

# Patientcare+ Coding Project Report



**A Document for Connecting Doctors and Patients  
through Code**

**Prepared by Martin Gawron, Nolan Reilly, Micah Olugbamila, and  
Maryann Olugbamila for use in CS 440  
at the  
University of Illinois Chicago**

**December 2024**

## Table of Contents

	List of Figures.....	4
	List of Tables .....	5
I	Project Description .....	6
1	Project Overview .....	6
2	Project Domain.....	6
3	Relationship to Other Documents .....	7
4	Naming Conventions and Definitions .....	7
4a	Definitions of Key Terms.....	7
4b	UML and Other Notation Used in This Document .....	8
4c	Data Dictionary for Any Included Models.....	8
II	Project Deliverables.....	9
5	First Release .....	9
6	Second Release .....	11
7	Comparison with Original Project Design Document.....	14
III	Testing .....	18
8	Items to be Tested.....	18
9	Test Specifications.....	20
10	Test Results .....	23
11	Regression Testing .....	25
IV	Inspection .....	25
12	Items to be Inspected .....	25
13	Inspection Procedures.....	28
14	Inspection Results.....	29
V	Recommendations and Conclusions.....	30
VI	Project Issues .....	30
15	Open Issues.....	30

16	Waiting Room .....	31
17	Ideas for Solutions .....	32
18	Project Retrospective .....	32
VII	Glossary .....	33
VIII	References / Bibliography .....	35
IX	Index .....	36

## List of Figures

Figure 1 - First Release Login Page.....	9
Figure 2 - First Scenario Patient Dashboard .....	10
Figure 3 - Second Release Add Doctors Page .....	11
Figure 4 - Second Release Doctor Dashboard .....	12
Figure 5 - Second Release Patient Health Page .....	13
Figure 6 - Second Release Urgent Care List.....	14
Figure 7 - Inspecting Analytics Page Alerts .....	25
Figure 8 - Login/Registration Toggle Inspection.....	26
Figure 9 - .NET Core Passwords Inspection.....	26
Figure 10 - Navbar Component Inspection.....	27
Figure 12 - Code Inspection Example 2 .....	28
Figure 11 - Code Inspection Example 1 .....	28

## **List of Tables**

Table 1 - Features for Medical Practitioners.....	31
Table 2 - Features for Patients .....	31
Table 3 - Project Retrospective Table.....	32

# **I Project Description**

## **1 Project Overview**

PatientCare+ is an Android app designed to monitor patient vitals, including heart rate, blood sugar, and blood pressure, using a chip with embedded sensors. The app calculates health risks, provides recommendations for improving health, and shares this information with doctors. By enabling remote monitoring and data sharing, it aims to reduce unnecessary hospital visits, particularly for managing chronic conditions like diabetes. Through Bluetooth communication with the chip and internet connectivity for database integration, the app ensures seamless functionality. It is designed to be user-friendly, secure, and accessible to patients and doctors, facilitating better health management and encouraging positive lifestyle changes with reminders and notifications.

PatientCare+ offers significant benefits for both patients and doctors by streamlining health monitoring and communication. For patients, it eliminates the need for frequent hospital visits by providing real-time tracking of vital and personalized health recommendations from the comfort of their homes. This saves time, reduces stress, and encourages consistent self-care with reminders and notifications. For doctors, the app ensures 24/7 access to patient data, allowing for proactive health management and early detection of potential issues. By leveraging technology to bridge the gap between patients and healthcare providers, PatientCare+ enhances efficiency, fosters better communication, and ultimately improves health outcomes for all.

## **2 Project Domain**

PatientCare+ is a health monitoring system based on the tracking of major health measurements such as heart rate, blood sugar levels and blood pressure. These vital signs are critical indicators of one's general health and are essential in the management of chronic conditions such as diabetes, high blood pressure and heart disease. Heart rate shows how healthy the heart is and can indicate stress, fitness levels or possible heart problems. The blood sugar level is important for controlling diabetes because abnormal readings can lead to serious problems like nerve damage or organ failure. On the other hand, blood pressure represents an important sign for hypertension, which significantly raises the risk of heart attack, stroke or renal diseases if not controlled. PatientCare+ keeps patients healthy by enabling real-time monitoring and securely sending this information to doctors through an app. This enables doctors to find health risks early and act quickly. Doctors can look at trends, give personalized advice and take steps to prevent health problems before they get worse. This reduces the need for many hospital visits and leads to better long-term health. This mix of technology with healthcare helps both patients and providers, promoting active health management based on data.

### 3 Relationship to Other Documents

This document is the report on the coding project that is based on the work of Group 2 for the Fall 2022 Semester in CS 440 at UIC. The individuals behind this project are Saif Alnuaimi, Sultan Alshkeili, Hazaa Alhosani and Sultan Aljneibi; we could not have completed this project without the work that they have done. More specifically, they wrote the development project report for the Patient Care + system, our entire coding project is based off of the work and ideas that they outlined in that development report document. Since the coding project incorporates many of their ideas, this coding project report will refer to much of their work directly in the development project report.

### 4 Naming Conventions and Definitions

#### 4a Definitions of Key Terms

**User:** An individual who interacts with the PatientCare+ app.

**Patient:** An individual who uses the PatientCare+ app to monitor their health vitals, receive health recommendations, and communicate with healthcare providers.

**Doctor:** A licensed healthcare professional who assesses patient data through PatientCare+ to monitor health statuses, provide prescriptions, and communicate with patients.

**Health Index:** A composite score generated by PatientCare+ that reflects a patient's overall health status based on various monitored vitals and health metrics.

**Glucose:** The level of blood sugar in a patient's blood, monitored by PatientCare+ to manage and track conditions such as diabetes

**Heart Rate:** The number of heart beats per minute, measured by PatientCare+ to assess a patient's cardiovascular health.

**Blood Sugar:** The concentration of glucose in a patient's bloodstream, essential for managing diabetes and overall metabolic health.

**Prescription:** Medications or treatment plans prescribed by doctors to patients, which can be managed and tracked within the PatientCare+ app.

**Messages:** Communications exchanged between doctors and patients through PatientCare+, facilitating discussions about health status, recommendations, and other relevant information.

**Appointments:** Scheduled meetings between patients and doctors for consultations, check-ups, or other healthcare services, managed and reminded through PatientCare+.

**Monitor:** The activity performed by doctors using PatientCare+ to oversee and analyze the health data and vitals of their patients.

**Urgent Care:** A status within PatientCare+ indicating that a patient's health score has reached a critical level, necessitating immediate attention and continuous monitoring.

**Database Integration:** The process of connecting PatientCare+ to a centralized database via internet connectivity, enabling secure storage and retrieval of patient data.

**Data Sharing:** The process of sharing patient health data from PatientCare+ to doctors, allowing them to make informed decisions.

**Dashboard:** Interface within the PatientCare+ app that provides users with access to different functionalities such as analytics, data input and messaging. There are two different versions of the dashboard based on the role of the user.

