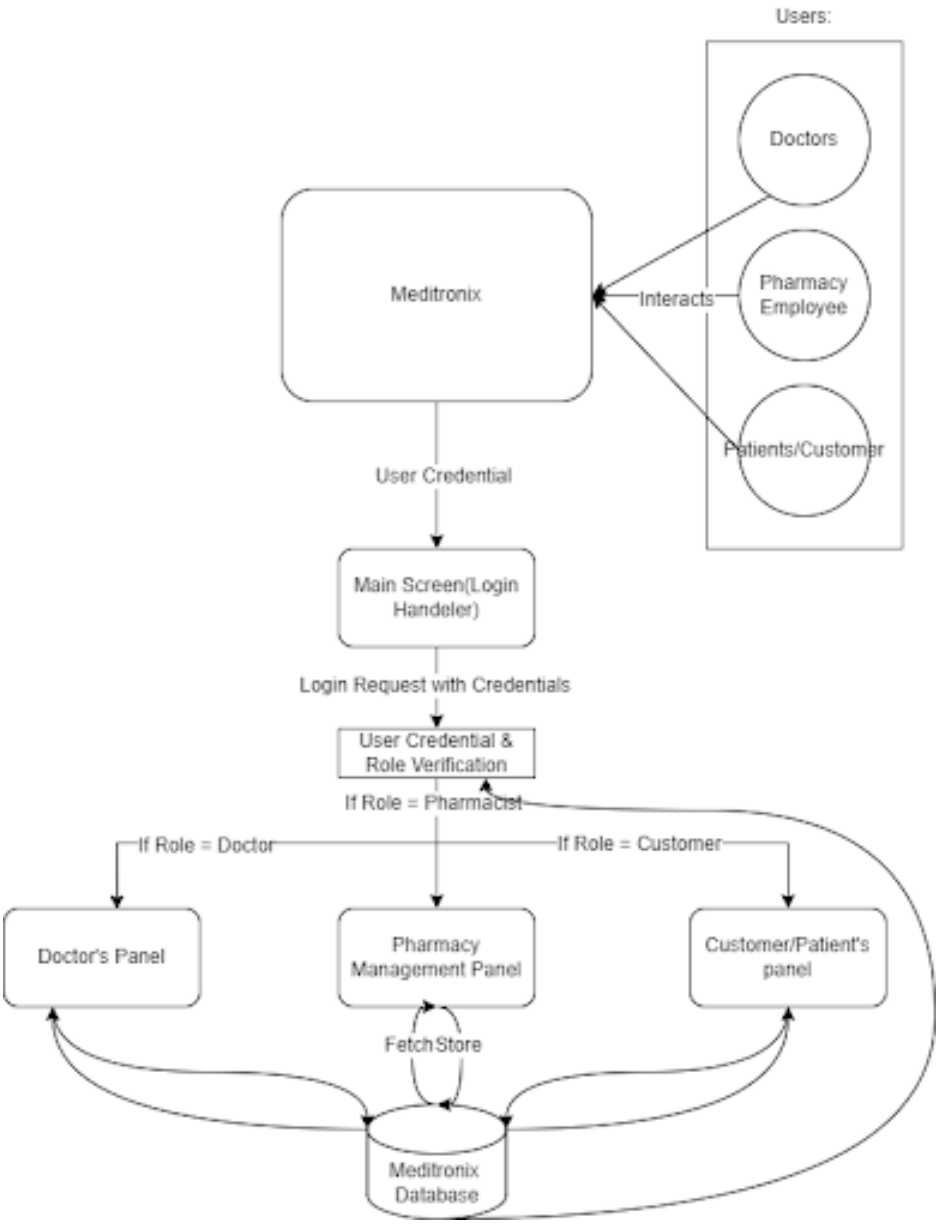
- Patients can sign up by creating a new account from the sign up page just by providing basic identification information.
- The prescriptions are stored in secure databases. Patients can access/view these prescriptions by inserting their unique prescription codes but cannot bring any changes in the allotted medication values.
- Patients can also Generate PDFs of the prescriptions and create printouts using the download PDF button.

- Customers are given access to the generic available inventory of the medicine shop to purchase regular day-to-day drugstore necessities. The available medications as well as dosage, price ,available quantity are shown in a table. Customers can click and add the products to cart before checking out and finalizing the transaction.
- After checkout the stock quantities are updated accordingly.
- Enable patients to purchase medications prescribed by their doctors. Patients need to enter their unique prescription codes in order to access the prescriptions and then purchase according to limit.
- View a list of prescribed medications and their details.
- Control the buying limit, ensuring that no harmful prescription medicine is over-purchased.
- View products based on different store location. Since for demonstration purposes,we are showing the software,this has been employed to simulate buying physically from a different store.
- Track patient billing information.
- Generate invoices and receipts.
- Process payments.

2 System Design

2.1 Overall System Architecture

2.1.1 Context level diagram



The users interact with the system through a main screen which acts as a login handler. Users logging in provide their credentials which are then verified by the system. Based on the user's role (doctor, pharmacist, or patient/customer), the system directs them to a specific user panel.

- Doctors' panels allow a doctor to prescribe medicines to patients, create and save prescriptions and upload them to the database.
- Pharmacy management panels allow a pharmacy employee to manage medicine inventory effectively & efficiently.
- Patients' panels allow patients to manage their prescriptions and by generic & prescribed medicines.

2.2 Chosen Design Patterns and Technologies Used

To design the software, we used JavaFx and Scene Builder to render the GUI and its component controller scripts. For login authentication and data storage,updation,retrieval & deletion, we used MySQL ORM, MySQL drivers, JDBC & MySQL workbench.

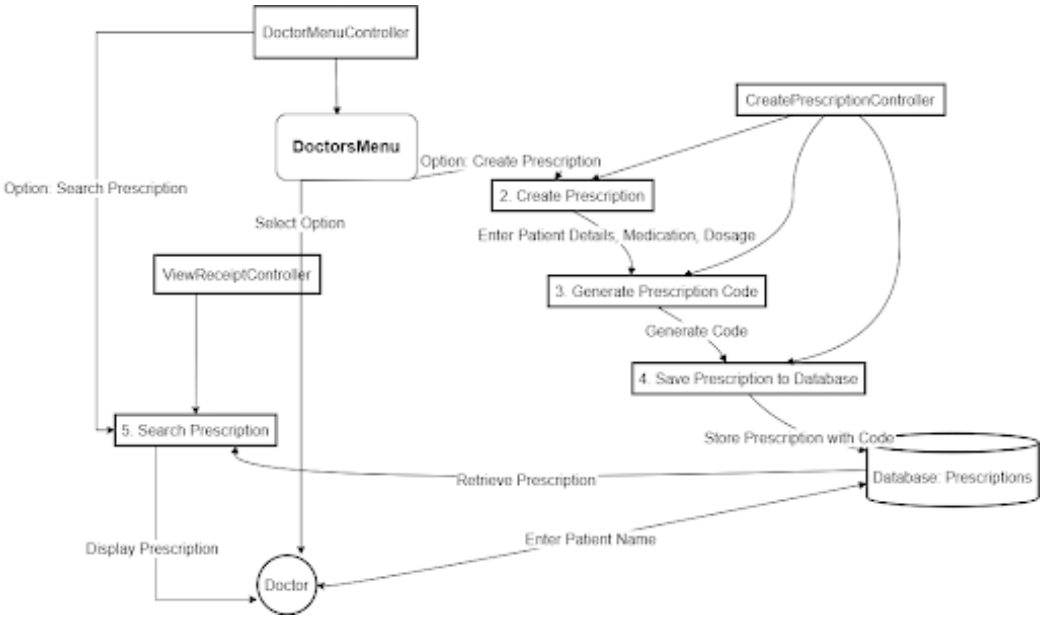
2.3 User Interaction With the System

2.3.1 Doctor's Menu Interaction

The doctor selects options from the Doctor Menu to either create a new prescription or search for an existing one.

Creating a Prescription:

- When the doctor opts to create a prescription, the system collects patient details, medication, and dosage information.
- The system then generates a unique prescription code.
- The prescription details, along with the generated code, are saved to the database.



