Medical Billing App

Problem:

Managing the operations of a pharmacy can be complex and time-consuming. Manual billing leads to errors, takes time, and can lead to discrepancies in sales records. Slow billing can also result in customer dissatisfaction and revenue loss. Pharmacy staff struggles to know exactly when to restock or remove items, resulting in inventory inefficiencies. The pharmacy owner is unable to keep records of daily and monthly sales. This app will help manage the pharmacy and reduce human error.

Objectives:

The purpose of building this project is to provide a simple billing process which will help generate quick and accurate bills. It will help users maintain stock for medicine and other pharmacy products and generate daily sales reports. It will also help maintain financial statements for the pharmacy and provide alerts when products are low in stock.

Goals:

1. Develop a quality desktop software application that is efficient and accurate.

2. Provide a user-friendly interface.

3. Ensure better record management.

4. Reduce the time required to generate bills.

5. Improve stock management for medicines and other pharmacy products.

6. Provide stock updates and notifications for low stock.

7. Ensure high security and optimization in both functional and non-functional aspects.

8. Deliver enhanced performance.

Stakeholders:

1. Owner of the Pharmacy

2. Cashier of the Pharmacy

3. Stock Manager

Requirements:

FR-1:

User authentication via specific username and password.

FR-2:

Forget password using a specific 6-digit code received on company email.

FR-3:

Authentication of new users via username, password, and specific 6-digit verification code on company email.

FR-4:

Allows the user to select the desired quantity of required products (if available) and generate a bill according to respective prices.

FR-5:

Allow owner to add new users.

FR-6:

Allows the user to manage previous users.

FR-7:

Allows the user to view available stock

FR-8:

Allow user to add new products to the stock.

FR-9:

Supports categorization of products by type, brand, or medical category for easier management.

FR-10:

Notify users when a product falls below the defined stock threshold.

FR-11:

Displays the sales report for the overall day.

FR-12:

Allows users to generate weekly and monthly sales reports for comprehensive insights.

FR-13:

Includes a search bar with filters to quickly locate products by name, category, or stock level.

FR-14:

 Enables exporting sales reports or inventory lists to formats like CSV or PDF.

FR-15:

Allow user to sign out

FR-16:

User should have role based-access.

Non-Functional Requirements:

- **NRF-1: Performance**:
  - o Quick load time (under 2 seconds).
  - o Fast operations (under 3 seconds for billing and stock management).
- **NRF-2: Scalability**:
  - o Supports growing users
- **NRF-3: Security**:
  - o Secure authentication (password hashing, 2FA if needed).
  - o Encrypt sensitive data (in transit and at rest).
  - o Role-based access control for different user types.
- **NRF-4: Availability**:
  - o 99.9% uptime, with backups and redundancy in place.
- **NRF-5: Reliability**:
  - o Bug-free with logging for quick issue resolution.
  - o Handles edge cases like out-of-stock or incorrect input.
- **NRF-6: Usability**:
  - o Intuitive UI, accessible design, and helpful tooltips.
- **NRF-7: Maintainability**:
  - o Modular, easy-to-update code with clear documentation.

Prototypes:

1. **Launching page**

Your Health, Our Priority

2. **Login Page:**

Login to Get Started

Username

Password

Create Account?   **Forget Password**

Login

3. **Sign Up Page:**

LOREM IPSUM

Create account to Get Started

Username

○ Owner   ○ Cashier   ○ Manager

Sign in?

Next

LOREM IPSUM

Password

Confirm Password

Sign in?