**Analytics:** Tools and visual representations within the Dashboard that process and display patient health data to assess health risks.

**Health Metrics:** Quantitative measures of specific health indicators tracked by the PatientCare+ app such as heart rate, blood sugar, blood pressure and the overall health index.

**Data input:** The ability in the Dashboard to allow users to enter and update health related information such as a patient logging their heart rate or a doctor prescribing a prescription.

**User Interface:** The visual layout and interactive elements of the Dashboard that allow users to interact with the PatientCare+ app.

#### **4b UML and Other Notation Used in This Document**

This document follows the “UML Distilled” practices as described by Fowler – Version 2.0 OMG UML standard.

#### **4c Data Dictionary for Any Included Models**

Not applicable for this document...



## II Project Deliverables

### 5 First Release

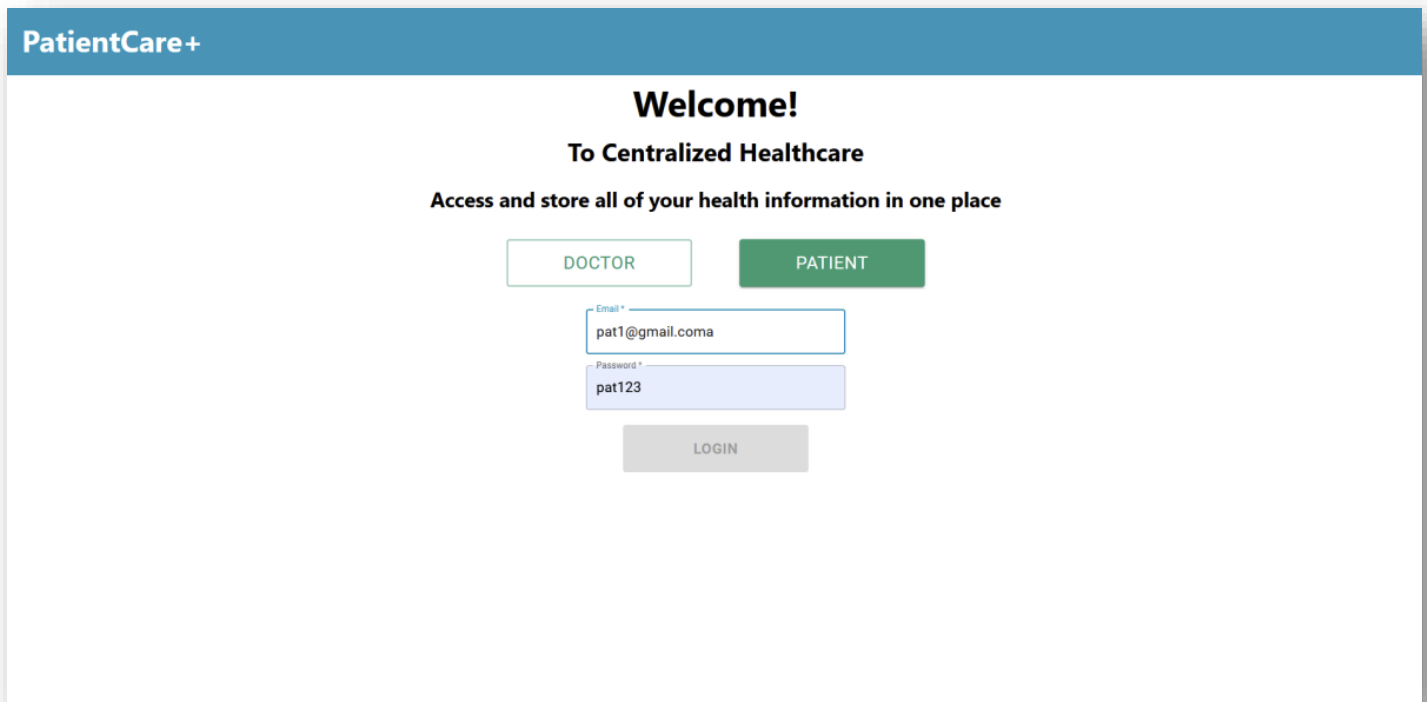
The first release of Patientcare+ focused on Patient and Doctor registration and login, providing patients with an interface to input their health data, which is then securely saved in the database.

#### **Scenario “Patient/Doctor Registration”**

Users can register by providing their personal details, including name, email, and password, through the registration interface. The registration process securely stores user credentials and data, utilizing encryption for sensitive information.

#### **Scenario “Patient/Doctor Login”**

Once users are registered, they can securely log in using their email and password through the login interface, which is secured by JWT authentication.

The image shows a web application interface for PatientCare+. At the top, there is a blue header bar with the text "PatientCare+" in white. Below the header, the main content area has a white background. It starts with a "Welcome!" message in bold black text, followed by "To Centralized Healthcare" in a smaller bold black font. Below this is a line of text: "Access and store all of your health information in one place". There are two buttons: a light green button labeled "DOCTOR" and a dark green button labeled "PATIENT". Below these buttons are two input fields. The first is labeled "Email \*" and contains the text "pat1@gmail.coma". The second is labeled "Password \*" and contains the text "pat123". Below the password field is a grey button labeled "LOGIN".

**Figure 1 - First Release Login Page**

### **Scenario "Patient Inputs Health Data"**

After a patient is successfully registered, the patient can now login to the application and access the patient dashboard of action items they can decide to use. The only available action item available at this time was the option to input their health data to be stored in the database. Upon submission the application will handle sending this information to be stored in the database under the patient's health table along with other personal data for future retrieval by patient or registered physician.

The dashboard is a web application interface for a patient. It consists of a sidebar on the left and a main content area. The sidebar contains a top section with a user profile icon, a dropdown menu for 'App Name' (currently showing 'Web app'), and a list of navigation links: Home, Analytics, Clients, Tasks, Settings, About, and Feedback. The main content area has a header with four input fields labeled 'Glucose Level', 'Heart Rate', 'Blood Sugar', and 'Blood Pressure', followed by an 'ENTER' button. Below this header is a large blue button labeled 'GET HEALTH DATA'. At the bottom of the sidebar, there is a user profile section showing a name and email address.

**Figure 2 - First Scenario Patient Dashboard**

## 6 Second Release

The second release improved patient-doctor interactions, empowering patients with tools to analyze their health and providing doctors with deeper insights into practice and patient needs.

### Scenario “Connecting Patients and Doctors”

Registered patients were given access to view all available doctors practicing with PatientCare+. A menu option on the patient dashboard was updated to allow patients to browse through the list of available doctors. Only doctors whose patients have added them as their registered physicians can access the patient's health data.