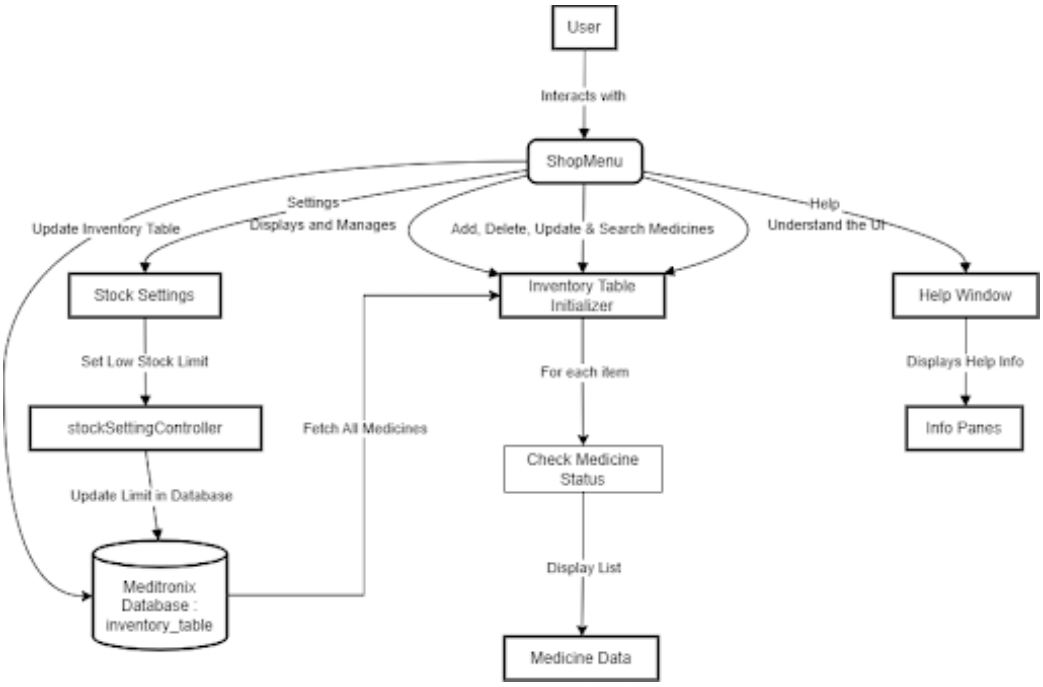
Searching for a Prescription:

- The doctor can search for prescriptions by entering the patient's username.
- The system retrieves the relevant prescription data of that patient from the database and displays it to the doctor.

2.3.2 Shop Menu Interaction

- The **ShopMenu** controller serves as the central hub, coordinating between the UI components and the backend.
- The **stockSettingController** updates stock settings and communicates changes to the **ShopMenu** and the database.
- The **HelpWindow** assists users by providing easily accessible help information.

- The **Database** stores all critical data, including medicine details and stock settings, and is accessed and updated by the controllers as needed.
- The **AddPanelController**,**UpdatePanelController**,**SearchPanel** controller handles operations and processes required when the Add, Update or Search button is clicked. This mounts its corresponding **fxml** graphics onto the **Hbox** panel below the inventory table.



2.3.3 Customer/Patient's Menu Interaction

1. **Buy Prescribed Medicine**
   - **Prescription Code Entry:** The patient enters a unique prescription code provided by their doctor.
   - **Load Prescription:** The system retrieves the prescription details from the database using the entered code.

- **View Prescription Details:** The patient can view the list of prescribed medications along with dosage, quantity, and price.
- **Purchase Medications:** The patient selects the necessary medications and proceeds to purchase them. The system updates the stock quantities accordingly and generates a receipt for the purchase.
- Each entry in the prescription table is marked with red if it is expired in the current location pharmacy inventory or with purple if its out of stock. If the medicine status is okay, then the default row color (white) is used.

2. **Generic Purchase**

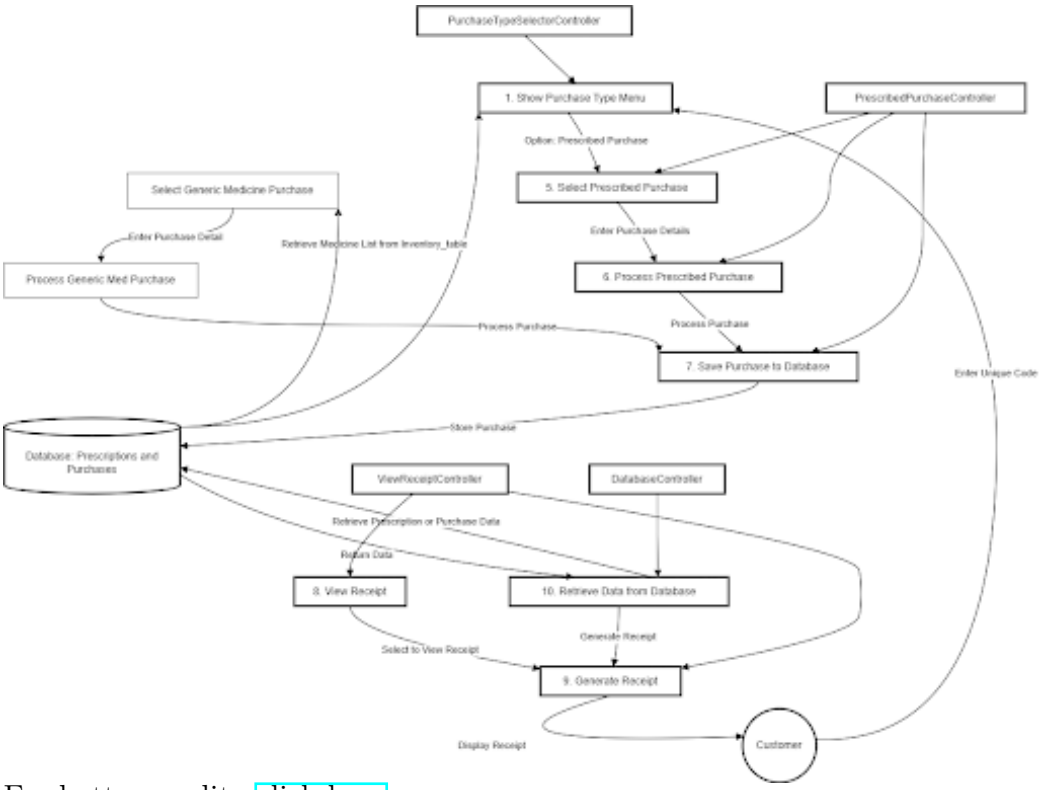
- **View Generic Medicine Inventory:** The customer is shown a list of available generic medications.
- **Select Medications:** The customer selects the desired generic medications, specifying the quantity for each item.
- **Add to Cart and Purchase:** The selected medications are added to a cart. The customer can review the cart and proceed to purchase the items. The system updates the stock quantities and generates a receipt.
- Each entry in the generic medicine table is marked with red if it is expired in the current location pharmacy inventory or with purple if its out of stock. If the medicine status is okay, then the default row color (white) is used.

3. **View Prescriptions**

- **List of Prescriptions:** The customer can view a list of all their past and current prescriptions.
- **Detailed View:** For each prescription, the customer can see detailed information including the date of issue, prescribing doctor, and list of medications.
- **Download PDF:** Patients can download and print the PDF version of the prescriptions.

4. **View Receipts**

- **Detailed Receipt View:** Each receipt includes details such as date of purchase, items bought, quantities, prices, and total amount paid.
- **Download Receipt:** Patients can download PDF of the receipts and make printed copies.



