Bilkent University
Department of Computer Engineering

Senior Design Project

T2312

MediXAI

MediXAI

Final Report

Project Team Member Names:

Ahmet Arda Ceylan	(22003148)
Mahmut Furkan Gön	(21902582)
Mustafa Yetgin	(21903191)
Omar Hamdache	(22001398)
Ömer Kağan Danacı	(21902584)

Advisor: Selim Aksoy

Course Instructors: Atakan Erdem, Mert Bıçakçı

13/05/2024

Contents

1 Introduction

1.1 Purpose of the System

We have identified challenges in the current healthcare system that lead to difficulties for individuals and institutions involved, including high costs, time-consuming processes, and significant error rates. The growing population and diversity of diseases further complicate the ability of the current healthcare systems to meet the demand. Additionally, inadequate working conditions for doctors have led us to develop a project to make improvements in this field.

Technological advancements, such as robotic applications and machine learning techniques, have inevitably started to automate the healthcare system due to their advantages. We are focusing on this transition process and aim to propose a transition model developed with novel technologies.

In our project, we have chosen to focus on breast cancer. Breast cancer accounts for 12.5% of all new annual cancer cases worldwide, making it the most common cancer in the world [1]. It is more common among women. One out of every eight women has the possibility of having breast cancer [2]. Breast cancer can become fatal in late stages, comprising 15.5% of cancer-related deaths [2]. However, if it is detected in early stages and treatment begins, about 70-80% recovery rate has been observed [3]. Therefore, periodic controls are necessary to enable early diagnosis, particularly mammography, which is recommended once a year for women between the ages of 40 and 69 for breast cancer screening [2].

We propose an end-to-end system that is a doctor and patient assistant for early breast cancer screening. Empowered by XAI (explainable AI), it helps doctors, especially doctors analyze breast cancer at early stages with higher accuracy. The flexibility and accessibility that our system provides make it easier for patients to follow their screening results and contact their doctors. The availability of many doctors on one platform allows patients to access many opinions, satisfying the desire of the second opinion market.

In our system's workflow, patients upload their mammography images, which are analyzed by a doctor, and our XAI model is used separately. Then, if the results match, the diagnosis will be finalized; otherwise, another doctor will review both results by taking advantage of the explainability of the model and give a terminal decision.

The key objectives of our project include improving healthcare accessibility, reducing diagnosis errors through doctor and XAI collaboration, and providing a more flexible and comfortable working environment for doctors. Ultimately, our pioneering work seeks to revolutionize breast cancer screening and contribute to broader improvements in the healthcare system by integrating medical expertise with cutting-edge technology for the benefit of both patients and healthcare providers.

1.2 Definitions, Acronyms, and Abbreviations

NoSQL Database: NoSQL databases, short for "not only SQL databases," eschew traditional table structures for data storage, employing alternative methods such as JSON document storage. They excel in performance, adeptly managing large volumes of data and heavy user traffic.

JSON: JSON stands for JavaScript Object Notation, serving as a user-friendly method for object representation. It offers ease of parsing and serves as an excellent format for data interchange.

Flutter: Flutter is a Dart-based open-source framework created by Google. It enables the development of native mobile applications for both iOS and Android platforms using a unified codebase.

Dart: A programming language based on object-oriented principles, widely favored for constructing applications that run seamlessly across multiple platforms.

UI: UI stands for user interface, denoting the visual and interactive elements of software or hardware systems enabling user interaction. The primary aim of a user interface is to streamline communication between users and the system, ensuring efficient achievement of user objectives.

Firebase: Firebase is a comprehensive platform developed by Google that provides various cloud-based services for building and managing web and mobile applications. It offers services such as real-time database, authentication, hosting, storage, and more, all integrated into a single platform. Firebase enables developers to quickly develop and deploy applications without managing infrastructure, allowing them to focus more on creating engaging user experiences. Currently, we are utilizing authentication, storage, and real-time database.

AWS: AWS stands for Amazon Web Services, a cloud computing platform offered by Amazon. It encompasses various cloud-based services like computing power, storage, database management, and content delivery. Currently, we're exclusively utilizing hosting and storage services.

FlutterFlow: FlutterFlow is a visual development platform that simplifies the creation of Flutter applications. It offers an intuitive interface for designing and connecting user interfaces to backend services without writing code. With FlutterFlow, developers can easily prototype, iterate, and deploy cross-platform Flutter apps.

KVKK: Personal Data Protection Law No. 6698 (6698 Sayılı Kişisel Verilerin Korunması Kanunu)

GDPR: General Data Protection Regulation

BI-RADS: Breast Imaging-Reporting and Data System

API: Application Programming Interface

HIPAA: Health Insurance Portability and Accountability Act

KETEM: Cancer Early Diagnosis, Screening and Education Center (Kanser Erken Teşhis, Tarama ve Eğitim Merkezi)

XAI: Explainable Artificial Intelligence

SageMaker: Amazon SageMaker is a managed service by AWS that builds, trains, and deploys machine learning models at scale. It simplifies ML development with built-in algorithms, scalable training, hyperparameter tuning, and easy deployment.

Lex chat: Amazon Lex is a fully managed artificial intelligence (AI) service with advanced natural language models to design, build, test, and deploy conversational interfaces in applications [4].

Route 53: Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. Route 53 connects user requests to internet applications running on AWS or on-premises [5].

Simple Email Service (SES): Amazon Simple Email Service (Amazon SES) lets you reach customers confidently without an on-premises Simple Mail Transfer Protocol (SMTP) email server using the Amazon SES API or SMTP interface [6].

Simple Storage Service (S3): Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can store and protect any amount of data for virtually any use case, such as data lakes, cloud-native applications, and mobile apps [7].

Elastic Compute Cloud (EC2): Amazon Elastic Compute Cloud (Amazon EC2) provides on-demand, scalable computing capacity in the Amazon Web Services (AWS) Cloud [8].

Elastic Load Balancer (ELB): Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple targets and virtual appliances in one or more Availability Zones (AZs) [9].

Load Balancer: A load balancer distributes workloads across multiple compute resources, such as virtual servers. Using a load balancer increases the availability and fault tolerance of your applications [10].