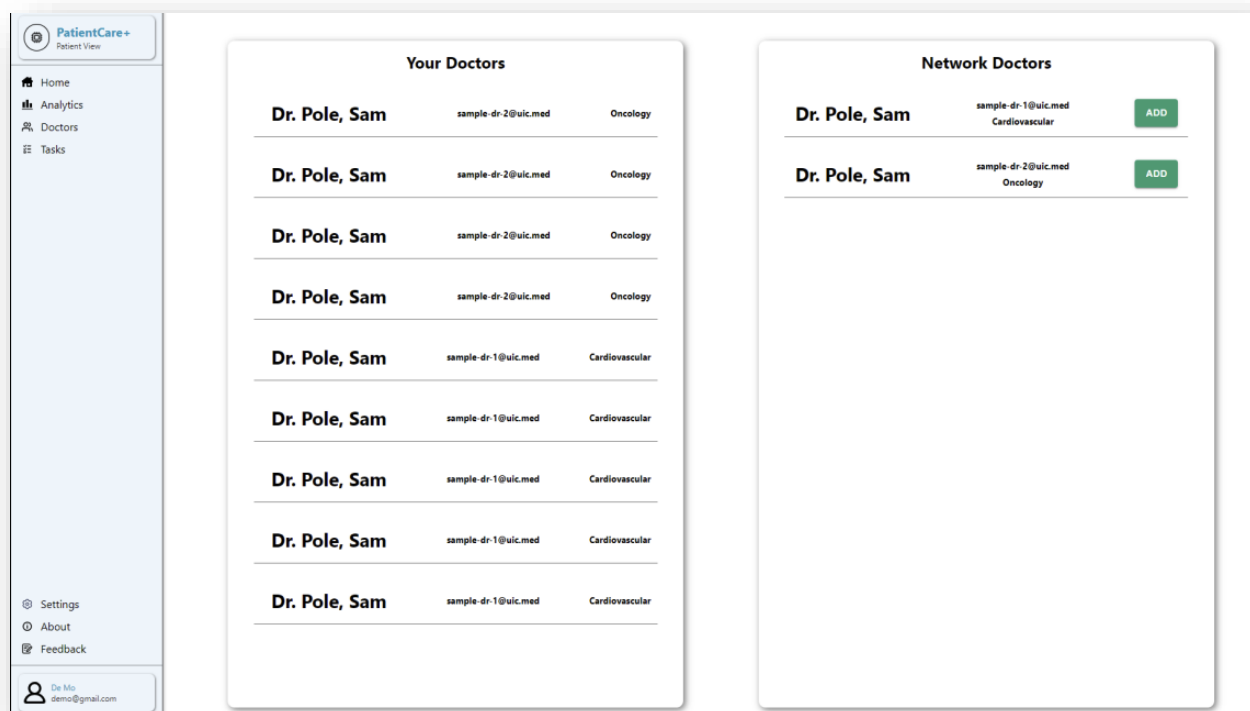
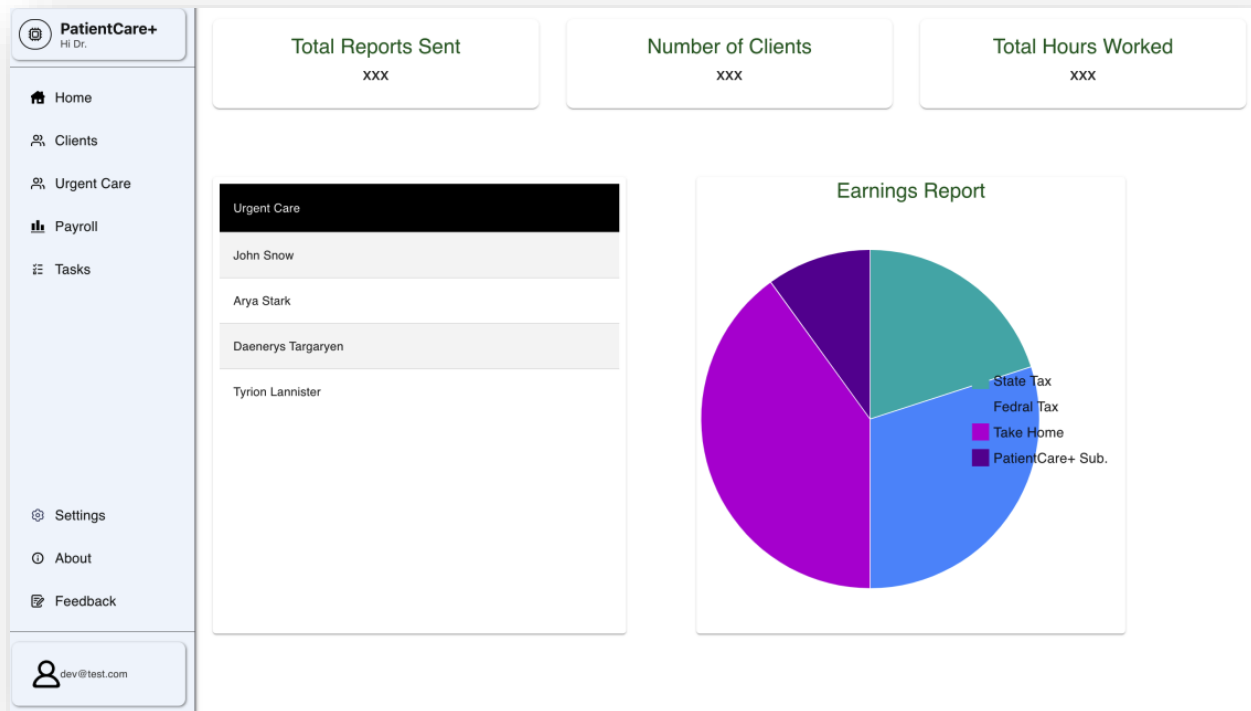


Figure 3 - Second Release Add Doctors Page

The doctors dashboard introduced a new quick overview of the total number of patients they are connected with. A menu button on the sidebar was developed to provide more detailed information about each patient, including access to their health data. This feature allowed doctors to provide personalized feedback, recommendations, and prescriptions based on a patient's health data, ultimately improving the quality of care and patient outcomes.



**Figure 4 - Second Release Doctor Dashboard**

### Scenario “Profile Modification”

Patients and doctors were not given the feature to modify certain flexible information provided during registration as there are ongoing legal and data concerns of feature implication. Rather, a view-only page was designed to display users’ information. Both patients and doctors can now view their account details as a **"profile card."**

### Scenario “Health Analysis View”

The first release of the application supported patients viewing graphical trends of their health data giving them a good overview of their health over time. This feature was improved upon in the second release for patients to have access to an AI that gives them advice on common practices to improve their health.

In addition to this, Doctors were given the option of sending a message to a patient regarding their health data, this message option will be available under their patient view. The Patient's dashboard was updated to include a message center where they can view their doctor’s advice and prescription data if any.

