Blood Pressure Tracker System

Created by Patrick Sakamornlertsakul

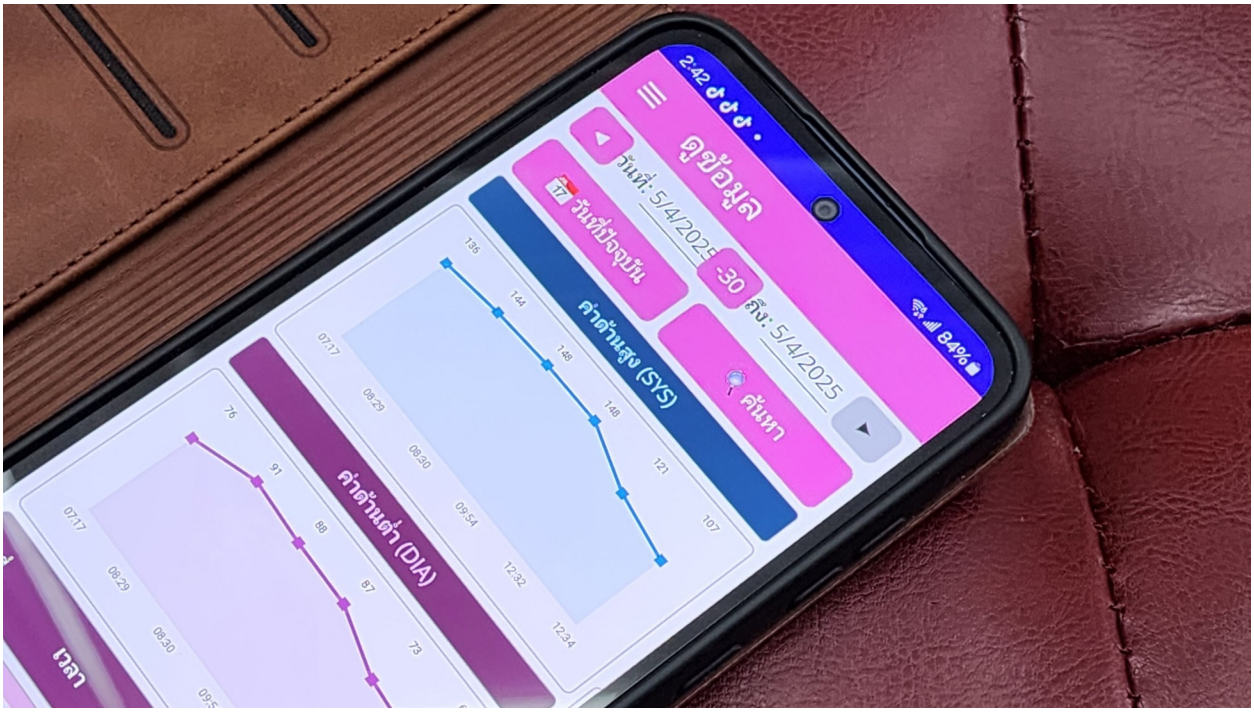
Introduction

This report outlines the creation and composition of a system for users to track their own blood pressure. The front-end is coded in C# and uses the .NET MAUI framework, which allows for the generation of a cross-platform application (one for mobile devices, in this case); the back-end is coded in Java and uses Spring Boot alongside a REST API.

