# Lab 3 Report: Frontend Application Development

Course: IoMT-Based Stress Monitoring System  
Student: Miltone  
Institution: Michigan Technological University  
Instructor: Dr. Ronghua Xu  
Duration: Week 3  
Lab Title: Building the Frontend Application Using Flutter  
Objective: To develop a functional Flutter application framework based on the UI design created in Figma, implementing navigation and interface components for the IoMT stress monitoring system.

## 1. Introduction

This lab marks the transition from conceptual design to tangible application development. Using the Figma blueprint established in Lab 2, the objective was to create the Flutter-based frontend framework for the IoMT stress monitoring application. The Flutter framework was selected for its cross-platform compatibility, rapid development cycle, and ability to deliver consistent user experiences across Android and web platforms.

## 2. Objectives

By the end of this lab, the following objectives were achieved:

* Develop the Flutter application structure based on the Figma UI design.
* Implement core screens including login, home, and data-related pages.
* Establish navigation among pages using Flutter’s Navigator and routing system.
* Ensure functional UI elements such as buttons, text fields, and navigation bars.
* Test and validate the frontend interface using Android emulator and web preview.

## 3. Application Structure and File Hierarchy

The application was developed as a multi-page Flutter project, maintaining a modular structure for ease of integration with backend services in later labs. The file hierarchy is as follows:

• main.dart – Application entry point handling initialization and routing.  
• login.dart – Manages user login and navigation to the home screen.  
• home.dart – Central dashboard displaying navigation options and summary widgets.  
• stress\_evaluate.dart – Interface for daily stress input via text field and buttons.  
• data\_search.dart – Page designed for fetching Fitbit or stress data.  
• stress\_reference.dart – Page showing educational content or health tips.  
• routes.dart – Contains named route definitions for easy page management.

## 4. Development Process

1. Stage 1: Project Setup

A new Flutter project was initialized using the command `flutter create stress\_monitor\_app`. Dependencies for material design and navigation were automatically included. The pubspec.yaml file was verified for core dependencies such as flutter, cupertino\_icons, and http (to be used in later labs).

1. Stage 2: Main Application Entry

The main.dart file served as the root of the application. It initialized a MaterialApp widget with a theme, title, and initial route set to the login screen. Navigation routes were registered to connect each page seamlessly.

1. Stage 3: UI Screen Development

Each Dart file represented a different functional page in the app. Stateful and stateless widgets were used as needed to build reusable, reactive components. Widgets such as Scaffold, AppBar, Column, and ElevatedButton were used to design responsive layouts matching the Figma design.

1. Stage 4: Navigation Implementation

Navigation between screens was implemented using Flutter’s Navigator.push and Navigator.pop methods. Additionally, a bottom navigation bar and drawer were introduced in the home screen to enhance user experience and simplify movement across pages.

1. Stage 5: Interface Testing

The app was compiled and tested on both the Android emulator and Chrome browser. Navigation flow, button responsiveness, and widget rendering were validated. Hot reload was used frequently to debug and refine layout designs in real time.

## 5. Application Screens and Features

|  |  |
| --- | --- |
| Screen | Key Features Implemented |
| Login Screen | Contains username and password fields, login button, and route to Home page. |
| Home Screen | Displays navigation buttons to Stress Evaluation, Data Fetch, and Reference screens. |
| Stress Evaluation Screen | Accepts daily self-reported stress scores with input validation. |
| Data Search Screen | Prepares UI elements to fetch Fitbit data through backend APIs (placeholder). |
| Reference Screen | Displays static information and visual aids related to stress management. |

## 6. Testing and Validation

Testing was performed using Flutter’s emulator and web preview modes. The focus was on interface usability, layout alignment, and navigation consistency. The following validation steps were executed:  
1. Launch app on Android emulator.  
2. Navigate from Login → Home → Stress Evaluation → Reference Page → Logout.  
3. Verify navigation buttons and app bar icons function correctly.  
4. Validate layout scalability for different screen sizes.  
5. Confirm that no exceptions occurred during page routing or rendering.

## 7. Challenges and Resolutions

|  |  |  |
| --- | --- | --- |
| Challenge | Description | Resolution |
| Widget Layout Issues | Inconsistent alignment across screens during testing. | Used Flutter’s Expanded and Padding widgets for uniform spacing. |
| Navigation Errors | Initial route conflicts caused runtime exceptions. | Defined named routes in MaterialApp and used pushReplacement for login transitions. |
| Hot Reload Failures | Frequent reloading lagged due to open background emulators. | Restarted debug sessions periodically and optimized emulator memory. |
| App Theme Inconsistency | Dark mode text visibility issue. | Implemented custom light theme with consistent contrast colors. |

## 8. Outcome and Deliverables

* Functional Flutter application framework built and tested successfully.
* Navigation between all pages implemented using Flutter’s route system.
* Consistent UI components across Android and Web platforms.
* Validated interface structure for backend integration in Lab 4.
* Demonstration video recorded showing full navigation workflow.

## 9. Reflection

This lab marked a significant milestone where design became interaction. The code written in Dart transformed static screens into a breathing digital interface. The experience demonstrated how software design merges logic with empathy — translating user needs into buttons, inputs, and motion. By maintaining clean architecture and reusability, the app is well-prepared for data connectivity in future labs.

“Every tap and transition in this app isn’t just a function — it’s a conversation between human intent and machine understanding.”

## 10. Forward Outlook

With the frontend now complete, the next step (Lab 4) focuses on setting up backend infrastructure in AWS. DynamoDB will handle data persistence, Lambda will process requests, and API Gateway will enable communication with the mobile client. The established Flutter screens will soon interact with real Fitbit and AWS data services, transforming the app into a true IoMT system.

✅ Lab 3 Successfully Completed. The frontend app is operational, visually consistent, and ready for backend service integration.