### Client Health Record

Select a client to view details

S/N	Glucose Level	Sugar Level	Heart Rate (SI)	Blood Pressure (SI)
1	100	200	80	120/80
2	120	220	90	130/90
3	110	210	85	125/85
4	105	205	82	122/82

Note

ADD

### Prescription

+ ADD RECORD

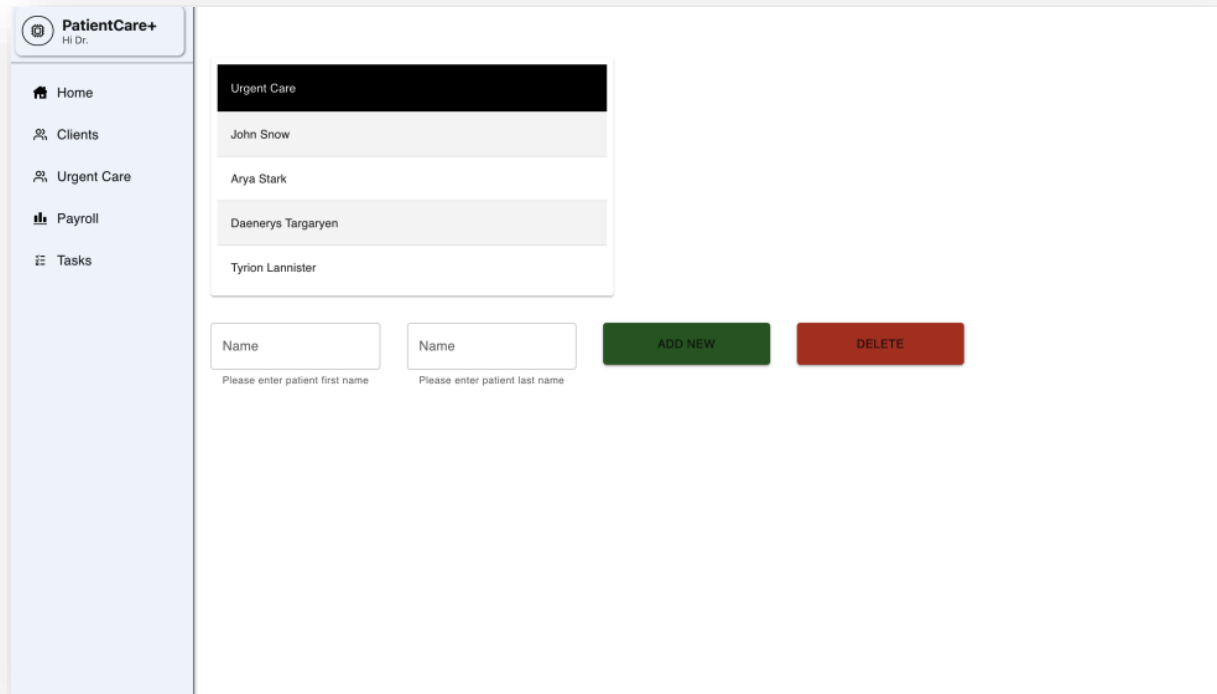
Medication	Dosage	Start Date	End Date	Remarks
Paracetamol	500mg	2/7/2024	6/13/2024	Take after meal

Rows per page: 100 1-1 of 1

Figure 5 - Second Release Patient Health Page

### Scenario: Doctor Insights

An important feature provided to doctors was the "**urgent care**" view, which prioritizes patients needing immediate attention. The current implementation allows doctors to add and remove patients from the urgent care list with the **add** and **remove** button.



**Figure 6 - Second Release Urgent Care List**

**\*\*Note:** The figures above illustrate features as developed at the time of the release. Features may have undergone further development, and the user interface (UI) may have changed since then.

## **7 Comparison with Original Project Design Document**

### Device Compatibility

The project outline provided to us specifically wanted this to be a mobile application [1, p. 9], most likely for the reason that mobile apps are more easily accessible, and most people are comfortable with using mobile devices every day. However, due to our experience with web app development we decided to create a React based web application. Although the development cycle for a mobile app would be very similar to our approach with React and web development, it was the tool and framework that we were more comfortable with and confident in creating an encompassing prototype to display the more important components and features of the backend which is made using a .NET Core framework (mainly C#).

**Summary:** The project was a mobile application, we made the prototype as a web app.

## **Data Input**

The original project outline wanted the data input to be done via sensors [1, p. 9] that would be able to automatically and periodically read data from the user/patient that would then be uploaded to the patient's account in the database. This data would then be viewable for both the patient and the patient's doctor. However, we did not have devices or sensors at our disposal that we would be able to leverage for this functionality. Therefore, we instead opted for the patient to manually input their health data so that they could record themselves using various medical devices that can be found and purchased on their own. This choice does affect the nature of being able to automatically collect data, but it does not impede the end goal of easily connecting a patient with their doctor to share personal health information and get immediate feedback from a health care professional for trying to lead a healthier lifestyle

**Summary:** The original report wanted to use sensors for automatic data collection, but our project decided to simplify this with manual data entry from the user.

## **Use Cases Implementation**

The project report included a Use Case Diagram of the desired use cases that users will be able to use with the application. These included [1, p. 19]:

- Patient
  - Receive ID Code
  - Create Password
  - View Sensor Reading
  - Receive Health Advice
  - Communicate with Doctor
- Doctor
  - Register
  - Login
  - Generate Patient Code
  - View Patient Data
  - Schedule Appointment Time
  - Communicate with Patient

We, however, implemented some of these, not all of them, and included some other cases we felt were appropriate as well. Our use cases included:

- Patient
  - Register
  - Login
  - Add health data

- View health data (various forms)
- Add doctor to doctor list
- View doctor messages
- View doctor prescribed medication
- View algorithmically generated health scores/statistics
- View AI generated health feedback
- Doctor
  - Register
  - Login
  - View patients (list of doctor's patients)
  - View patients' health history
  - Add prescription for patients
  - Send message/note to patient
  - Add patient to an urgent care list
  - Add income information (personal tracking)
  - Add task to to-do list (personal tracking)

Notice that we added many features that we thought were appropriate for an application of this type, but we did leave out some key features as we did not have time to implement them given the deadline. These important features were two-way communication between doctor and patient as well as scheduling appointments with a doctor. But we did add many features not originally included such as prescription data, AI feedback, health scores/statistics, etc.

### **Summary:**

Features we did not include:

- Scheduling appointments

Features implemented differently (Our Implementation vs. Proposed Implementation):

- Patient adds doctor vs. Doctor generating patient code
- One-way communication vs. Two-way communication

Extra features we added:

- Prescription Data
- Health statistics (Patient)
- AI feedback (Patient)
- Task list (Doctor)
- Income tracking (Doctor)
- Urgent care list (Doctor)



## **Absence of Classes**

When analyzing the class diagram in the project report given to us [1, p. 40], it is apparent that the original authors of this project had intended for a notification system to be used in the application. This idea makes sense as it was supposed to be a mobile application and does not work as well for a web app unless we were to implement an email based notification system. We decided not to implement this feature since we instead just made a dedicated page for the patient where they can see whether any of their doctors have sent a message to them. Although if we had more time we could always add a notification system, it would be via email if we kept the web app but we could make it as a normal phone notification if it were built as a mobile application.

It is also apparent that the authors of the report were thinking of organizing health information into a file-like structure based on the presence of a Medical Record File class in the class diagram. We did not take this approach and instead took the approach of a patient just having a collection of data where each piece of data needs all fields filled out and has a date attached for better understanding of change over time. We could simulate a pseudo record file system by representing each data point as a Medical Record File while not storing it that way in the database.