1.3 Overview

MediXAI is a cross-platform application that is built using Flutter and Firebase. It is designed to be used as a mobile app by the patients and a web application by the doctors. There are two types of users: patients and doctors. Upon the account creation and verification, patients can start breast cancer check-ups. They can upload their mammograms and additional documents like blood tests. Following that, they are asked for the payment. After payment completion, their check-up is assigned to a doctor, who assesses the mammogram. Our XAI model evaluates the mammogram at the same time. If both assessments yield the same result, a report will be provided to the patient with the BI-RADS classification and required steps. Otherwise, a second doctor's opinion will be consulted, and the results will be provided in the same format. If patients have questions regarding their check-up, they will be able to ask questions in the chat with the doctor part. Patients can also chat with the chatbot that was developed specifically for breast cancer detection and treatment. Doctors can answer the messages through the chat. They can ask for additional materials or follow-up mammograms to track the patient's progress.

2 Requirements Details

This section explains detailed information about how we built the MediXAI application. In general, our design is based on two main features of our application: time-saving for the patients and the doctors and extended accuracy with the help of XAI.

2.1 Functional Requirements

The following functional requirements are for the **patients**:

- The Turkish users must register/log in using their TR Identity No. The foreign users must register/log in using their Passport No.
- The user can upload/remove medical images in the .dcm format they want to be assessed by specifying when the image is taken.
- The user can see his/her already uploaded medical images.
- The user can request an assessment for diagnosis based on the selected medical images.
- The user can see the final diagnosis and take a report for that based on the images they uploaded.
- The user can see the optimal steps for the final diagnosis, such as making an appointment with an oncologist or requiring no further action.
- The user can see their old diagnosis decisions.
- The user can make payments for the medical image assessments with their preferred payment type and card.
- The user can add/remove the preferred payment type and card.

The following functional requirements are for the **doctors**:

- The user can register/log in using their username and password. (More would be required for registration, such as certificates.)
- The user can see the waiting medical image assessments.
- The user can access old images and diagnoses of the patient they assess.

- The user can see the explainable AI (XAI) diagnosis of the medical image they assess.
- The user can make an assessment and decide on the final diagnosis.
- The user can add notes and his/her suggestions to the final diagnosis and further steps.
- The user can see his/her accuracy rate compared to the XAI diagnoses.
- The user can see his/her income from assessing medical images.
- The user can add/remove bank account information.

The following functional requirements are for **doctors and patients**:

- The user can edit his/her profile.
- The user can delete his/her profile.

2.2 Non-functional Requirements

The non-functional requirements are analyzed under five sections.

2.2.1 Usability

The mobile application and web-based platform will feature an intuitive and user-friendly interface with a learning curve not exceeding 30 minutes for new users. Being a user-friendly application means users can achieve their planned operations as quickly as possible. For this reason, medical image uploading can be done in three clicks, and XAI explanations to the doctors will be accessible in two clicks. The system will support Turkish, English, and Arabic languages.

2.2.2 Reliability

The system is aimed to maintain a minimum uptime of 99% to ensure 24/7 availability. We will implement our system in a way that it is fault-tolerant. The system will implement robust encryption and access control mechanisms to ensure data integrity and prevent unauthorized access. We will integrate KPSPublic Web Service into our system to enable users to register and sign in to our system with TR ID No and hence enhance security. The system will recover from failures within 5 minutes and maintain core functionality even with faults. The system will follow GDPR and KVKK regulations in private data management.

2.2.3 Performance

Medical image uploads, processing, and assessment will have response times of 2 seconds, 5 seconds, and 3 seconds, respectively. The system will support a 200% increase in concurrent users without a decrease in response time. The system will ensure a minimum data transfer speed of 10 MBps for medical image uploads. System resource utilization will be at most 70% of available CPU and memory to ensure optimal performance.

2.2.4 Supportability

This application will initially be developed as a cross-platform mobile application on the patient side and a web application on the doctor side. For the upcoming releases, the application scope can be extended as a web application on the patient side, and new features can be added. System updates and maintenance will be scheduled during off-peak hours to minimize disruption, and the process will not exceed 1 hour. The system will provide a full suite of documentation, including FAQs, troubleshooting guides, and system architecture documentation, accessible from the user interface for patients and doctors.

2.2.5 Scalability

The system will automatically scale resources to accommodate a 100% increase in concurrent users within 5 minutes. Load balancing mechanisms will evenly distribute incoming requests across available servers to prevent overload and optimize performance. We aim to develop our application so that our ML model is usable across many platforms, including other applications, such as integration to e-Nabız.

2.3 Pseudo Requirements

MediXAI will have a mobile application for patients and a web application for doctors. Both applications will be implemented by using Flutter (also Dart). Firebase will be used to authenticate and store data, as well as the BaaS. AWS is also used for storage, hosting, and inference code.

● Flutter version is v3.16.1, which are the environment variables.

● Android Studio [11], Visual Studio Code [12], and FlutterFlow [13], Firebase Console [14] are used for Flutter and Firebase development.

● Users will be authenticated by entering their phone numbers via SMS. However, their full phone number will never be stored completely on the database. The encrypted version of the phone number will be saved. It increases the complexity of the authentication, but at the same time, it increases the anonymity since the phone numbers can not be readable from the database

● Firebase Analytics [15] will be used to track user flow in the app to get more information about users' behaviors. Moreover, possible bugs will be followed from here.

● All the external APIs and other data will be fetched through our server side. The reason for this, some APIs have keys that we do not prefer to embed into the mobile application since reverse engineering can reveal them.

● FlutterFlow [13] is used for UI design for mobile applications.

● The mobile application will be deployed to both the App Store [16] and the Google Play Store [17].

● The web application will be deployed on the web using Firebase's hosting service.

● Git [18] and GitHub [19] are used for the version control system.

● Jira [20] is used for project management.

● Telegram [21] is used for communication among the group.

● Each team member has to open a new branch while adding a new feature or fixing one.

● For each pull request, each member assigns a new reviewer. The GitHub Actions bot will assign a reviewer if he doesn't assign a reviewer.