For better quality click here

3 Implementation Details

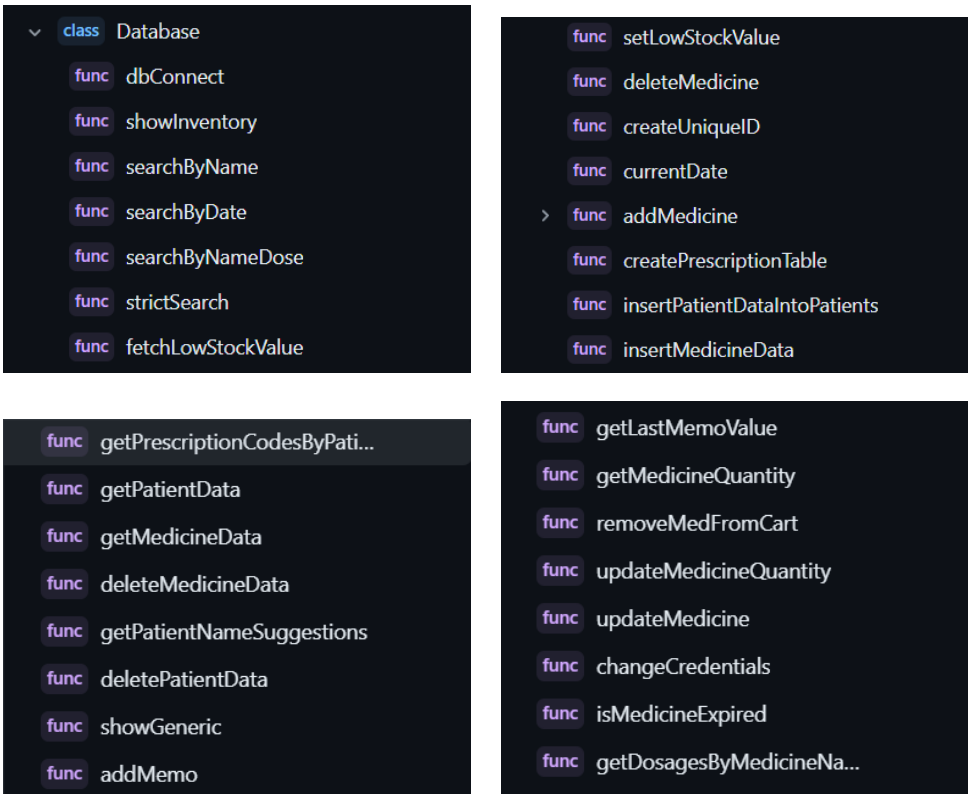
3.1 General Class Implementation Details

This is the Database class used throughout the code base to establish a connection between the cloud database and all functions that are required to run an SQL statement. All database related functions are written in the database java class file with functions having a connection parameter to reduce the need for creating a new connection every time an SQL is executed. Example :

```
1 usage   AhmedRafid3S5
public int fetchLowStockValue(Connection con) throws SQLException{
    Statement stmt = con.createStatement();
    String sql = "SELECT lowStockValue FROM stock_parameters;";
    ResultSet rs = stmt.executeQuery(sql);

   if(rs.next()) {
        return rs.getInt( columnLabel: "lowStockValue");
    }
   else
       return 1; //default value
}
```

```
AhmedRafid3S5 +2 *
public class Database {

    31 usages   AhmedRafid3S5 *
    public Connection dbConnect() {
        try {
            Class.forName( className: "com.mysql.jdbc.Driver");



            //-----------Aiven MySQL connection configuration----------------------
            String host = "meditronix-ahmedrafidx360-5ecd.f.aivencloud.com";
            String port = "22467";
            String databaseName = "meditronix";
            String userName = "avnadmin";
            String password = "AVNS_8kqCc13cKBsjHicTaW5";
            Connection con =DriverManager.getConnection( url: "jdbc:mysql://" + host + ":" + port + "/" + databaseName + "?ssl=

            System.out.println("Connected to database successfully");
            return con;

        }
        catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }

        return null;
    }
```

Following are the all database functions:



For enhanced security of the application, password hashing is implemented using the SHA-256 algorithm. The hashPassword() method first obtains an instance of the MessageDigest class for SHA-256. The method then converts the input password to a byte array using UTF-8 encoding. This byte array is fed to the digest() method of the MessageDigest instance, producing the hashed byte array. Each byte of the hashed array is converted to a hexadecimal string and appended to a StringBuilder object, ensuring that single-digit hex values are prefixed with '0'. The resulting hexadecimal string represents the hashed password.

```
1 usage
private String hashPassword(String password) throws NoSuchAlgorithmException {
    MessageDigest digest = MessageDigest.getInstance( algorithm: "SHA-256");
    byte[] hash = digest.digest(password.getBytes(StandardCharsets.UTF_8));
    StringBuilder hexString = new StringBuilder();
    for (byte b : hash) {
        String hex = Integer.toHexString( i: 0xff & b);
        if (hex.length() == 1) hexString.append('0');
        hexString.append(hex);
    }
    return hexString.toString();
}
```

3.2 Implementation Details of the Doctor's Panel

Key Functionalities:

1. Prescription Input:
    - Users can fetch all relevent patient's info by entering username.
    - Medicine details can be added, including name, dosage, quantity, and frequency.
2. Medicine Addition/Removal:
    - Users can add multiple medicines to a prescription.
    - Available medicine suggestion (name, dosage).
    - Medicines can be removed if added by mistake or no longer needed.
3. Unique Prescription Code Generation:
    - Each created prescription is assigned a unique code for easy retrieval and tracking.
4. Search Functionality:
    - Users can search for prescriptions using the patient's username.
    - The search results display all relevant prescriptions with sortable columns.

5. Prescription Details Display and PDF Generation:

    - Detailed view of the selected prescription, with an option to download it as a PDF.

Relevant Code Snippets and Explanations:

The following function is used to generate the Unique Prescription code. It generates a four-character random prescription code using uppercase alphabets and digits. It employs a StringBuilder to construct the code by appending randomly selected characters from the defined set in a loop. This approach ensures efficiency and readability of the code:

```
1 usage
private String generatePrescriptionCode() {
    String characters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    StringBuilder code = new StringBuilder();
    for (int i = 0; i < 4; i++) {
        code.append(characters.charAt(random.nextInt(characters.length())));
    }
    return code.toString();
}
```

The following function fetches a list of medicine names matching the input string from the database and returns it as a list. It connects to the database, performs a search, and adds the results to the suggestions list, handling any SQL exceptions that occur.

```
1 usage
private List<String> getSuggestions(String input) {
    List<String> suggestions = new ArrayList<>();

    try (Connection con = database.dbConnect()) {
        ResultSet rs = database.searchByName(con, input);
        while (rs.next()) {
            String name = rs.getString( columnLabel: "Name");
            suggestions.add(name);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    return suggestions;
}
```

The following function displays these suggestions in a ListView positioned below the input field, allowing the user to select a suggestion. Upon selection, the input field is updated with the chosen suggestion, and the suggestion list is hidden.

```
1 usage
private void showSuggestions(List<String> suggestions) {
    // Create a new ListView to display the suggestions
    ListView<String> suggestionsListView = new ListView<>();
    suggestionsListView.getItems().addAll(suggestions);

    // Set the size and position of the ListView
    suggestionsListView.setPrefWidth(MedicineName.getWidth());
    suggestionsListView.setPrefHeight(100);
    suggestionsListView.setLayoutX(MedicineName.getLayoutX());
    suggestionsListView.setLayoutY(MedicineName.getLayoutY() + MedicineName.getHeight());

    // Add an event handler to handle selection of a suggestion
    suggestionsListView.setOnMouseClicked(event -> {
        String selectedSuggestion = suggestionsListView.getSelectionModel().getSelectedItem();
        if (selectedSuggestion != null) {
            MedicineName.setText(selectedSuggestion);
            hideSuggestions();
        }
    });

    // Add the ListView to the scene
    ((Pane) MedicineName.getParent()).getChildren().add(suggestionsListView);
}
```

The following function enables users to save a PDF file via a file chooser dialog. It sets a filter to display only PDF files, ensuring that users select the appropriate file type. When the user selects a file location, it invokes the createPDF() function to generate and save the PDF. The dialog is displayed using the current window context (dnldPDF.getScene().getWindow()), providing a seamless user experience for downloading PDFs.

```
@FXML
void downloadPDF(ActionEvent event) {
    FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter( s: "PDF files (*.pdf)", ...strings:
    File file = fileChooser.showSaveDialog(dnldPDF.getScene().getWindow());

    if (file != null) {
        createPDF(file);
    }

}
```

3.3 Implementation Details of the Shop Management Panel

Key Functionalities:

1. Users can add a medicine to inventory:
    - Users can choose to fill up necessary fields only, specific medicines that don't require an info field is set to a default character.
    - System checks if the same batch of med is already present in the inventory and decides whether to add it as a new record or add to update the existing inventory.
2. Users can delete a medicine from inventory
3. Users can update an existing medicine:
    - Update can be done by selecting an item in list and pressing update.
    - Update may also be done while on the update panel by selecting a medicine and loading its info by index.
4. Users can search for a medicine:
    - Search by Name or Generic drug name.
    - Search by Name & Dose.
    - Search by Date Added.
    - Search by Name,Dose & Date Added (*Strict Search Feature).