Sign Up

4. **Owner Dashboard Page**

Owner Dashboard

Dashboard

Manage Users

View Inventory

Generate Bill

Sales Report

Sign Out

Welcome, Owner

Low Stock Alert

5 products are out of stock

View Details

View Inventory

View available products and manage products

View Stock

Manage Users

View and edit users

Manage

5. **Cashier Dashboard Page**

Cashier Dashboard

Dashboard

Generate Bill

Sign Out

Welcome, Cashier

Start Generating Bill
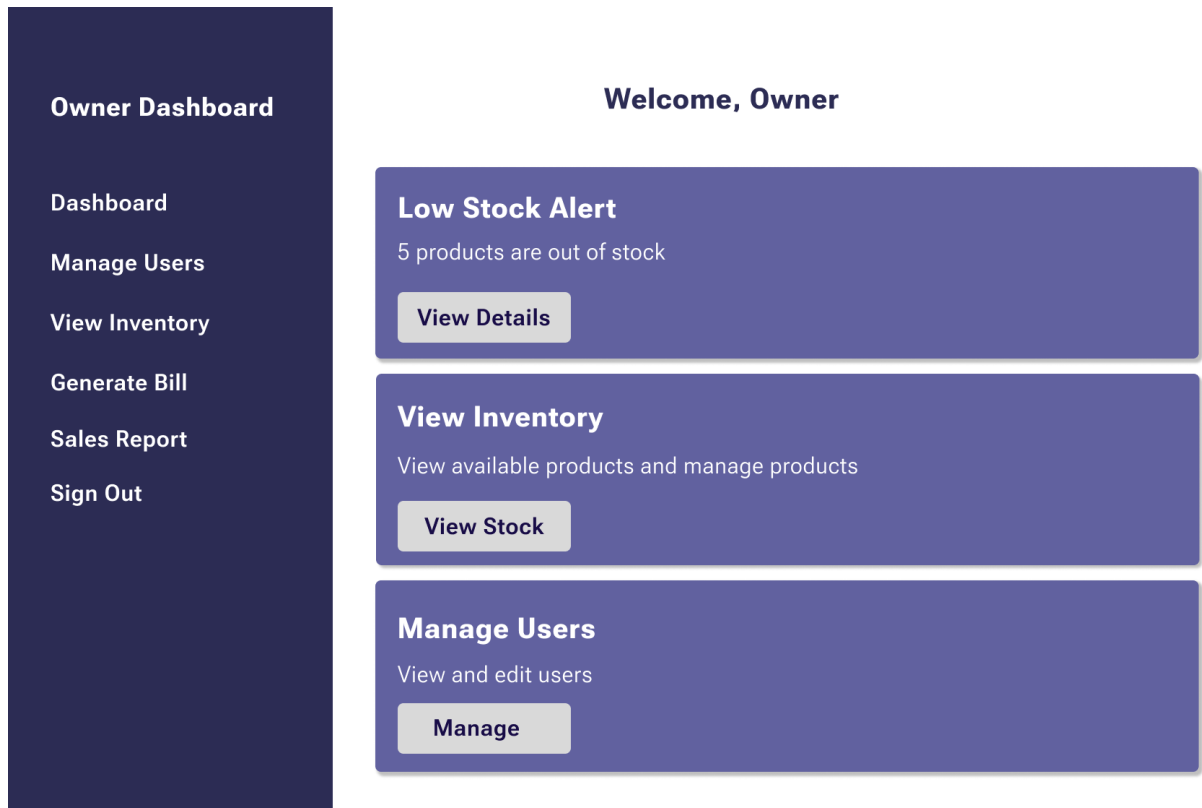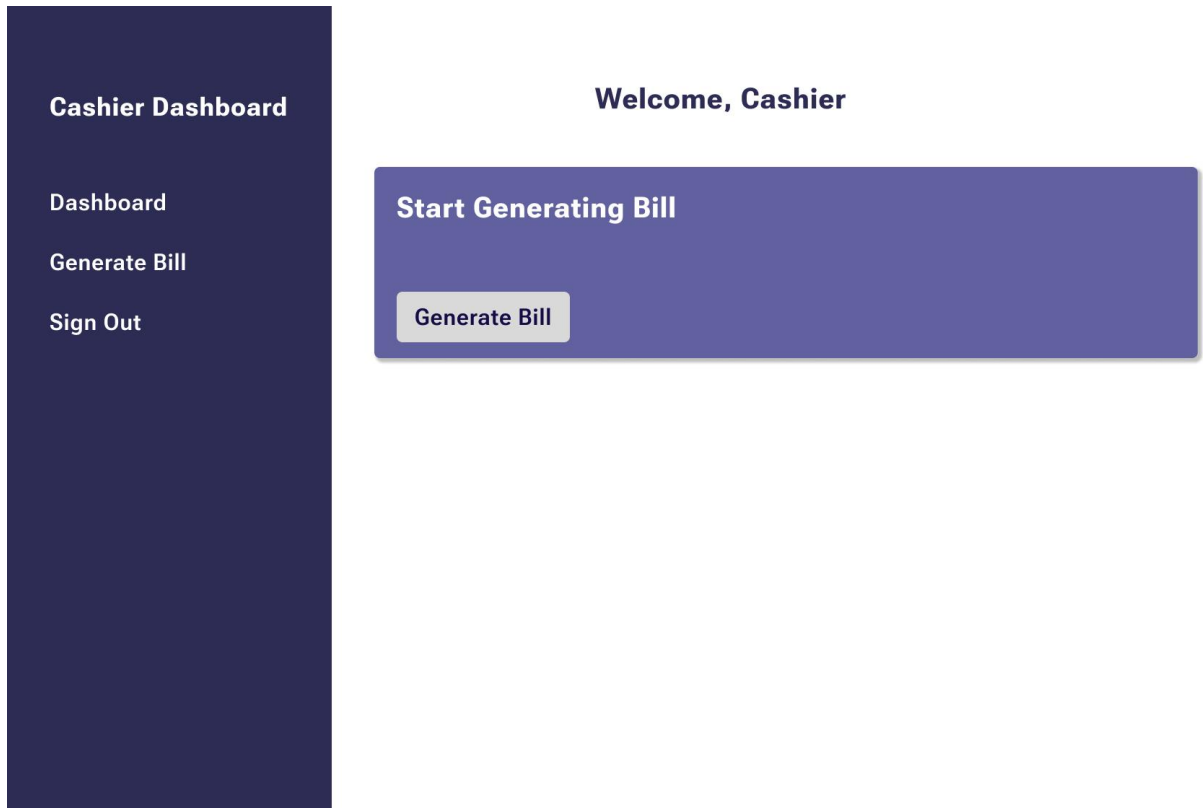
Generate Bill