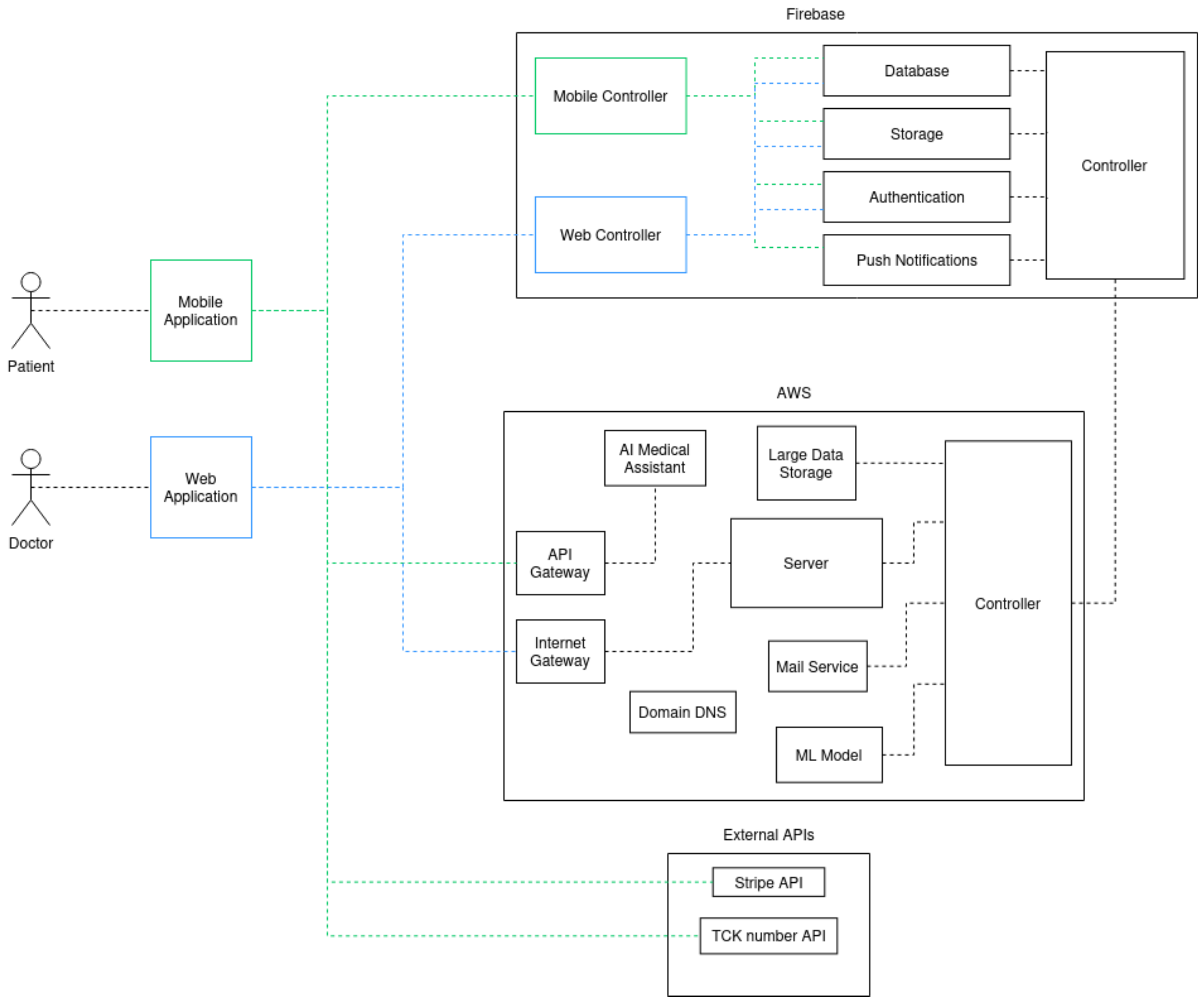
● Issue templates are prepared for new feature requests, bug reports, and an available custom issue template.

3 Final Architecture and Design Details

3.1 Overview

The architecture or system of the project comprises 5 main subsystems, which are the mobile application, the web application and integration with AWS, Firebase, and two external APIs. In the following sections, we'll go over subsystem decomposition, breaking down our system into the mentioned 5 parts and how they talk to each other. We'll also discuss hardware/software mapping, explaining how our system works with hardware and how different parts interact. Plus, we'll cover persistent data management, explaining how to save information for later use and ensure it stays available. Lastly, we'll talk about access control and security, detailing how we keep everything safe and only accessible to the right people. These sections will give you a clearer picture of how our system works behind the scenes.

3.2 Subsystem Decomposition



We have decided to build our application on five main components: Mobile Application, Web Application, AWS, Firebase, and External APIs.

Mobile Application: The mobile application is designed for patients to upload their mammography data, view the BI-RAD result conveyed according to the uploaded mammography data, and communicate with the radiologist who reviewed the mammography.

Web Application: The web application is intended for radiologists to evaluate incoming mammography data, communicate with patients, and review AI model results for the same data.

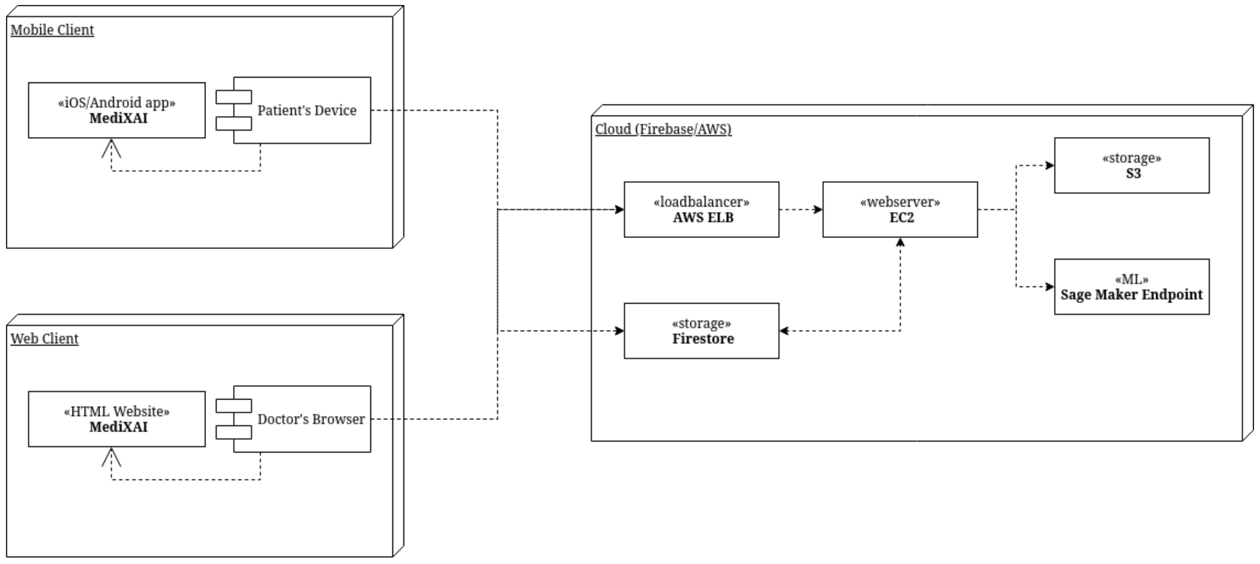
AWS: Amazon Web Services (AWS) is a crucial component of our architecture. It hosts the ML model that is responsible for evaluating incoming mammography data. Additionally, AWS provides services for email communication, large data storage, and Domain DNS management and hosts an AI medical assistant, which functions as a chatbot for users to discuss their medical concerns.