Relevant Code Snippets and Explanations:

```
15 usages
public static final ShopMenu instance = new ShopMenu();
12 usages
ObservableList<Medicine> list = FXCollections.observableArrayList(
  //new Medicine("Lumona","125mg","20/12/25","Generic", 100.0F, 10.0F, 95.0F),
    //    new Medicine("Lexotanil","125mg","20/12/25","Specialized", 100.0F, 10.0F, 95.0F)
);

//Implementing singleton pattern so any class can access the Inventory properties
18 usages  AhmedRafid3S5
public static ShopMenu getInstance() { return instance; }

1 usage  AhmedRafid3S5
public int getLowStockLimit(){return instance.lowStockLimit;}
1 usage  AhmedRafid3S5
public void setLowStockLimit(int value) { instance.lowStockLimit = value; }
```

The ShopMenu class is created as a singleton class to ensure only one instance of its object is created and used throughout the run time of the application. This is to ensure that all fxml that gets attached as a child to a component is rendered onto the ShopMenu window and not any other null object of the class when referencing the ShopMenu class. In this way, the addpanel controller, updatepanel controller and searchPanel controller makes use of the singleton instance of the ShopMenu to mount its fxml pane onto the menu. All updates to the inventory list are made to the inventory list object of this static object.

```
//Highlight rows which have expired or low on stock
3 usages  AhmedRafid3S5
public void detectStockEmpty()
{
   inventoryTable.setRowFactory(tv -> updateItem(item, empty) → {
         super.updateItem(item, empty);
         if (item == null || empty) {
            setStyle("");
         } else {
            //Set priority with higher priority given to expiry date,then to stock level
            // If quantity is 0, color the row red
            if(medExpired(item.getExpiry())){
               item.setStatus("Expired!");
               setStyle("-fx-background-color: #c94059;-fx-font-weight: bold;");
            }
            else if (item.getQuantity() == 0) {
               item.setStatus("Out of Stock");

               setStyle("-fx-background-color: #8a61bd;-fx-font-weight: bold;");
            }
            else if (item.getQuantity() <= instance.lowStockLimit){
               item.setStatus("Low Stock");
               setStyle("-fx-background-color: #d2b939;-fx-font-weight: bold;");
            }
            else {
               item.setStatus("Valid");
               setStyle(""); // Default style
            }
         }
   });
}
```

One of the most important function that checks the whole inventory for expired medicines then its stock value and sets item status accordingly every time the application is launched.

```
1 usage  AhmedRafid3S5
public String createUniqueID()
{
    LocalDateTime currentTime = LocalDateTime.now();

    // Define a DateTimeFormatter for the desired timestamp format
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

    // Format the LocalDateTime to a string using the formatter

    return currentTime.format(formatter);

}
```

This function automatically assigns the exact time the medicine was added as the primary key for that record of medicine added to the database. This ensures that all record entered into the database are unique and insertion never fails.

```
AhmedRafid3S5
public Medicine(ResultSet rs) throws SQLException {
    this.Name = rs.getString( columnLabel: "Name");
    this.Dose = rs.getString( columnLabel: "Dose");
    this.Expiry = rs.getString( columnLabel: "Expiry");
    this.Type = rs.getString( columnLabel: "Type");
    this.price = rs.getFloat( columnLabel: "Selling_price");  // Assuming "
    this.Quantity = rs.getFloat( columnLabel: "Available_Quantity");
    this.UnitCost = rs.getFloat( columnLabel: "unit_cost");
    this.serial_id = rs.getString( columnLabel: "serial_id");

    //set to a default status since status is handled on a system level
    this.status = "Undefined";
}
```

The medicine class method takes a row from the resultset and creates a medicine object. A resultset of the `inventory_table` is passed as an argument to this function.

The addMedicine function under the database class (lines 185 to 279) is crucial for adding a medicine to the inventory list. Due to its large size, the code is not included here but refer to it by the lines mentioned here. To summarize, this function employs the following algorithm:

1. Check for Existing Medicine:
    - Use a PreparedStatement to set the parameters (name, dose, expiry, type) for the SQL query.
    - Execute the query and store the result in ResultSet rowsSelected.
2. If Medicine Exists:
    - Check if the query returns any results (i.e., a medicine with the same criteria already exists).
    - If a match is found, prepare an SQL UPDATE statement to update the existing record's `Selling_price`, `Available_Quantity`, and `unit_cost`.

- Set the parameters for the UPDATE statement using values from the Medicine object and the existing `Available_Quantity` from the result set.
- Execute the update statement.
- If the update is successful `(rowsAffected > 0)`, set the added message to indicate that the medicine was added to an existing record.

3. If Medicine Does Not Exist:
    - If no matching medicine is found, generate a unique serial ID for the new medicine record.
    - Prepare an SQL INSERT statement to add a new record to the `shop_inventory` table.
    - Set the parameters for the INSERT statement using values from the Medicine object and the generated serial ID.
    - Execute the insert statement.
    - Set the added message to indicate that a new medicine was added to the inventory.
4. Update UI and Hide Status Label:
    - Set the tellStatus label text to the added message.
    - Make the label visible and wrap the text.
    - Create a Timeline to hide the label after 1.5 seconds.
    - Play the timeline once to hide the label.
5. Refresh Shop Menu:
    - Call ShopMenu.getInstance().refreshList() to refresh the list of medicines in the shop menu.

This algorithm ensures that if a medicine with the same name, dose, expiry, and type already exists, its quantity and other details are updated. If it doesn't exist, a new record is added to the inventory. The status message is displayed to the user and then hidden after a brief period.

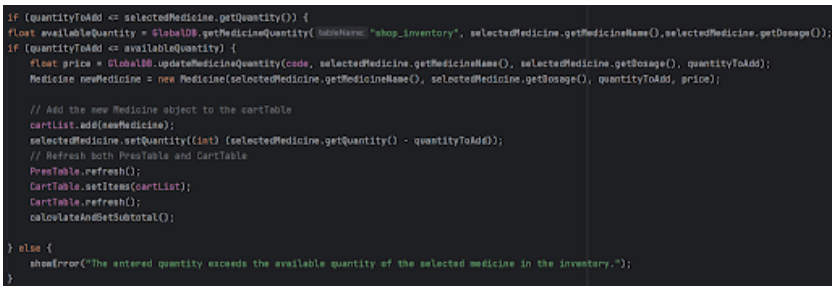
3.4 Implementation Details of the Customer/Patient's Panel

Key Functionalities:

1. Sign Up:
    - Patients can sign up by creating a new account from the sign up page just by providing basic identification information.
2. Prescription Management:
    - The prescriptions are stored in secure databases. Patients can access / view these prescriptions by inserting their unique prescription codes but cannot bring any changes in the allotted medication values
    - Patients can also Generate PDFs of the prescriptions and create printouts using the download PDF button.
3. Buying Generic Medications:
    - Customers are given access to the generic available inventory of the medicine shop to purchase regular day-to-day drugstore necessities. The available medications as well as dosage, price, available quantity are shown in a table. Customers can click and add the products to cart before checking out and finalizing the transaction.
    - After checkout the stock quantities are updated accordingly.
4. Buying Prescribed Medications:
    - Enable patients to purchase medications prescribed by their doctors. Patients need to enter their unique prescription codes in order to access the prescriptions and then purchase according to limit.
    - View a list of prescribed medications and their details.
    - Control the buying limit ensuring no harmful prescription medicine is over purchased.
5. Billing and Payments:

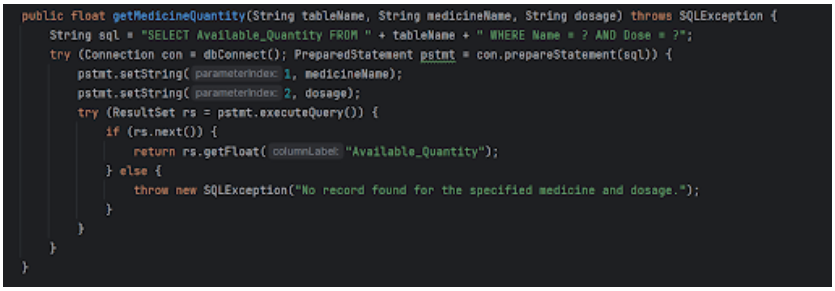
- Track patient billing information.
- Generate invoices and receipts.