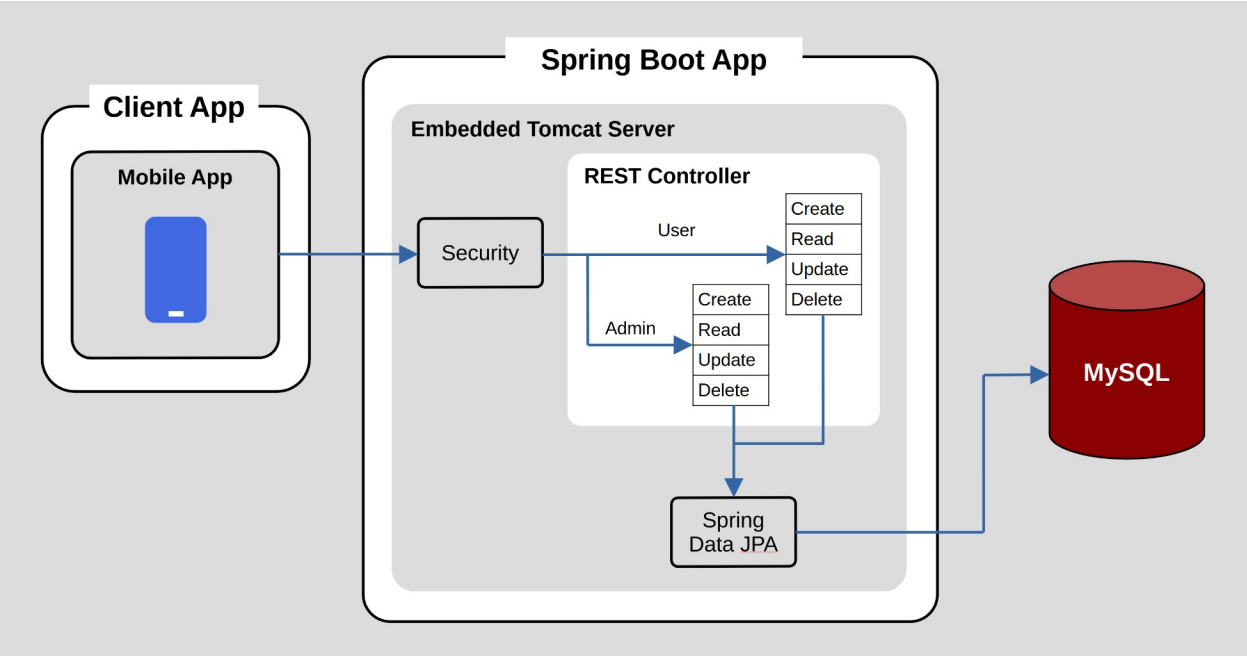
Objectives

The project was designed with multiple purposes in mind:

- To replace my grandmother's paper logging.
- To make it easier for my grandmother to share her blood pressure data among her caretakers, so that her prescribed medicine doses can be adjusted properly.
- To implement a working SMS alert system, for when new blood pressure data has values outside of a designated safe range.
- To have support for multiple languages, including English and Thai.
- To put my own learned skills in software and application development to use.



Project Architecture



The Blood Pressure Tracker System ships with:

- A cross-platform application developed with .NET MAUI, which supports iOS, Android, macOS (for Mac Catalyst and above), and Windows. In this case, **Android** was used as the main target platform.

The system includes the following backend components:

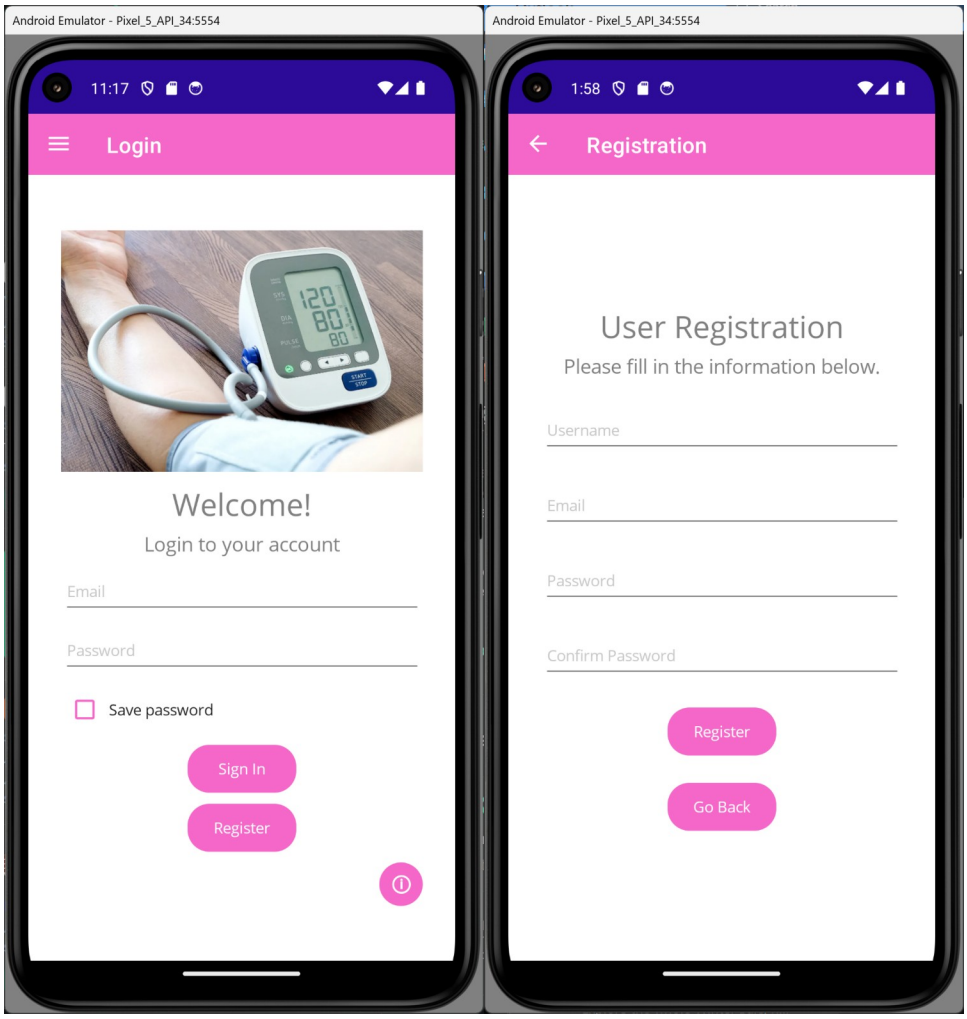
- An embedded Tomcat server that utilizes Spring Framework.
- A REST API that uses CRUD (Create, Read, Update, Delete) for modifying data in the User and Admin classes.
- The use of Spring Data JPA in order to ease the process of development.
- A MySQL database to store all of the data.
- Security for supporting multiple users.

Client Application

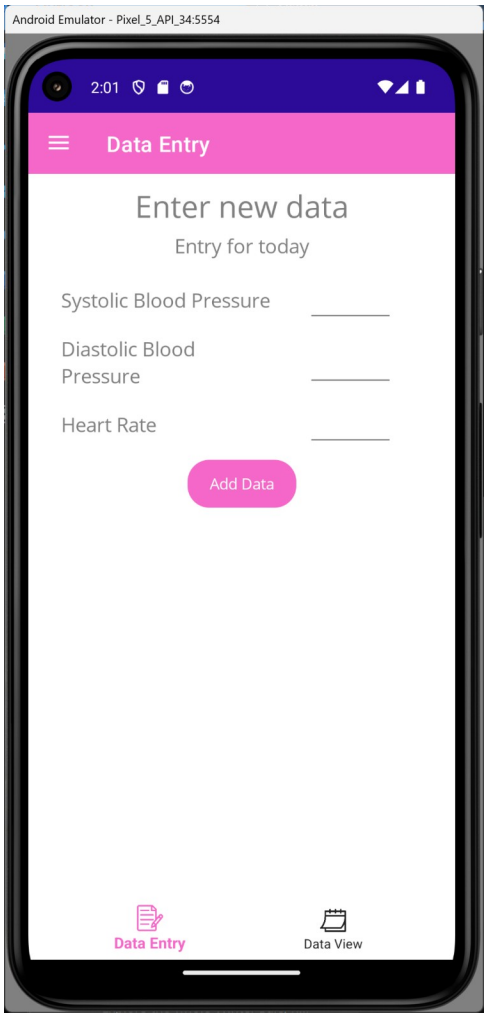
As stated before, the front-end for this project was coded in C# while utilizing the .NET MAUI framework. .NET MAUI was chosen as the target framework for a variety of reasons:

- It allows for the development of cross-platform applications that can run natively on Windows, macOS, iOS, and Android.
- It uses a single C# codebase and project system for all target devices.
- The provided integrated tooling in Visual Studio provides a seamless development process.
- The output performance is generally high.
- It is officially backed and developed by Microsoft themselves.

Mobile Application



The application starts at a simple **login page** that asks for a user's email and password. If you don't have an account, you can tap a button to go to a **registration page**, where you will be asked to set up one with new credentials. A checkbox on the login page enables saving your inputted password to make future logins faster.