Firebase: Firebase is utilized for various purposes, including database management for user registration information, authentication, notifications, and storage of user medical data. It ensures secure and efficient data handling for our application's backend operations.

External APIs:

- **TCK Number API:** This API is integrated into our system for specific functionality related to TCK number validation or other relevant operations.
- **Stripe API:** The Stripe API is utilized for payment management within the application, ensuring secure and seamless transactions for users.

3.3 Hardware/Software Mapping



In our system architecture, we have two primary endpoints: the Patient Device, where users access our MediXAI application, and the Doctor's Browser, through which healthcare professionals interact with our system by accessing the MediXAI HTML website.

Patient Device:

- Users utilize our MediXAI application on their devices.
- Transactions initiated on the Patient Device are directed to either AWS or Firestore based on the request.

Doctor's Browser:

- Healthcare professionals access our system via their web browsers by directing to the MediXAI HTML website.

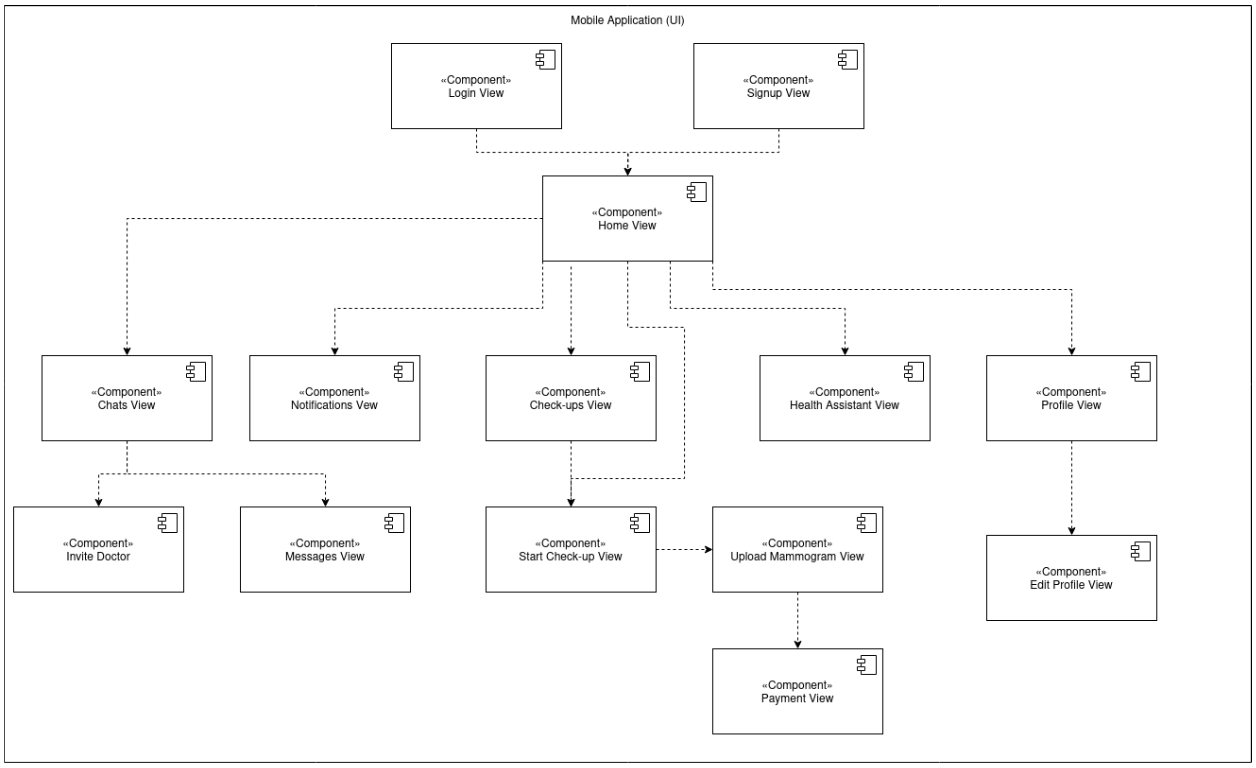
- Similar to the Patient Device, transactions from the Doctor's Browser are routed to either AWS or Firestore, depending on the request.

AWS Path:
- Requests from both the Patient Device and the Doctor's Browser are directed through AWS Elastic Load Balancer (ELB) for load distribution.
- The requests are then forwarded to server instances hosted on Amazon Elastic Compute Cloud (EC2).
- Depending on the nature of the request, transactions proceed to either Amazon S3 for storage or to a SageMaker Endpoint for machine learning model inference.

Firestore Side:
- Firestore handles requests made to it.
- Firestore is connected to EC2 instances, enabling bidirectional communication between Firestore and AWS.
- This connectivity allows for seamless integration between Firestore and AWS services, facilitating the exchange of data and requests between the two platforms.