Relevant Code Snippets and Explanations:



```
if (quantityToAdd <= selectedMedicine.getQuantity()) {
float availableQuantity = GlobalDB.getMedicineQuantity( tableName: "shop_inventory", selectedMedicine.getMedicineName(),selectedMedicine.getDosage());
if (quantityToAdd <= availableQuantity) {
    float price = GlobalDB.updateMedicineQuantity(code, selectedMedicine.getMedicineName(), selectedMedicine.getDosage(), quantityToAdd);
    Medicine newMedicine = new Medicine(selectedMedicine.getMedicineName(), selectedMedicine.getDosage(), quantityToAdd, price);

    // Add the new Medicine object to the cartTable
    cartList.add(newMedicine);
    selectedMedicine.setQuantity((int) (selectedMedicine.getQuantity() - quantityToAdd));
    // Refresh both PresTable and CartTable
    PresTable.refresh();
    CartTable.setItems(cartList);
    CartTable.refresh();
    calculateAndSetSubtotal();

} else {
    showError("The entered quantity exceeds the available quantity of the selected medicine in the inventory.");
}
```

In the prescribed view section , when a customer tries to add a medicine to cart, the program checks whether the selected quantity is within the amount of his purchase allowance. It then updates the table and cart accordingly. The getQuantity function fetches this information from the database. This function is implemented in the Database Class.



```
public float getMedicineQuantity(String tableName, String medicineName, String dosage) throws SQLException {
    String sql = "SELECT Available_Quantity FROM " + tableName + " WHERE Name = ? AND Dose = ?";
    try (Connection con = dbConnect(); PreparedStatement pstmt = con.prepareStatement(sql)) {
        pstmt.setString( parameterIndex: 1, medicineName);
        pstmt.setString( parameterIndex: 2, dosage);
        try (ResultSet rs = pstmt.executeQuery()) {
            if (rs.next()) {
                return rs.getFloat( columnLabel: "Available_Quantity");
            } else {
                throw new SQLException("No record found for the specified medicine and dosage.");
            }
        }
    }
}
```

```
private void calculateAndSetSubtotal() {
    float subtotal = 0.0f;
    for (Medicine medicine : cartList) {
        subtotal += medicine.getQuantity() * medicine.getPrice();
    }
    // Update the subtotal label with the calculated subtotal
    subtotalLabel.setText(String.format("Subtotal:        %.2f   Tk", subtotal));
}
```

Every time the cart is updated the program recalculates the subtotal and shows it to the purchaser.

```
void LoadPresButtonPressed(ActionEvent event) {
    code = PresCode.getText().trim();

    ObservableList<MedicineDataPrescription> data = GlobalDB.getMedicineData(code);
    if (data == null || data.isEmpty()) {
        showError("Invalid prescription code, no data found.");
    } else {
        PresTable.setItems(data);
    }
    setPatientName();
}
```

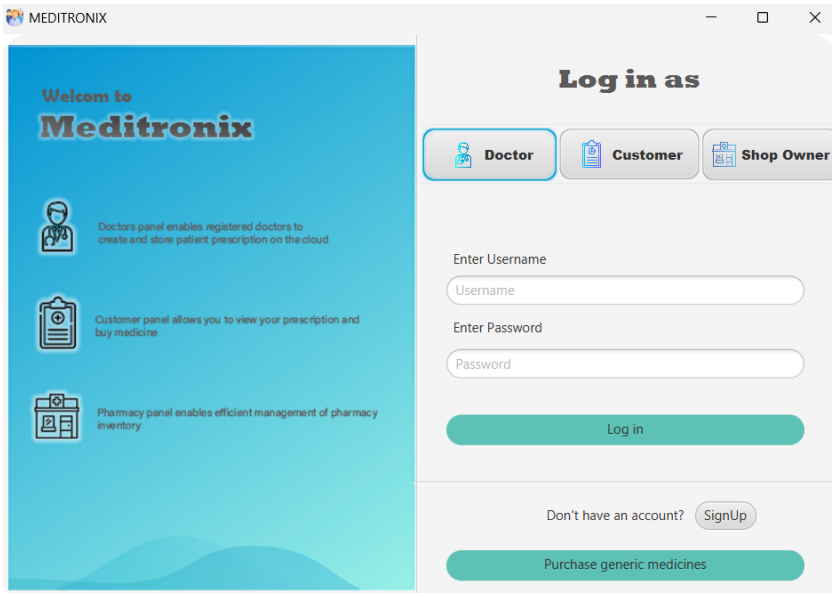
This code section fetches the prescription from database by using the unique code provided by the customer

```
try (Connection con = GlobalDB.dbConnect()) {
    if (GlobalDB.isMedicineExpired(selectedMedicine.getName(), selectedMedicine.getDose(), con)) {
        showAlert( title: "Expired Stock", message: "The selected medicine is expired.");
        return;
    }
}
```

The code also needs to check the expiry date of the medications / products before allowing the customers to add it to cart.

4 GUI Design

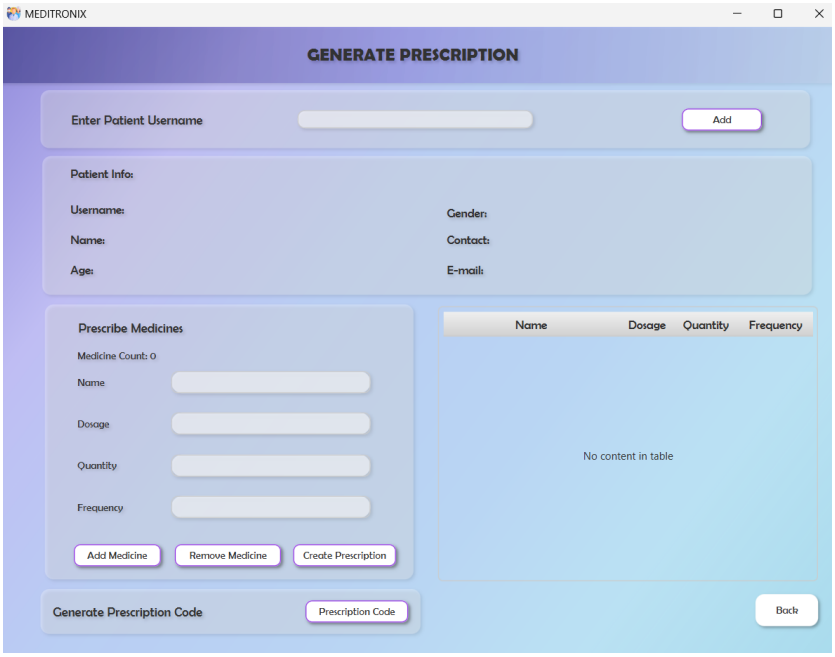
Login Page:



While logging in users need to select one of 3 roles which are neatly placed at the top below the bold letters to draw user attention. After selecting a role, users can log in like any other login systems. For a better understanding out of the box, the 3 roles are concisely described to the left.