**Summary:** The original report included a notification system and a class structure for medical record files that we did not include in the prototype of their project.

## **UI Differences**

The proposed style of the user interface is something that we did not follow closely for our project [1, pp. 51-53]. Although we did use a sidebar for page navigation and we did use plenty of tables for organizing data for the users, our styling was not exactly what the authors had envisioned. We took a very minimalistic approach without using a navigation bar at the top of the page, as well as not having a breadcrumb navigation feature with crumbs (links to previous pages) on the current page. Also, many of the tables resembled the structure of a database. We moved away from this design choice for our prototype of this application by trying to make the visualization of data as user friendly as possible to those that are not as technologically savvy.

**Summary:** Remnants of the proposed UI design exist in our prototype of the project, but we decided to format data differently for the user.

## **Holistic Approach**

Reading through the entire project report it is clear that the authors had intended for this web application to be a way to automate one's health tracking (pg 61-62). It would come with many added features such as diabetes diagnosis, abnormality triggered notification system, etc. This application was originally envisioned as a smart health application that would take care of data collection and processing for the patient. We, however, have built a system that could use the automatically collected data and format it into a user-friendly way for patients. Further development and release of our prototype could replace the manual data entry into the automatic data collection system that the authors originally envisioned, with more persistent data collection then the application could be transformed into a larger health database for a user to recognize health trends and help diagnose diseases and disorders.

**Summary:** Our prototype attempts to emulate the features that the authors envisioned for the project without the massive data collection through periodic sensor readings originally proposed.

## **III Testing**

### **8 Items to be Tested**

#### **Backend:**

- Persistent Data
  - Making sure that data stays persistent on shut down and can be retrieved again with no loss on system startup
- Consistent Response
  - Testing to make sure that API responses occur upon request, no matter the response type (200, 300, 400, etc.)
- Accurate Responses
  - Testing to make sure that API calls return the expected data in the expected format upon request
- Security
  - Making sure that sensitive information is only accessible with an API Key which will take the format of a Bearer Token

#### **Landing Page (Frontend):**

- Proper Authentication
  - Testing to make sure that the Home, Login, and Registration pages are accessible to non-authenticated users (not logged in)
  - Make sure that a valid login or registration attempt gives the user proper authentication to visit either the Doctor or Patient Dashboard pages
- Errors and Warnings
  - Making sure that the alert system informs the user of why a login or registration attempt was invalid, as well as notifying the user when fields aren't properly filled

#### Patient Dashboard (Frontend):

- Proper API Calls
  - Testing to make sure that the pages that require information from the server make proper API calls and handle responses appropriately
  - Bad responses should not break the application and instead handled to inform the user of the reason
- User Friendly UI
  - Formatting of data presented to the user should be clear and readable with no ambiguity in functionality
- Patient Input
  - Make sure that patient input that will be stored in the database through an API call to the server is formatted correctly for a valid response and proper storage
  - Input should update the database in real time and be reflected either by automatic page refresh or manual page refresh

#### Doctor Dashboard (Frontend):

- Proper API Calls
  - Testing to make sure that the pages that require information from the server make proper API calls and handle responses appropriately
  - Bad responses should not break the application and instead handled to inform the user of the reason
- User Friendly UI
  - Formatting of data presented to the user should be clear and readable with no ambiguity in functionality
- Doctor Input
  - Make sure that patient input that will be stored in the database through an API call to the server is formatted correctly for a valid response and proper storage
  - Input should update the database in real time and be reflected either by automatic page refresh or manual page refresh

## 9 Test Specifications

### **TC001: User Registration**

**Description:** Verifies that a new user can be successfully registered with valid input.

**Items covered by this test:**

- \_userManager - Creation of a new user.
- Validation of input data (e.g., username, password).

**Requirements addressed by this test:**

- User registration functionality.

**Intercase Dependencies:** N/A

**Input Specification:** Valid user details (e.g., username, email, password).

**Output Specifications:** New user created and saved in the database.

**Pass/Fail Criteria:**

- **Pass:** User is successfully created and stored in the database with valid input.
- **Fail:** User creation fails, or invalid input is not handled gracefully (e.g., no error message or incorrect error handling).

### **TC002: Role Assignment**

**Description:** Validates that roles can be assigned to users successfully.

**Items covered by this test:**

- \_roleManager - Creation and assignment of roles to users.

**Requirements addressed by this test:**

- Role assignment functionality.

**Intercase Dependencies:** TC001

**Input Specification:** User and role details (e.g., valid username and role name).

**Output Specifications:** User successfully assigned the specified role.

**Pass/Fail Criteria:**

- **Pass:** User is successfully assigned the specified role.
- **Fail:** Role assignment fails, or invalid role details are not handled appropriately.

### **TC003: Configuration Access Test**

**Description:** Ensures the IConfiguration object retrieves app settings correctly.

**Items covered by this test:**

- \_configuration - Access and retrieval of configuration settings.

**Requirements addressed by this test:**

- Proper configuration file setup and access.

**Intercase Dependencies:** N/A

**Input Specification:** Request for a specific configuration key (e.g., JWT settings).

**Output Specifications:** Correct value retrieved for the requested configuration key.

**Pass/Fail Criteria:**

- **Pass:** Configuration values are correctly retrieved for valid keys.
- **Fail:** Retrieval of configuration values fails or incorrect values are returned.

### **TC004: Persistent Data**

**Description:** Need the data stored about the users (doctor and patients) to be persistent on system shutdown and startup, all data should stay the same.

**Items covered by this test:**

- Server Management
- Database Upkeep

**Requirements addressed by this test:**

- Persistent Data boundary case

**Intercase Dependencies:** N/A

**Input Specification:** N/A

**Output Specifications:** N/A

**Pass/Fail Criteria:**

- **Pass:** All data remains constant and persistent across server shutdown and startup
- **Fail:** Any sort of data deletion that occurs if server shuts down

### **TC005: Patient Side API Calls**

**Description:** Makes sure that API calls made from the patient dashboard are properly formatted GET and POST requests, which are handled in a way that does not affect user experience.

**Items covered by this test:**

- *All Pages in Patient Dashboard*
- API call functions:
  - `_fullPatientRegistration`
  - `_storePatientName`
  - `_submitHealthData`
  - `_getHealthData`
  - `_getAllDoctors`
  - `_getMyDoctors`
  - `_addDoctor`
  - `_removeDoctor`
  - `_getPatientProfileInfo`
  - `_getPatientHealthRecord`

**Requirements addressed by this test:**

- Secure server calls for updating and retrieving from the database
- Upholding user experience by handling boundary edge cases

**Intercase Dependencies:** TC004

**Input Specification:** Calls to the API functions

**Output Specifications:** Response payload from the server

**Pass/Fail Criteria:**

- **Pass:** If responses do not crash the site, most notoriously the bad responses from the server
- **Fail:** If a single response crashes the site and ends the user session abruptly

### **TC006: Doctor Side API Calls**

**Description:** Makes sure that API calls made from the patient dashboard are properly formatted GET and POST requests, which are handled in a way that does not affect user experience.