3.4 Subsystem Services

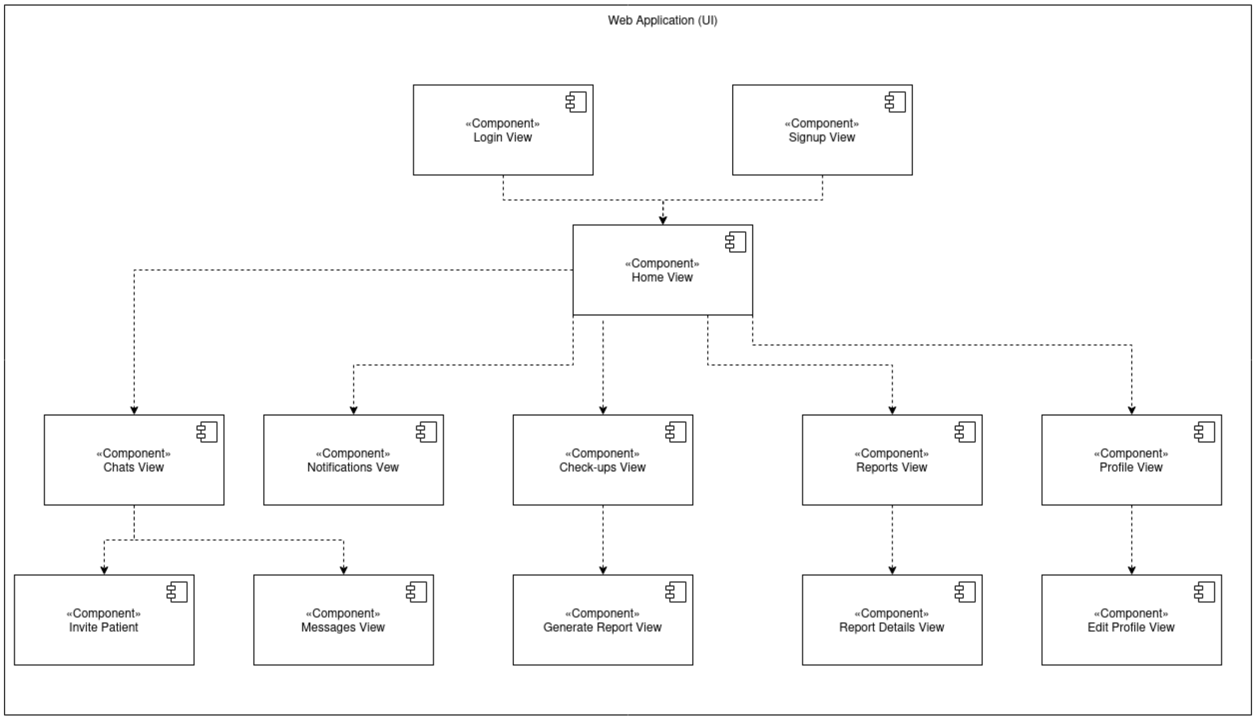
3.4.1 Mobile Application


Mobile Application (UI)
«Component» Login View
«Component» Signup View
«Component» Home View
«Component» Chats View
«Component» Notifications Vew
«Component» Check-ups View
«Component» Health Assistant View
«Component» Profile View
«Component» Invite Doctor
«Component» Messages View
«Component» Start Check-up View
«Component» Upload Mammogram View
«Component» Edit Profile View
«Component» Payment View

Login View: This view contains the Login screen.

Signup View: This view contains the Signup screen.
Home View: This view contains a screen where there is a dashboard with which patients can visit other Views.
Chats View: This view contains a screen where patients can see their chat history with others and also link to both Invite Doctor View and Messages View.
Invite Doctor View: In this view, patients can send chat invitations to their assigned doctor via a button.
Messages View: In this view, patients can review their messaging with a doctor.
Notifications View: This view contains notifications that come to patients.
Check-up View: This view contains a screen where patients can see all their check-ups made so far.
Start Check-up View: Via a button in this view, patients can start the process of starting a checkup and they will be directed to Upload Mammogram View.
Upload Mammogram View: In this view, patients can upload their mammogram data to the application. After uploading, they will be directed to Payment View if they click the “Continue” button.
Payment View: From this view, patients can pay their fee for forming a check-up.
Profile View: This view contains a screen where patients can see their information.
Edit Profile View: This view allows patients to edit their profile information.

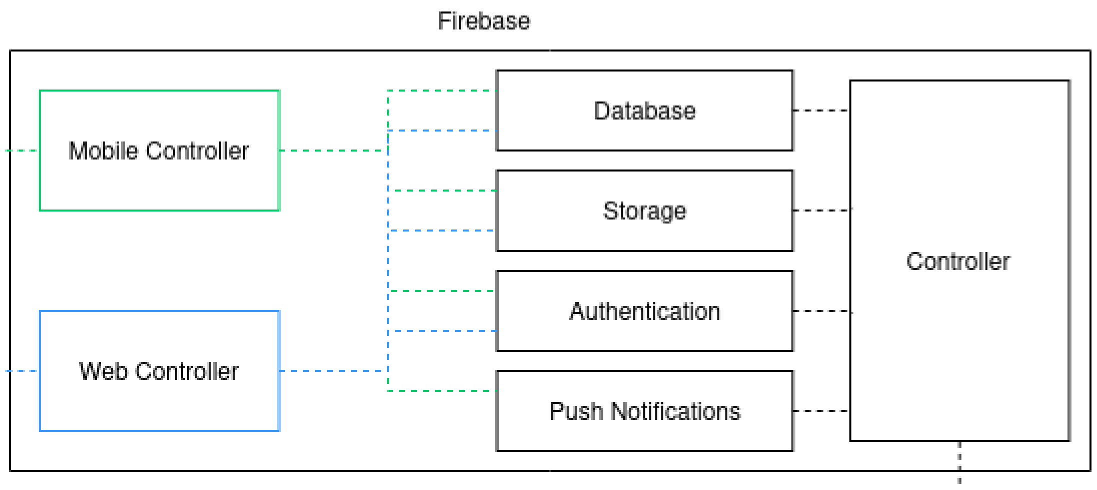
3.4.2 Web Application


Web Application (UI)
«Component» Login View
«Component» Signup View
«Component» Home View
«Component» Chats View
«Component» Notifications Vew
«Component» Check-ups View
«Component» Reports View
«Component» Profile View
«Component» Invite Patient
«Component» Messages View
«Component» Generate Report View
«Component» Report Details View
«Component» Edit Profile View

Login View: This view contains the Login screen.
Signup View: This view contains the Signup screen.
Home View: This view contains a screen where there is a dashboard with which doctors can

visit other views.
Chats View: This view contains a screen where doctors can see their chat history with others and also link to both Invite Patient View and Messages View.
Invite Patient View: In this view, doctors can send chat invitations to their patients via a button.
Messages View: In this view, doctors can review their messaging with a patient.
Notifications View: This view contains notifications that come to doctors.
Check-up View: This view contains a screen where doctors can see all the check-ups assigned to them.
Generate Report View: This view allows doctors to generate a report about their diagnosis on a check-up assigned to them.
Reports View: This view contains a screen where doctors can see all the reports they generated so far.
Report Details View: This view contains a screen where doctors can review the details of a chosen report.
Profile View: This view contains a screen where doctors can see their information.
Edit Profile View: This view allows doctors to edit their profile information.