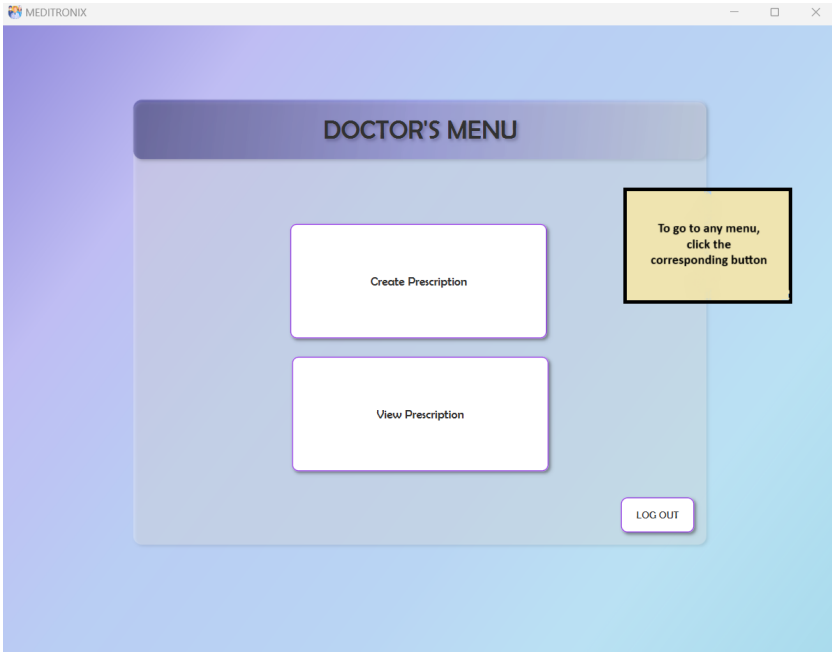
4.1 Doctor's Panel

Generate Prescription Page:

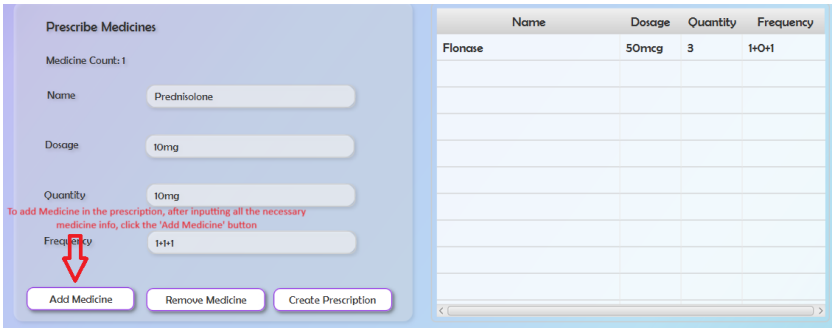


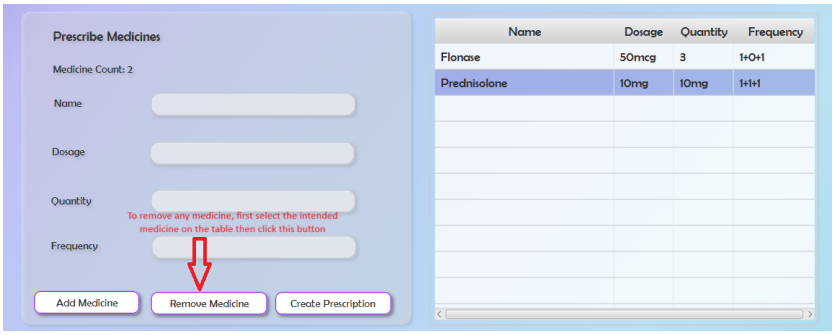
This is the snippet of one page among the two menus in the Doctor's panel. By looking at this single page, the UI layout, design and the ideology it was employed can be easily comprehended. A type of 'Glass Morphism' ideology was chosen to design the Doctor's Menu. Here, the panels and sections, such as "Enter Patient Info" and "Prescribe Medicines," utilize the semi-transparent backgrounds with a blurred effect, giving the appearance of frosted glass transparent layers. Thus it creates a 3D effect where components appear to float above others. Also, all the elements including text-fields, buttons and panels, have rounded corners with drop shadow, which is a common feature in glass morphism.

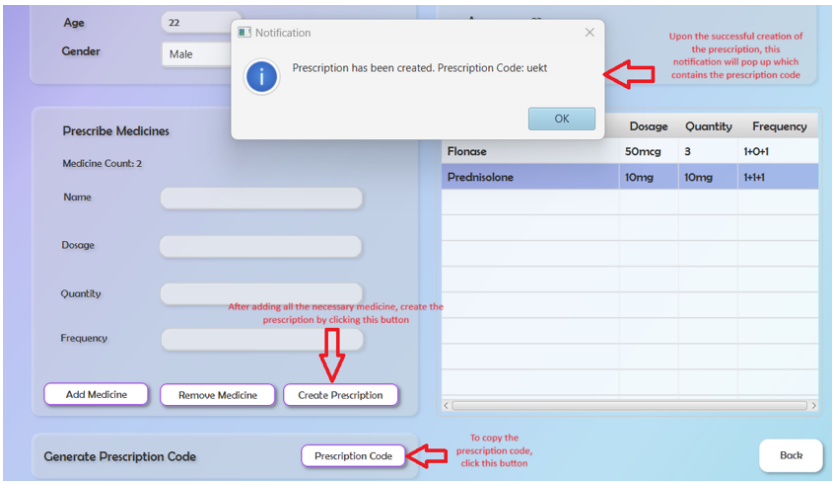
User Navigation:

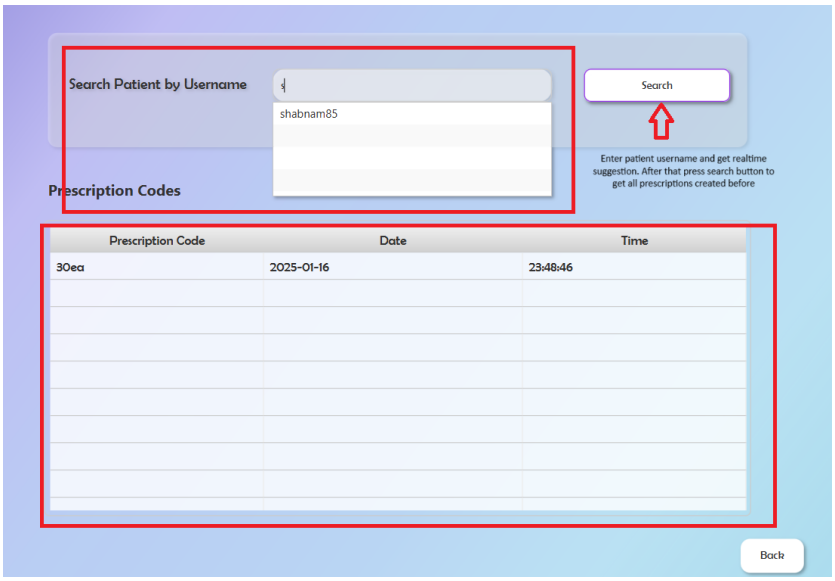

MEDITRONIX
DOCTOR'S MENU
Create Prescription
View Prescription
To go to any menu, click the corresponding button
LOG OUT

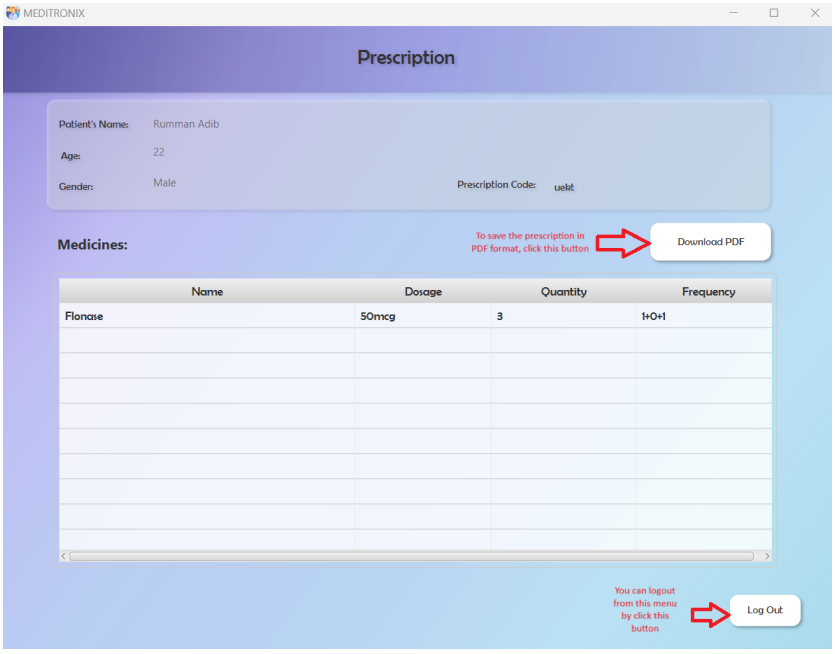

Enter username and get realtime suggestion
arif2000
Add
Patient Info:
Username: arif2000
Name: Arif Khan
Age: 24
Gender: Male
Contact: +8801798765432
E-mail: arif.khan@example.com
Press Add button to add patient info in the prescription