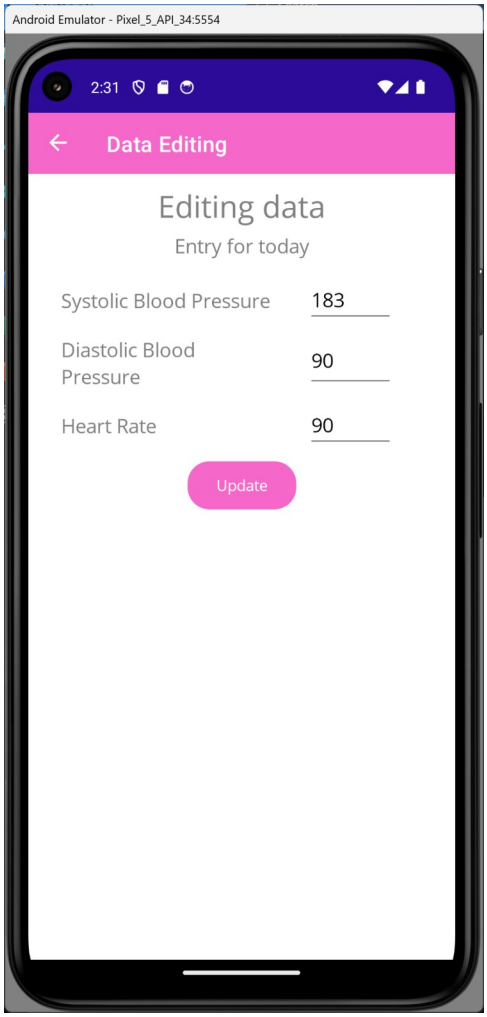
Upon successfully logging in, the user is brought to a screen with two tabs: **data entry** and **data viewing**. **Data entry** is highlighted by default.

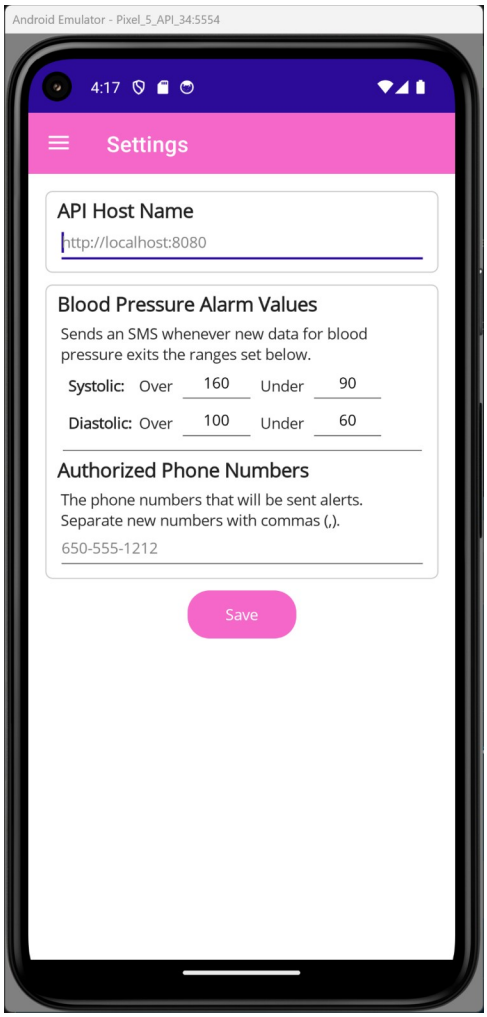
Data entry is a page for adding new blood pressure data, with numerical entry fields for **systolic blood pressure**, **diastolic blood presure**, and **heart rate**. The 'Add Data' button will send these values to the database as a new record, along with the current time and date.

The **data viewing** tab will show a list of previous blood pressure records sent to the database across a period of time, with two individual charts for systolic and diastolic blood pressure. By default, it will display the records that have been sent on the current day, with arrow buttons for going forward and backward a single day.

The two DatePickers at the top of the page allow the user to manually select the start and end dates to pull data from. By tapping the '**-30**' button, monthly mode will be enabled, which will widen the time period to the maximum of thirty days. Tapping it again will shift the selection another thirty days back. Hitting the '**Reset Date**' button will switch the page back to daily mode, along with resetting the time period to the current day.