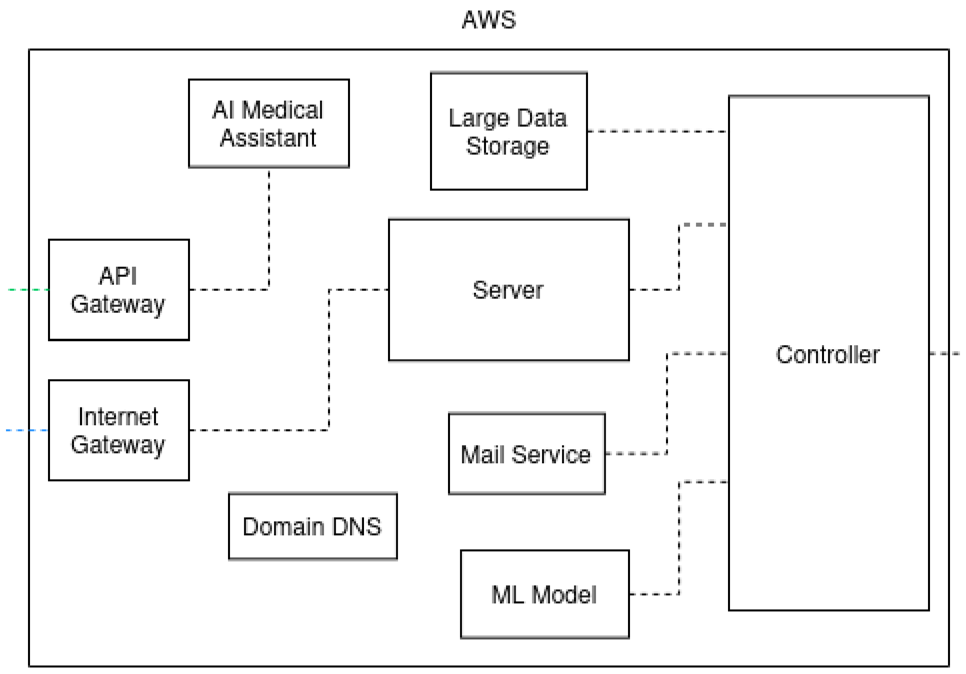
3.4.3 Firebase



Web Controller: The Web Controller handles requests initiated from the doctor's side and interacts with Firebase to perform various operations such as retrieving patient data, updating medical records, and communicating with the backend system.
Mobile Controller: The Mobile Controller manages requests made from the patient's side through the mobile application and communicates with Firebase to handle tasks like uploading mammogram data, accessing medical reports, and facilitating communication with healthcare providers.
Database: The Database component in Firebase stores structured data related to the breast cancer mammogram project, including patient profiles, medical records, mammogram results, and communication logs between patients and doctors.
Storage: The Storage feature in Firebase provides scalable cloud storage for storing large files such as mammogram images, patient documents, and other medical data. It ensures secure and reliable storage for user-generated content associated with the project.

Authentication: Firebase Authentication handles user authentication and authorization, ensuring secure access to the application. It supports various authentication methods such as email/password, phone number authentication, and third-party providers, allowing users to log in and access their medical data securely.
Push Notifications: Push Notifications enable real-time communication and updates for users of the application. Firebase Cloud Messaging (FCM) is used to send push notifications to patients and doctors, notifying them of new medical reports, appointment reminders, and important updates related to the project.

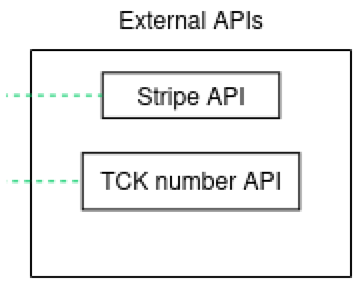
3.4.4 AWS



AI Medical Assistant (Lex chat): The AI Medical Assistant, powered by Amazon Lex, provides a conversational interface for users to interact with the application. It understands natural language queries and provides responses, helping users navigate the application, access medical information, and receive assistance with their healthcare concerns.
Domain DNS (Route 53): Route 53 is used to manage the domain name system (DNS) for our project. It translates human-readable domain names into IP addresses, ensuring users can access the application using a custom domain name. Route 53 also provides features like domain registration, DNS health checks, and routing policies for improved reliability and performance.
ML Model (SageMaker Endpoint): SageMaker Endpoint hosts the machine learning model used in the application for analyzing mammogram images and providing diagnostic insights. It serves as an endpoint for real-time inference, allowing the application to make predictions based on incoming data.
Mail Service (Simple Email Service (SES)): The Mail Service, powered by Amazon Simple Email Service (SES), facilitates email verification and report distribution within the app. It

ensures reliable email delivery, enables email verification for user registration, and facilitates the automated sending of medical reports to patients and healthcare providers.
Large Data Storage (Simple Storage Service (S3)): S3 is utilized for large-scale storage of data associated with MediXAI project, including mammogram images, medical records, and other healthcare-related files.
Server (Elastic Compute Cloud (EC2)): EC2 instances serve as the underlying compute infrastructure for hosting backend services, web applications, and other components of the MediXAI project. EC2 offers scalable virtual servers in the cloud, providing flexibility and control over computing resources to support the application's requirements for processing data, running algorithms, and serving user requests.

3.4.5 External APIs



TCK Number API: This API is integrated into our system for TCK number validation.
Stripe API: The Stripe API is utilized for payment management within the application, ensuring secure and seamless transactions for users.

4 Development/Implementation Details

Under this heading, the subsystem details will be shared in two headings: User Interface and Server subystems.

4.1 User Interface Subsystem

To operate the MediXAI applications more easily from patients' and doctors' perspectives, the application has shrunk the number of pages as much as possible while customizing the app. When a patient enters the application, the user will enter the home page. Other pages can only be reached from the home page. When a doctor enters the application Main_Home page welcomes him. Other pages are reachable from the Main_Home page.