6. **Billing Management Page**

Cashier Dashboard

Dashboard
Generate Bill
Sign out

Search for product

Product

Quantity

Add Product

Price

0.00/-

Generate Bill

7. **Stock Management Page**

View Products

Add New Product

Product Name

Category

Select Category

Quantity

Add Product

Category: Medicines

Paracetamol - 100 units

Ibuprofen - 50 units

Category: Supplements

Vitamin C - 200 units

Omega 3 - 120 units

Category: Equipment

Blood Pressure Monitor - 10 units

Thermometer - 30 units

8. **Low Stock Notifications Page**

Low Stock Alerts

PRODUCT NAME	CATEGORY	CURRENT STOCK	THRESHOLD
Paracetamol	Medicines	8	10
Vitamin C	Supplements	5	15
Thermometer	Equipment	3	5

Ensure to restock the low-stock items immediately!

User Management Page:

Manage Users

Add New User

Username

Password

Role

Select Role ▾

Add User

Existing Users

Username	Role	Actions
john_doe	Owner	Delete
jane_smith	Cashier	Delete
mike_jones	Stock Manager	Delete

Sales Report Page:

Sales Report

Date	Product Name	Quantity Sold	Total Revenue
2025-01-18	Paracetamol	50	250/-
2025-01-18	Vitamin C	30	150/-
2025-01-18	Thermometer	20	300/-

Export to PDF/CSV

Users:

1. **Owner**: The pharmacy owner who has access to all functionalities in the app. They are responsible for managing users, stock, sales reports, and settings.

2. **Cashier**: A user who handles billing.

3. **Stock Manager**: A user who is specifically responsible for managing stock (viewing and updating products, handling low stock alerts).