**Items covered by this test:**

- *All Pages in Doctor Dashboard*
- API call functions:
  - `_fullDoctorRegistration`
  - `_storeDoctorName`
  - `_getDoctorUrgentCareList`

- \_addPatientToUrgentCareList
- \_removePatientFromUrgentCare
- \_getDoctorProfileInfo
- \_getMyPatientsData
- \_addDoctorNote
- \_addPrescription
- \_getHomeStatsData

**Requirements addressed by this test:**

- Secure server calls for updating and retrieving from the database
- Upholding user experience by handling boundary edge cases

**Intercase Dependencies:** TC004

**Input Specification:** Calls to the API functions

**Output Specifications:** Response payload from the server

**Pass/Fail Criteria:**

- **Pass:** If responses do not crash the site, most notoriously the bad responses from the server
- **Fail:** If a single response crashes the site and ends the user session abruptly

## 10 Test Results

### TC001: User Registration

**Date(s) of Execution:** 2024-11-15

**Staff Conducting Tests:** Micah Olugbamila, Backend Engineer

**Expected Results:** New user is successfully registered and saved in the database. Error messages are displayed for invalid input.

**Actual Results:** User was created successfully with valid input. Error messages displayed correctly for invalid input.

**Test Status:** Pass

### TC002: Role Assignment

**Date(s) of Execution:** 2024-11-15

**Staff Conducting Tests:** Micah Olugbamila, Backend Engineer

**Expected Results:** Specified role is successfully assigned to the user. Invalid role inputs are rejected with appropriate error messages.

**Actual Results:** Role assignment worked correctly for valid input. Errors were shown for invalid roles.

**Test Status:** Pass

### **TC003: Role Assignment**

**Date(s) of Execution:** 2024-11-15

**Staff Conducting Tests:** Micah Olugbamila, Backend Engineer

**Expected Results:** Configuration values are retrieved correctly for all valid keys. Invalid keys result in a proper error or null response.

**Actual Results:** Configuration values were fetched correctly for valid keys. Null values returned for invalid keys as expected.

**Test Status:** Pass

### **TC004: Persistent Data**

**Date(s) of Execution:** 2024-11-22

**Staff Conducting Tests:** Martin Gawron, Frontend Engineer

**Expected Results:** Data remains persistent after server shutdown and can retrieve previously stored data on server startup

**Actual Results:** Data in the database did not change (no additions, deletions, or updates) when server was forcefully shutdown and started back up

**Test Status:** Pass

### **TC005: Patient Side API Calls**

**Date(s) of Execution:** 2024-11-22

**Staff Conducting Tests:** Martin Gawron, Frontend Engineer

**Expected Results:** Navigating to pages goes smoothly and entering incorrect data did not cause the website to crash.



**Actual Results:** Navigating between web pages did not interrupt user flow, and when entering incorrect data was notified via a notification bubble that input was incorrect.

**Test Status:** Pass

#### **TC006: Doctor Side API Calls**

**Date(s) of Execution:** 2024-11-22

**Staff Conducting Tests:** Martin Gawron, Frontend Engineer

**Expected Results:** Navigating to pages goes smoothly and entering incorrect data did not cause the website to crash.

**Actual Results:** Navigating between web pages did not interrupt user flow, and when entering incorrect data was notified via a notification bubble that input was incorrect.

**Test Status:** Pass

### **11 Regression Testing**

Not applicable for this project at this stage...

## **IV Inspection**

### **12 Items to be Inspected**

Analytics Page Alerts

```
261
262 // might find a way to use alert on this page, but dont have to
263 v const showAlert = (type, message) => {
264   |   showAlert({ open: true, type: type, message: message});
265   |   setTimeout(() => showAlert((prev) => ({ ...prev, open: false})), 3000);
266   | }
267
```

**Figure 7 - Inspecting Analytics Page Alerts**

## Login and Registration Toggle Button

```
<ToggleButtonGroup class="login-btn-group" color="primary" value={alignment} exclusive onChange={handleChange} aria-label="platform">
  <Box className="flex">
    <ToggleButton className="login-toggle-choice" value="doctor" onClick={handleDoctorClick}>
      Doctor
    </ToggleButton>

    <ToggleButton className="login-toggle-choice" value="patient" onClick={handlePatientClick}>
      Patient
    </ToggleButton>
  </Box>
</ToggleButtonGroup>
```

Figure 8 - Login/Registration Toggle Inspection

## .NET Core Passwords

```
29
30 // Add Identity Services
31 builder.Services.AddIdentity<IdentityUser, IdentityRole>(options =>
32 { // todo: change password requirement to secured
33   options.Password.RequiredLength = 6;
34   options.Password.RequireNonAlphanumeric = false;
35   options.Password.RequireDigit = false;
36   options.Password.RequireUppercase = false;
37   options.Password.RequireLowercase = false;
38 })
39 .AddEntityFrameworkStores<PatientCarePlusDbContext>()
40 .AddDefaultTokenProviders();
41
```

Figure 9 - .NET Core Passwords Inspection

## Navbar React Component

```
22
23   return (
24     <div>
25       <nav>
26         <Box className="flex">
27           /* Nav-Logo */
28           <Link to="/" className="flex nav-link">
29             <p className="nav-logo-text">PatientCare</p>
30             <img src={Logo} className="nav-logo-img" alt="Logo"/>
31           </Link>
32
33           /* Buttons container */
34           <Stack spacing={2} direction="row">
35             <Button>About Us</Button>
36             <Button onClick={handleLogin}>
37               Log in
38               <LoginIcon />
39             </Button>
40
41
42             <Button onClick={handleRegister}>
43               Register
44               <AppRegistrationIcon />
45             </Button>
46           </Stack>
47         </Box>
48       </nav>
49
50       <Divider />
51     </div>
52   );
53
```

Figure 10 - Navbar Component Inspection

## 13 Inspection Procedures

Used Discord (electronically) for a quick and easy form of communication, as it provides a way to share screenshots of code blocks and discuss necessary changes. The platform's reactions feature, and group chat capabilities allowed our group to have an effective collaborative environment for discussions.