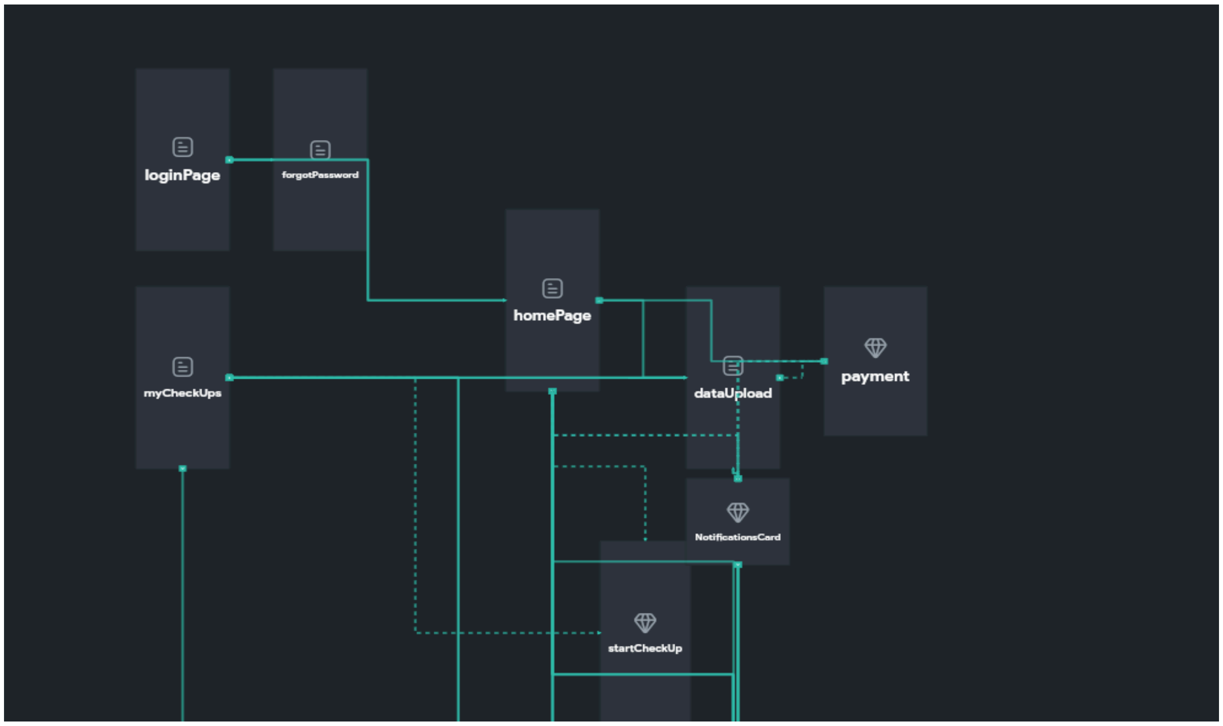


Figure: Mobile Application Subsystem a

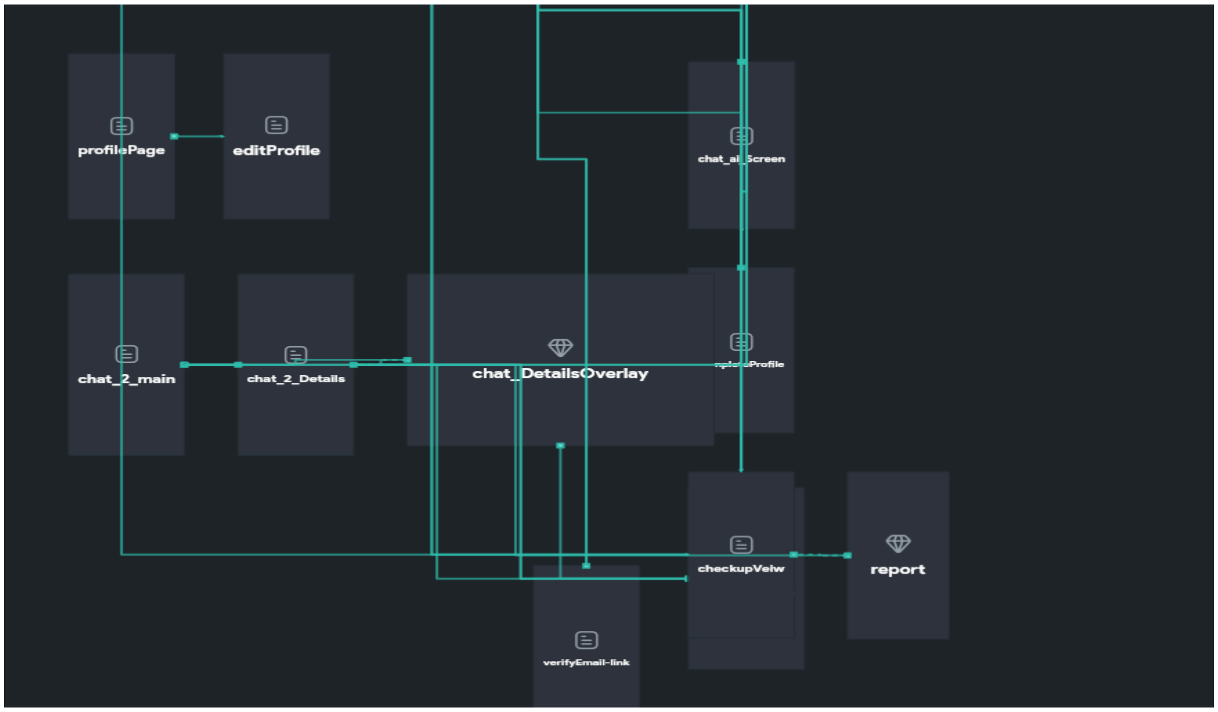


Figure: Mobile Application Subsystem b