Use Cases:

Use Case ID	Use Case Name	Actor(s)	Description	Preconditions	Postconditions
UC-1	**User Login**	User	User logs in using a specific username and password.	User has a registered account.	User gains access to the system.
UC-2	**Password Reset**	User	User resets their password using a 6-digit code received via email.	User has a registered account and email access.	User can log in with the new password.
UC-3	**New User Registration**	New User	New user registers using a username, password, and 6-digit verification code.	User does not have an account.	New user account is created.
UC-4	**View Products in Stock**	Manager/Owner	User views available products in stock.	User is logged in.	User sees the list of available products.
UC-5	**Add New Product**	Manager/Owner	Manager/Owner	User has the appropriate access level.	New product is added to the inventory.

Use Case ID	Use Case Name	Actor(s)	Description	Preconditions	Postconditions
UC-6	**Low Stock Notification**	Manager/Owner	System notifies the user when a product falls below the stock threshold.	Stock level falls below the predefined threshold.	User is alerted about low stock.
UC-6	**Generate Bill**	Cashier/Owner	User selects products and quantities to generate a bill.	Products are available in stock.	Bill is generated with product details.
UC-7	**Generate Sales Reports**	Owner	Cashier generates weekly or monthly sales reports.	Sales data is available.	Report is generated and displayed.
UC-7	**Export Sales Report**	Owner	User exports sales reports and inventory lists in CSV or PDF format.	Sales reports are available.	Report is exported in the chosen format.
UC-11	**Add/Remove Users**	Owner	Owner adds, updates, or removes users from the system.	Owner is logged in.	Users are added, updated, or removed.
UC-12	**Access Role-Specific Dashboards**	All Actor	Different user types access different dashboards and functionalities.	User is logged in.	User accesses the appropriate dashboard.
UC-13	**Logout**	All Actor	User logs out of the application.	User is logged in.	User is logged out and redirected to the login screen.

Summary of Use Cases:

1. **Login**

2. **Sign Up**

3. **Forget password**

4. **Owner Dashboard**

5. **Cashier Dashboard**

6. **Manager's Dashboard**

7. **Billing Management** (Cashier)

8. **Stock Management** (Owner, Stock Manager)

9. **Low Stock Notifications** (Owner, Stock Manager)

10. **Sales Report Generation** (Owner)

11. **Exporting Reports/Inventory** (Owner)

12. **User Management** (Owner)

13. **Sign out**

Architecture Diagram:

Model- View-Controller:

It is responsible for managing the state of the application and handling data-related operations such as fetching, saving, and updating records.

It communicates directly with the database or data storage.

View:

The View is responsible for displaying the data to the user in a user-friendly way.

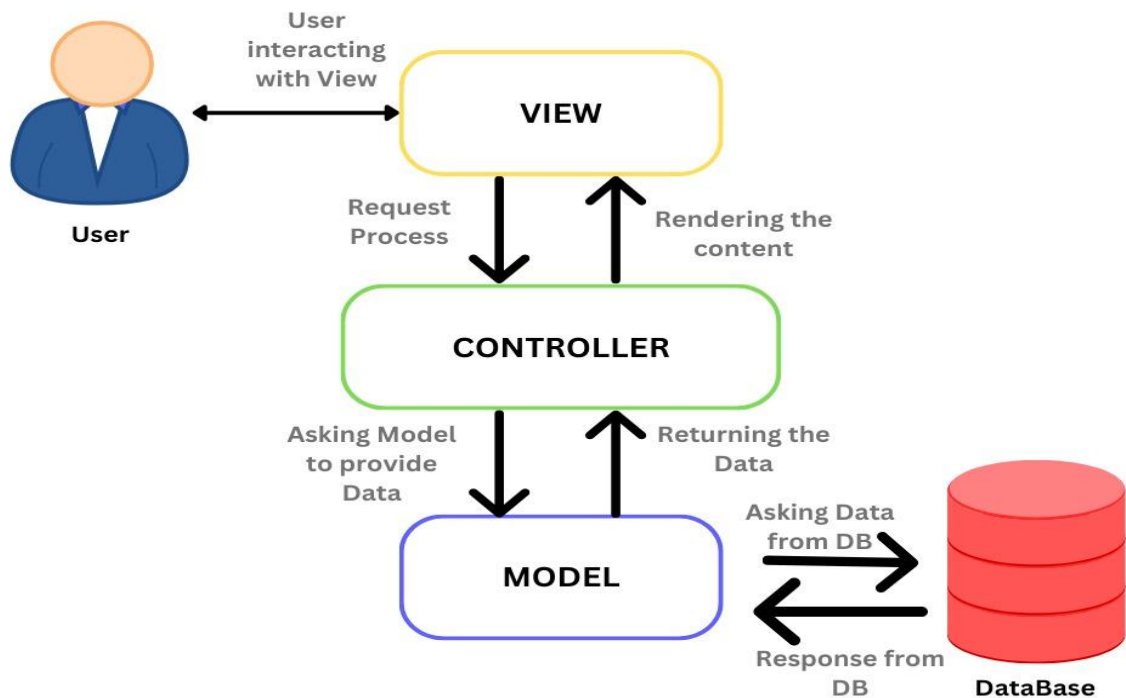It handles the user interface (UI) and presentation logic.