Editing or deleting a specific record can be done by swiping it to the left, which will present you with either option. The ‘Edit’ button will take you to a new **data editing page**, separate from the one used for regular data entry, which can be seen below.

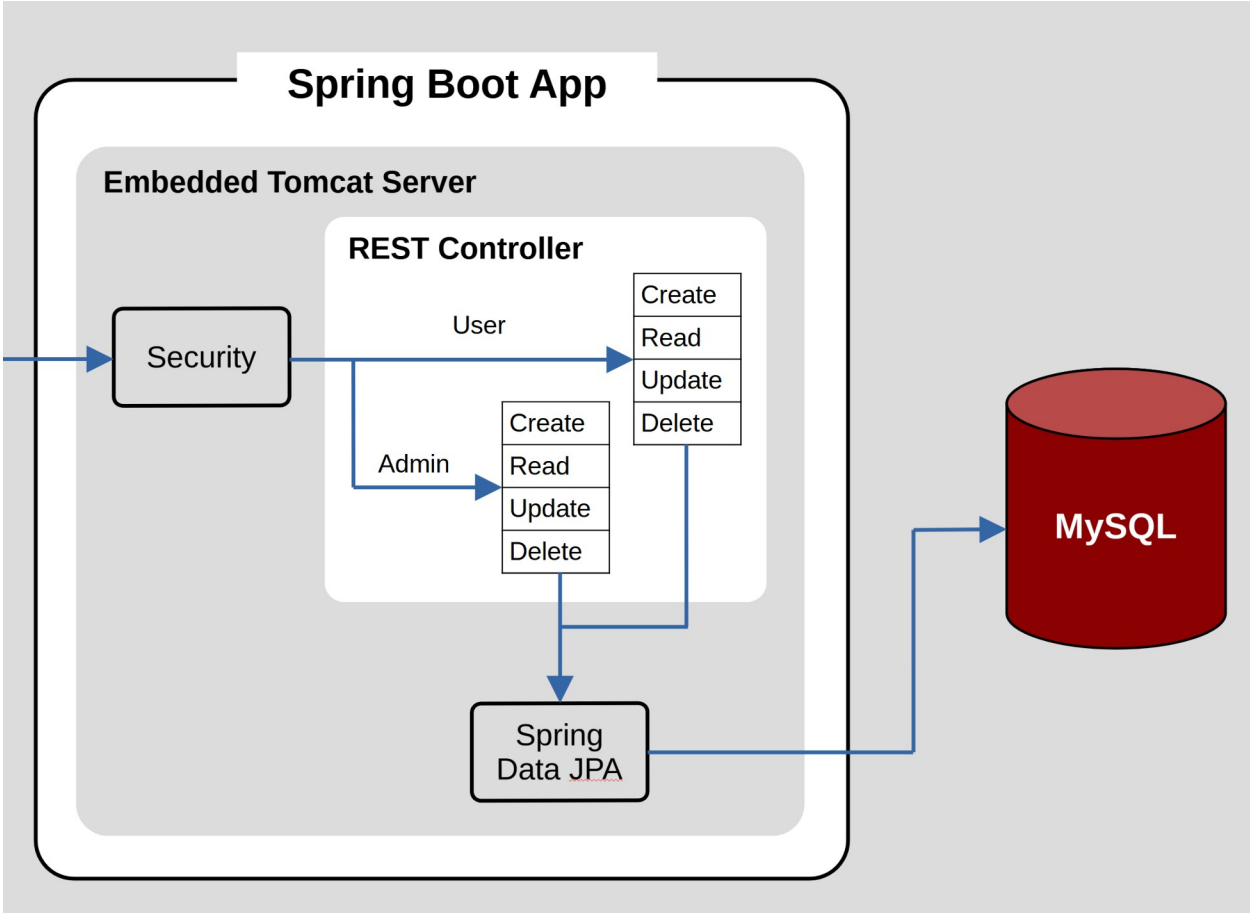




The **settings page** can be found in the shell menu accessible by swiping right from the left edge of the screen or by tapping the icon with the three horizontal lines. Here, you can change the following settings:

- The URL where the API is hosted, that the recorded data is sent to. Changing this will require you to close and restart the app for it to take effect.
- The accepted 'safety range' for systolic and diastolic blood pressure.
- Authorized phone numbers to send SMS messages to, in the case that blood pressure data exits the ranges set above.