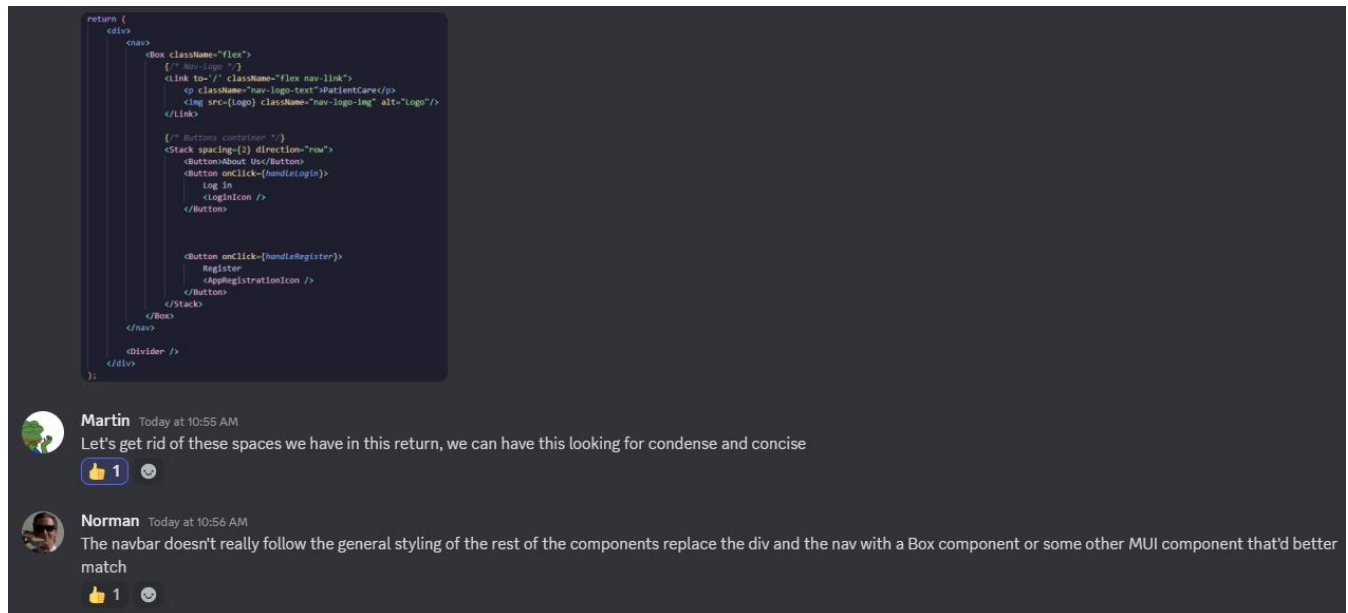


Figure 12 - Code Inspection Example 1

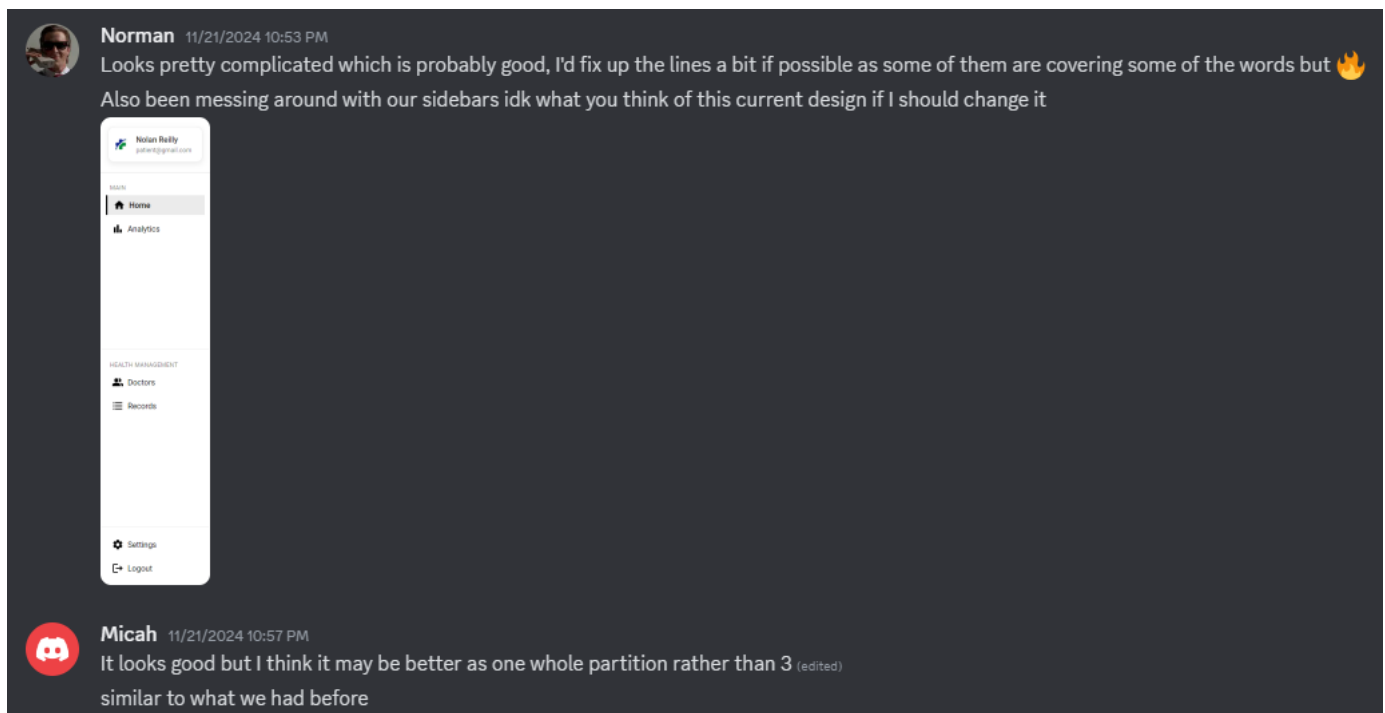


Figure 11 - Code Inspection Example 2

Additional communication methods included pre-demo meetings to review code or collaborative sessions for merging, where we collectively reviewed specific sections to determine what should be kept, removed or modified. However, no formal documentation was created for this process.

## **14 Inspection Results**

### **Analytics Page Alerts:**

Inspections were conducted on 11/28/24 during one of our final merges to the master branch before the live demo. We reviewed the functionality of each alert on every page and considered adding an alert to the Analytics page. However, it was decided that including an alert here would be redundant, as the existing icons already conveyed health data information effectively. Inspections were carried out by Nolan, Micah and Maryann.

### **Navbar Toggle Button:**

Inspections for the Navbar ToggleButton occurred on the same commit mentioned above on 11/28/24. During this review a developer noted that the attribute naming in the React components for the login and registration pages didn't align with React's standard naming conventions. The team reviewed this issue but no clear explanation was found and due to the component functioning correctly with the naming of class instead of className no changes were made. Inspections were performed by Martin, Micah and Maryann.

### **.NET Core Passwords:**

Inspections for the .NET Core Passwords occurred earlier in the semester, though the exact date is unknown as documentation wasn't maintained at the time. Before the first project demo, an overview of the backend code was done to determine how passwords should be implemented. The team decided to keep passwords minimal during development as further changes would have unnecessarily complicated the process. Inspections were conducted by Nolan, Martin and Maryann.

### **Navbar React Component:**

The Navbar React Component was inspected on 11/22/24 during a merge to the dashboard branch. All code was reviewed to be within standards and performance before committing to the dashboard branch. While the navbar's format didn't affect performance or user interface appearance, it was noted that it didn't fully match to the styling guidelines. No changes were made. Inspections were completed by Micah, Nolan and Martin.