Prescribe Medicines
Medicine Count: 1
Name
Prednisolone
Dosage
10mg
Quantity
10mg
To add Medicine in the prescription, after inputting all the necessary medicine info, click the 'Add Medicine' button
Frequency
1+1+1
Add Medicine
Remove Medicine
Create Prescription
Name
Dosage
Quantity
Frequency
Flonase
50mcg
3
1+0+1


Prescribe Medicines
Medicine Count: 2
Name
Dosage
Quantity
Frequency
To remove any medicine, first select the intended medicine on the table then click this button
Add Medicine
Remove Medicine
Create Prescription
Name
Dosage
Quantity
Frequency
Flonase
50mcg
3
1+0+1
Prednisolone
10mg
10mg
1+1+1

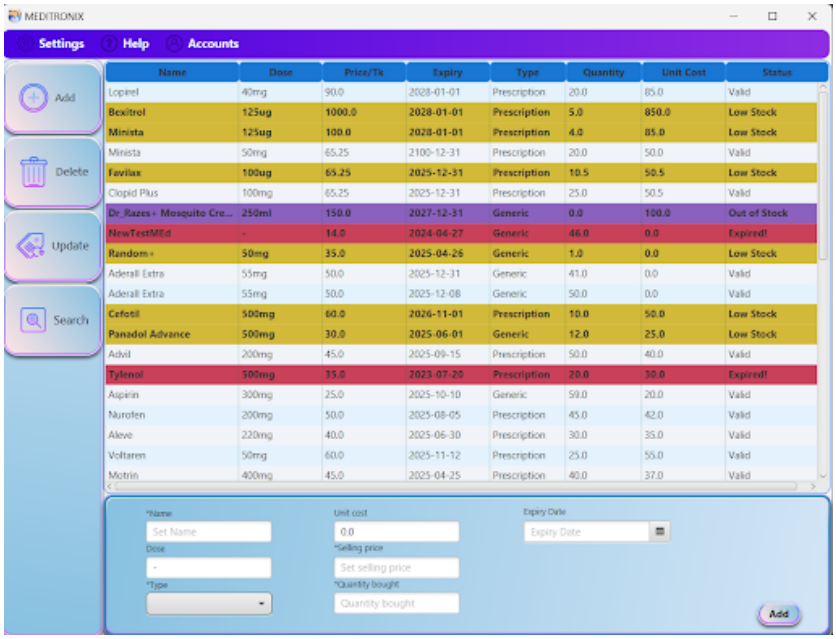

Age
22
Gender
Male
Notification
Prescription has been created. Prescription Code: uekt
OK
Upon the successful creation of the prescription, this notification will pop up which contains the prescription code
Prescribe Medicines
Medicine Count: 2
Name
Dosage
Quantity
Frequency
After adding all the necessary medicine, create the prescription by clicking this button
Add Medicine
Remove Medicine
Create Prescription
Generate Prescription Code
Prescription Code
To copy the prescription code, click this button
Back
Dosage
Quantity
Frequency
Flonase
50mcg
3
1+0+1
Prednisolone
10mg
10mg
1+1+1


Search Patient by Username
s
shabnam85
Search
Enter patient username and get realtime suggestion. After that press search button to get all prescriptions created before
Prescription Codes
Prescription Code
Date
Time
30ea
2025-01-16
23:48:46
Back


MEDITRONIX
Prescription
Patient's Name:
Rumman Adib
Age:
22
Gender:
Male
Prescription Code:
uelit
Medicines:
To save the prescription in PDF format, click this button
Download PDF
Name
Dosage
Quantity
Frequency
Flonase
50mcg
3
1+0+1
You can logout from this menu by click this button
Log Out

4.2 Shop Management Panel

Shop Menu:



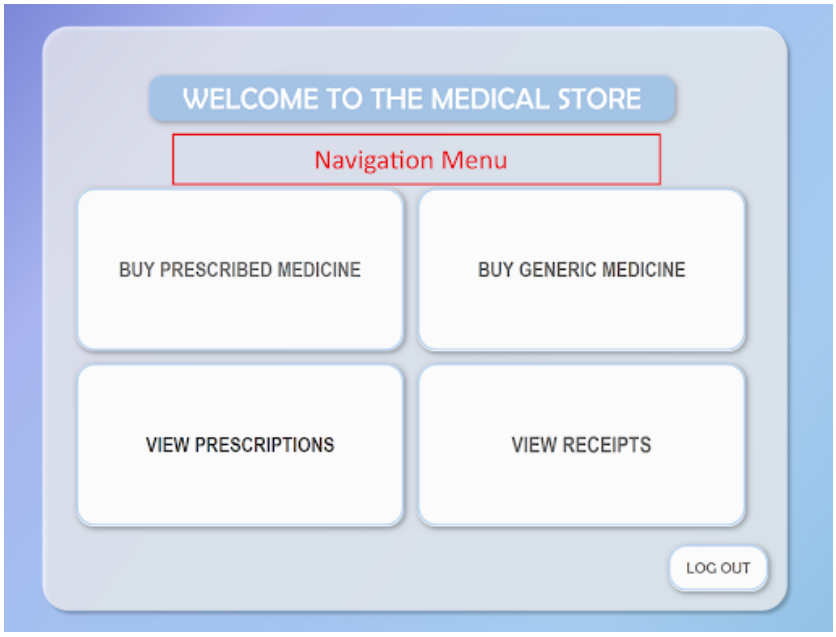
While designing the shop menu, it was intended to keep the GUI design a bit different yet similar to the doctor and patient menu to define a separate user experience for the pharmacists that are not directly related to the doctor-patient relationship yet compliment them together to provide a complete package. In doing so, a type of 'Glass Morphism' ideology was chosen to build up the GUI for the shop inventory management interface.This can be felt from the design where the buttons appear as frosted levitating glass slabs on top of the panels whose edges also mimic that of glass slabs.

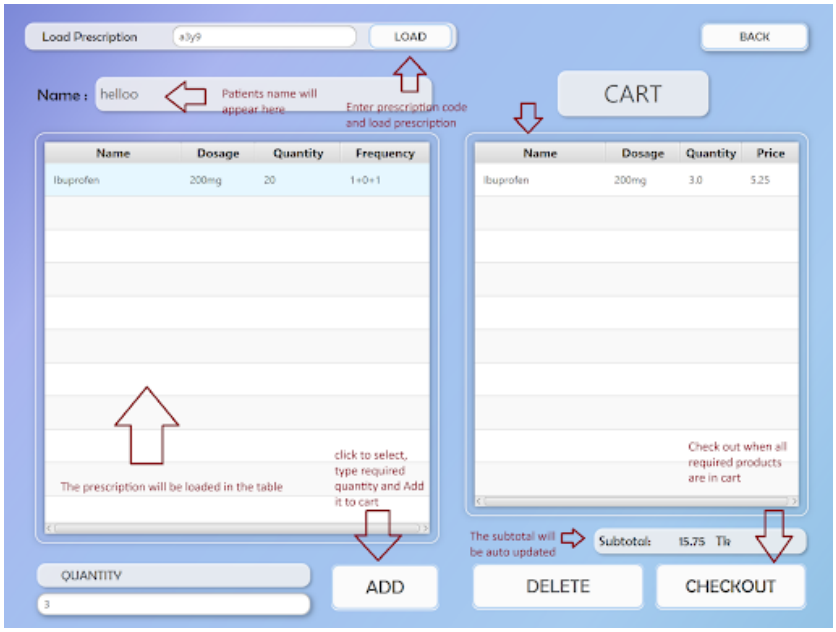
The menu bar was made flat in design with a darker and more pronounced color gradient to draw contrast. Lastly, use of gradient icons & color schemes is used throughout the system in all 3 panels to get rid of the monotonous atmosphere. Also the color scheme chosen is such that it relates to medical software and medicine.