Server

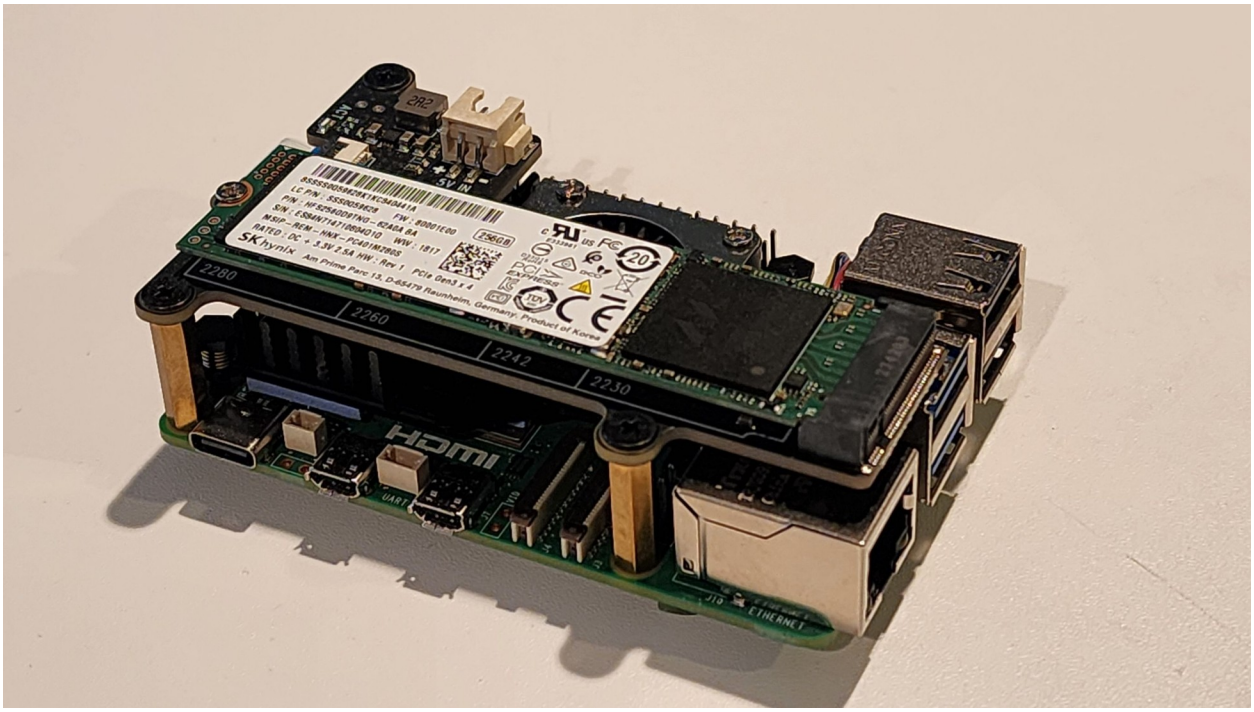


On the server side, Blood Pressure Tracker is developed with **Spring Boot** framework which rich in tools and ecosystem integration. The software architecture is base on **RESTful** Services to ensure a simple, stateless, and scalable communication between clients and servers. It is also flexible to interoperate with a wide range of client systems.

Spring Security is used to add authentication and authorization to the server, in order to ensure the resources within are protected from outside interference. Both Basic Authentication and JSON Web Tokens (JWT) are supported.

Spring Data JPA (implementation **Jakarta Persistence API**) is used to simplifies database interactions. With the ability to create repository

implementations automatically at runtime from a repository interface. It is also very easy to switch the primary database if needed—for example, **H2** was used during the process of development, and was replaced by **MySQL** when the time came for production deployment.



Since the code for the back-end is built off of the Java language, chosen such that the whole server can be easily hosted on varieties of devices and platforms. For example, the one we use on production is hosted and smoothly running on a Raspberry Pi 5 with Ubuntu.