## V Recommendations and Conclusions

All items tested passed the required testing and inspection process, ensuring that PatientCare+ version 1.0 is ready for a successful release. The following actions are recommended:

- **Monitoring:** The current version will be monitored in real-world usage to identify any issues that may arise post-deployment.
- **Client Communication:** Developers will maintain close communication with the client to address any concerns or feature requests.
- **CI/CD Implementation:** PatientCare+ version 1.0 will be maintained in a Continuous Integration/Continuous Deployment (CI/CD) state to adhere to industry standards, enabling rapid updates and improvements.

These steps will ensure the stability, scalability, and reliability of the product moving forward.

## VI Project Issues

### 15 Open Issues

**Urgent Care Table:** The Urgent Care Table will be more efficient if the PatientCare+ AI model is trained to assist doctors in updating the urgent care list, identifying critical patients, and allowing adjustments as needed. This ensures timely and appropriate medical attention for those requiring urgent care.

**Task Functionality:** Currently the Task functionality doesn't contain any database integration and only uses session storage. Adding logic to this would allow this feature to be used in production.

**CRUD Capabilities of Prescriptions:** The prescriptions component currently has the ability to add and edit prescriptions but needs to have the ability to remove and delete as well to be fully complete.

**Alert Styling:** The alert currently works as intended but needs to be fixed in terms of its location as it's not centered on the Dashboard section pages.

**Weather Integration:** A feature that was added in was a weather widget but was never implemented. Though not high on the list of issues, it's something to consider adding functionality onto to determine its place within the app.

**Creation of Monitoring Tool:** While we have a mockup showcasing how the app would enable communication between Patients and Doctors, the device intended to collect health data has not yet been implemented.

## 16 Waiting Room

The following ideas and features are not included in the current release of PatientCare+ version 1.0 but are noted for potential inclusion in future updates:

**Table 1 - Features for Medical Practitioners**

<b>Feature</b>	<b>Priority</b>
Ability to view patients' tests and lab results and provide recommendations.	MEDIUM
Option to adjust availability on a calendar for patients to book appointments.	MEDIUM
Live chat functionality to communicate with patients currently online who need immediate care or advice.	LOW

**Table 2 - Features for Patients**

<b>Feature</b>	<b>Priority</b>
Notifications for health tips related to individual health data.	MEDIUM
Reminders to take medications based on prescriptions.	MEDIUM
Ability to create tickets for new discomforts for doctor analysis and prescription.	LOW
Appointment scheduling for in-person checkups with medical practitioners.	HIGH
Dynamic medical information forms that adjust based on selected health parameters.	MEDIUM
Ability to add spouse, parent, or children to an account for easy access to family health data.	LOW

These features will be reviewed for feasibility and prioritization in subsequent releases, aligning with user feedback and system capability enhancements.

## 17 Ideas for Solutions

- **Extending the application to mobile/web framework**
  - Making the app more accessible by allowing users to switch between mobile and web platforms while keeping the user interface consistent across devices.
- **Integrating Sensor monitors for data collection**
  - Adding features to automatically collect health data through wearable devices or sensors, reducing manual input and improving accuracy. (i.e., Apple watches, Blood pressure monitors, etc.)
- **Enabling offline functionality**
  - Adding features to allow users to access and input data even without internet connectivity, syncing changes once a connection is re-established.
- **Developing multilingual support**
  - Adding multiple language options so the app can be used by people from different regions and backgrounds.
- **Enhancing Integration with External Healthcare Systems**
  - Connecting PatientCare+ and hospital networks or electronic health records (EHRs) to provide a more seamless healthcare experience.
- **Introducing achievement and rewards for improved user engagement**
  - Adding features like health challenges, rewards, or badges to motivate users to keep up with their health tracking.

## 18 Project Retrospective

**Table 3 - Project Retrospective Table**

Things We Should Continue	Things We Should Improve On
Team peer programming was introduced after release 2 and there were significant improvements in the quality of code produced and faster handling of bugs.	There were occasions when team members were not fully prepared for scrum meetings, which led to longer meetings, we should all be coming to meetings prepared.
It was a great experience working alongside great developers who challenge each other to be critical.	Team members should provide quick and honest feedback to one another's updates on discord channel.
A fantastic delivery on demo 2, client was pleased with the new release.	More details on Jira tickets will help developers understand current task.



## VII Glossary

**User:** An individual who interacts with the PatientCare+ app.

**Patient:** An individual who uses the PatientCare+ app to monitor their health vitals, receive health recommendations, and communicate with healthcare providers.

**Doctor:** A licensed healthcare professional who assesses patient data through PatientCare+ to monitor health statuses, provide prescriptions, and communicate with patients.

**Health Index:** A composite score generated by PatientCare+ that reflects a patient's overall health status based on various monitored vitals and health metrics.

**Glucose:** The level of blood sugar in a patient's blood, monitored by PatientCare+ to manage and track conditions such as diabetes

**Heart Rate:** The number of heart beats per minute, measured by PatientCare+ to assess a patient's cardiovascular health.

**Blood Sugar:** The concentration of glucose in a patient's bloodstream, essential for managing diabetes and overall metabolic health.

**Prescription:** Medications or treatment plans prescribed by doctors to patients, which can be managed and tracked within the PatientCare+ app.

**Messages:** Communications exchanged between doctors and patients through PatientCare+, facilitating discussions about health status, recommendations, and other relevant information.

**Appointments:** Scheduled meetings between patients and doctors for consultations, check-ups, or other healthcare services, managed and reminded through PatientCare+.

**Monitor:** The activity performed by doctors using PatientCare+ to oversee and analyze the health data and vitals of their patients.

**Urgent Care:** A status within PatientCare+ indicating that a patient's health score has reached a critical level, necessitating immediate attention and continuous monitoring.

**Database Integration:** The process of connecting PatientCare+ to a centralized database via internet connectivity, enabling secure storage and retrieval of patient data.

**Data Sharing:** The process of sharing patient health data from PatientCare+ to doctors, allowing them to make informed decisions.

**Dashboard:** Interface within the PatientCare+ app that provides users with access to different functionalities such as analytics, data input and messaging. There are two different versions of the dashboard based on the role of the user.

**Analytics:** Tools and visual representations within the Dashboard that process and display patient health data to assess health risks.

**Health Metrics:** Quantitative measures of specific health indicators tracked by the PatientCare+ app such as heart rate, blood sugar, blood pressure and the overall health index.

**Data input:** The ability in the Dashboard to allow users to enter and update health related information such as a patient logging their heart rate or a doctor prescribing a prescription.

**User Interface:** The visual layout and interactive elements of the Dashboard that allow users to interact with the PatientCare+ app.

## **VIII References / Bibliography**

- [1] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.
- [2] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.
- [3] Robertson and Robertson, Mastering the Requirements Process.
- [4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.
- [5] S. Alnuaimi, S. Alshkeili, H. Alhosani and S. Aljneibi, "Software Engineering Project Report: Patientcare+ Mobile Application," Chicago, 2022.

## **IX Index**

**No index entries found.**