The View receives data from the Model and renders it to the user, without directly interacting with the Controller or performing business logic.

Controller:

The Controller acts as the intermediary between the Model and the View.

It processes user inputs (e.g., clicks, form submissions) and decides what to do with the data.

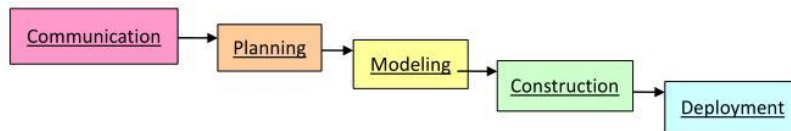The Controller fetches data from the Model and sends it to the View or updates the Model based on user actions.
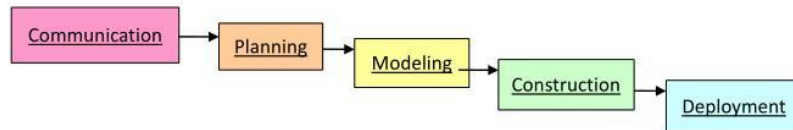
SDLC Model:

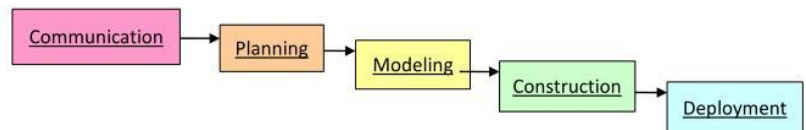**Incremental Model:**

Incremental Model (Diagram)

Increment #1

Communication → Planning → Modeling → Construction → Deployment

Increment #2

Communication → Planning → Modeling → Construction → Deployment

Increment #3

Communication → Planning → Modeling → Construction → Deployment

9

Gantt Chart:

Gantt Chart for Medical Billing App Development
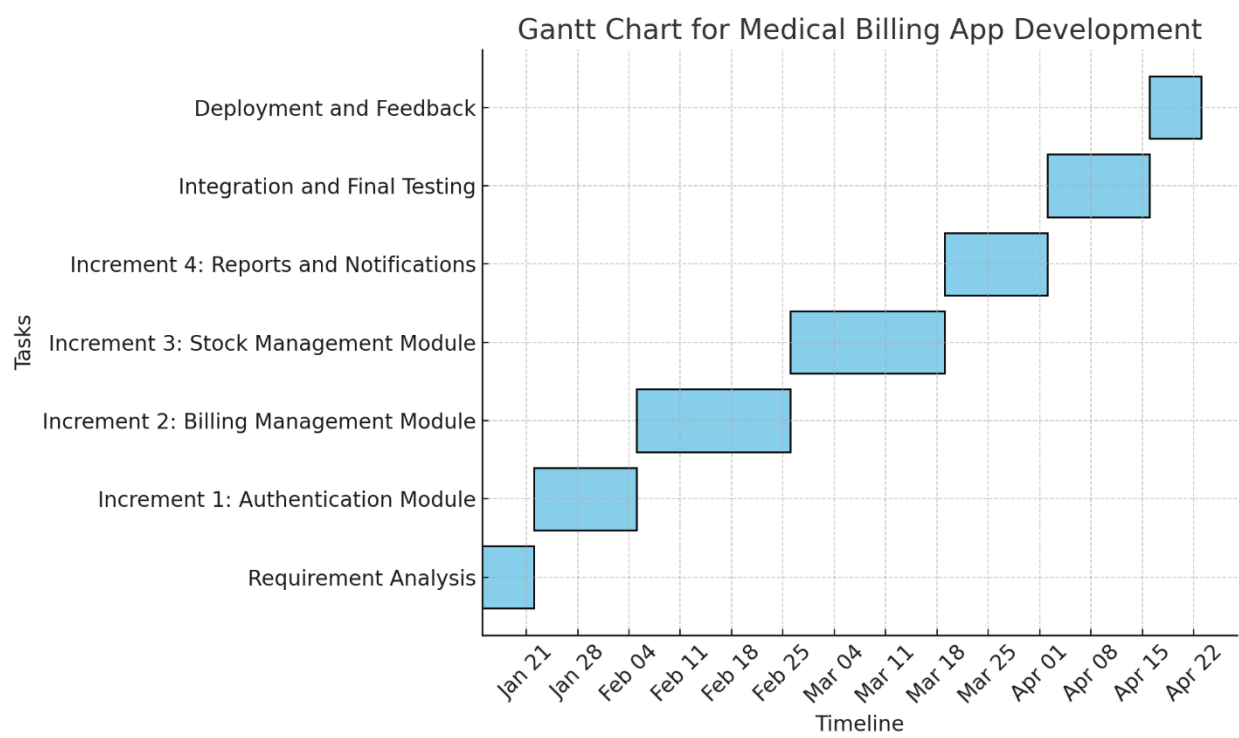
Unit Testing:

Unit testing in Flutter is a method to verify the correctness of a specific function, class, or method in isolation. It allows developers to ensure that individual components (units) of the Flutter app behave as expected, independent of other parts of the app.

Purpose:

- To test the logic and functionality of individual functions and methods.

- To ensure that the software components work correctly after updates or code changes.

- To catch bugs early in the development process.

Key Characteristics:

- **Isolated Testing**: Each test case checks a small unit of code, such as a function or method.

- **Automated**: Unit tests are often automated and can be executed whenever needed.

- **Quick Execution**: Unit tests are fast, as they don't involve the entire application but only small parts of it.

Tools for Unit Testing in Flutter:

- **Flutter Test**: Flutter's built-in testing library for writing unit tests.
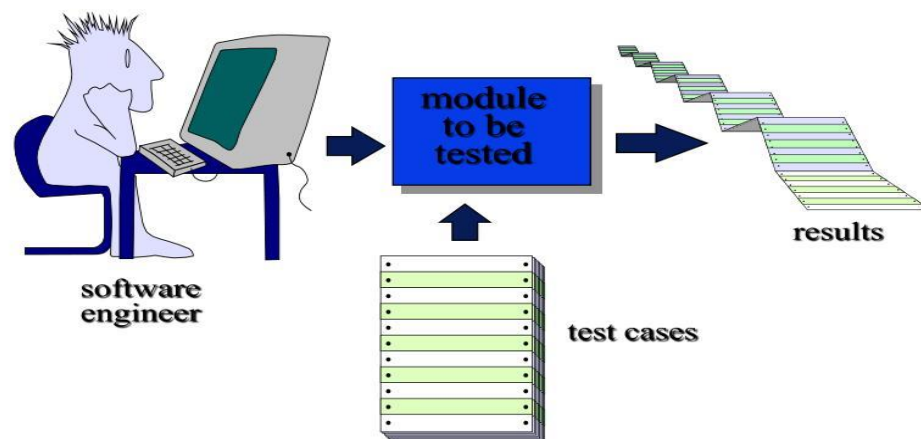
Development Tools & Technologies:

1. **Frontend**: Flutter (for desktop applications)

2. **Backend**: Express.js/Node.js

3. **Database**: SQLite or Firebase

4. **Deployment**: Local setup for desktops, scalable to AWS/Azure/Google Cloud if needed.

5. **Notifications**:

   o In-app: Flutter notifications (e.g.,awesome_notifications).