The basic layout of the shop menu is provided below.

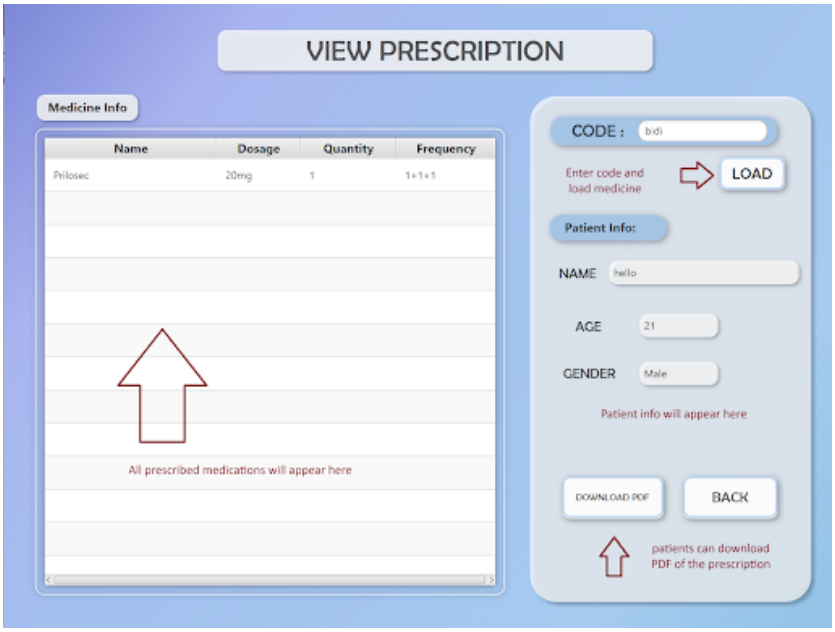
4.3 Customer/Patient's Panel

User Navigation:






BACK
GENERIC MEDICINES
CART
Name
Dosage
Price
Quantity
Prednisolone
5mg
7.0
5.0
Cetirizine
10mg
6.75
2.0
Levothyroxine
100mcg
8.5
0.0
Ibuprofen
200mg
5.25
3.0
Metformin
500mg
8.75
3.0
Name
Dosage
Quantity
Price
Cetirizine
10mg
2.0
6.75
Subtotal:
13.50 Tk
QUANTITY
2
ADD
DELETE
CHECKOUT


VIEW PRESCRIPTION
Medicine Info
Name
Dosage
Quantity
Frequency
Prilosec
20mg
1
1+1+1
All prescribed medications will appear here
CODE :
bidi
Enter code and load medicine
LOAD
Patient Info:
NAME
hello
AGE
21
GENDER
Male
Patient info will appear here
DOWNLOAD PDF
BACK
patients can download PDF of the prescription

5 Testing and Evaluation

5.1 Doctor's Panel

To test the Doctor's panel, **Unit Testing** had been used to ensure that individual components or units of the application work as expected. There is a potential bug in the search by patient's name to get the prescription, which is: it shows all the same names instead of showing one single patient name.

Unit testing:

- Prescription Input: The functions for inputting patient and medicine information were tested.
- Medicine Addition/Removal: Verified that medicines could be correctly added or removed.
- Unique Prescription Code Generation: Checked that each prescription generated a unique code.
- Search Functionality: Ensured that searching by patient's name retrieved all the correct prescriptions.

- PDF Generation: Validated the process of generating and downloading the prescription as a PDF.

Challenges in isolating units were faced due to dependencies on other modules or external libraries and it was addressed by using a mockup framework to simulate the dependencies.

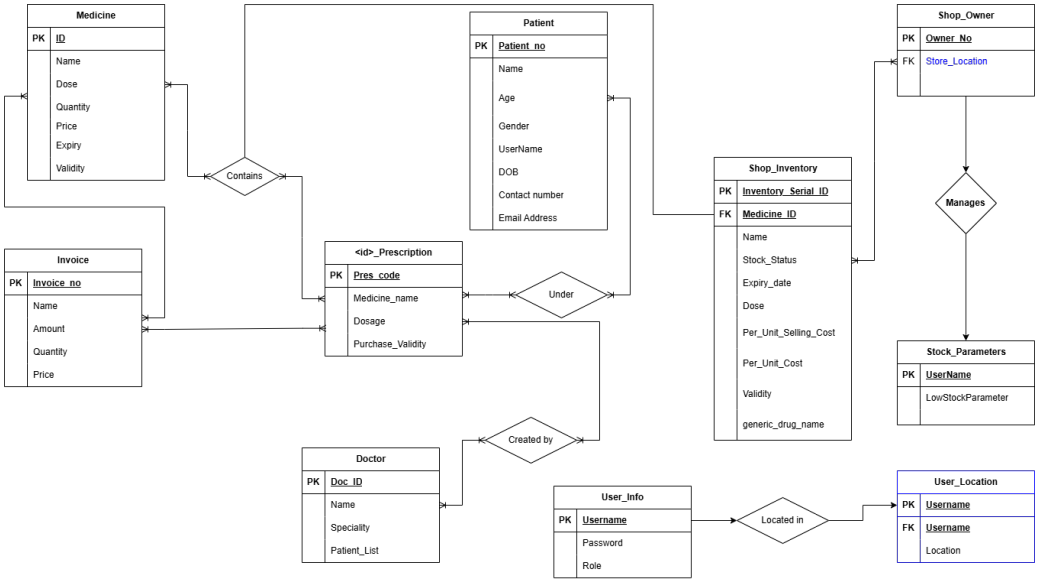
5.2 Shop Management Panel

Integration testing was used to test out whether all components were interacting properly with each other. Each add,delete,update & search requests were properly observed and cross checked with the SQL database to ensure changes were being reflected properly in any database environment, local or online. Unintended effects on UI were addressed using integration testing as well.

5.3 Customer/Patient's Panel

Unit Testing was done in this menu to check and correct all the functionalities (Prescription loading, inventory loading, expiry checking, cart updation, quantity validation etc). The main challenge here was to retrieve and process the data provided by both Shop owner and Doctor's panel. Integrating and utilizing the codes of the other contributors proved to be complex and time consuming.

6 Entity Relationship Diagram



7 Conclusion

7.1 key Achievements & Successful Implementation of Functionalities

- System is compatible with both cloud & local databases.
- Cloud database integration allows for seamless transfer of data, securely, between doctor, patient and affiliated pharmacies.
- Successful implementation of stock parameter setting, a setting that allows users to customize the limit for low stock value below which an item is given the status ‘Low stock’ and highlighted in yellow.
- SHA-256 Algorithm has been employed for advanced security measures.
- Automated patient information fetching by inputting username
- Unique Prescription Code for each prescription.
- Successfully implemented the logic for adding a medicine.

- Successfully implemented the status checking function, the most crucial feature that alerts shop owners about expired medicine and stock status.
- Implemented medicine status that is checked by the system to ensure expired medicines can't be added to a customer's cart.
- Successfully implemented the logic for suggesting available medicine(s) along with their dosage.
- Efficient searching of patients and their prescription thus prescriptions are readily available just by inputting username.
- Successfully implemented the PDF download option for the prescriptions.
- Implementation of an intuitive & attractive UI design.

7.2 Limitations or Future Improvements

7.2.1 Doctor's Panel

- Implement advanced search features such as filtering by date, prescription code, or other patient identifiers to improve search accuracy.
- Splitting the database into smaller, more manageable pieces to improve read and write speeds.

7.2.2 Shop Management Panel:

- Implement a feature to record the distributor/ sales persons name along with the name of medicine they sell.
- Implement a new tab to keep track of registry and day to day sales & expenses and provide shop sales summary.
- Integrate barcode scanning hardware with software to automate & streamline the adding of medicine batches to inventory.

7.2.3 Customer/Patient's Panel:

- Introduce advanced and reliable identification features like Biometrics or NID / Passport verification for a more secure experience.
- Extensive log of medical history including diagnosis, surgeries and other important information.
- Integrating payment gateway for transactions and allowing patients to save payment information for quick purchases.

8 External Library

- To generate PDF of the Prescriptions and Receipts, 'iText' library was used.
- ControlFx jar file was required in addPanel and updatePanel to enable autocomplete textfields.