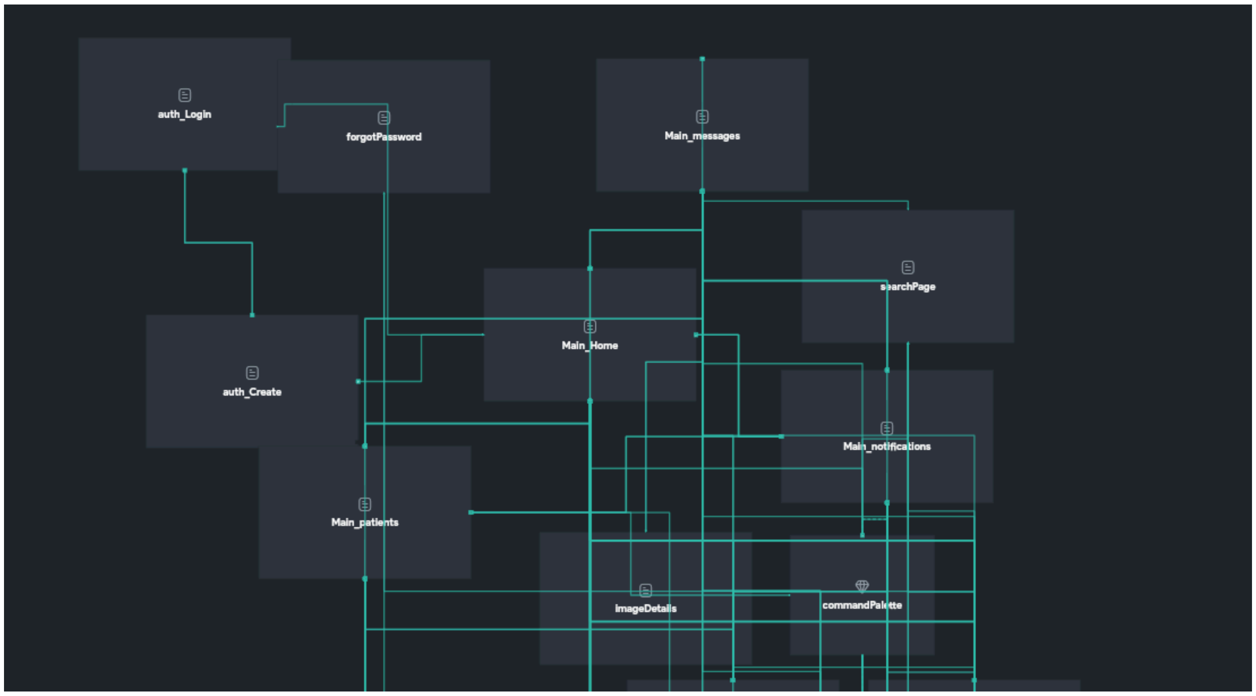


Figure: Web Application Subsystem a

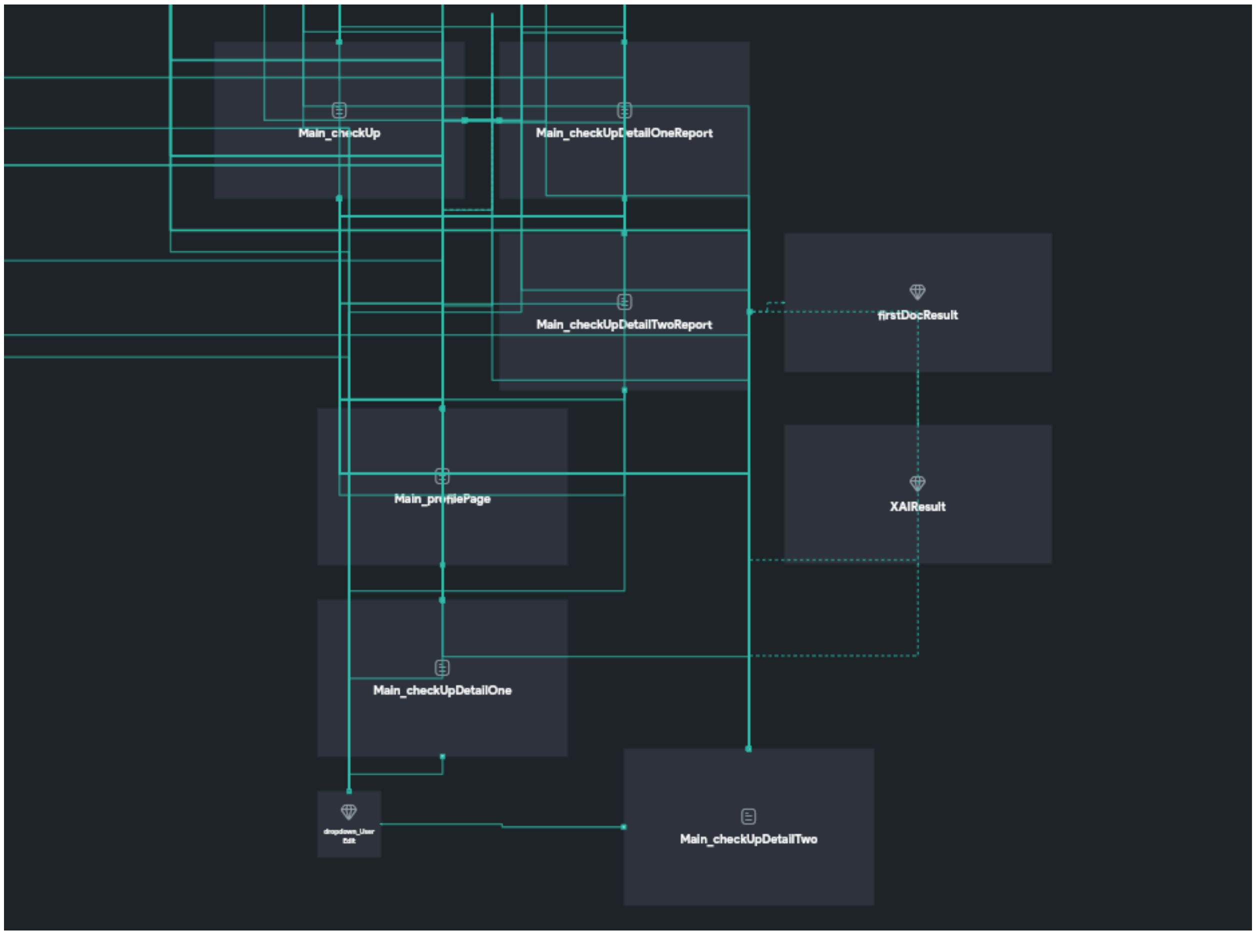


Figure: Web Application Subsystem b

4.2 Server Subsystem

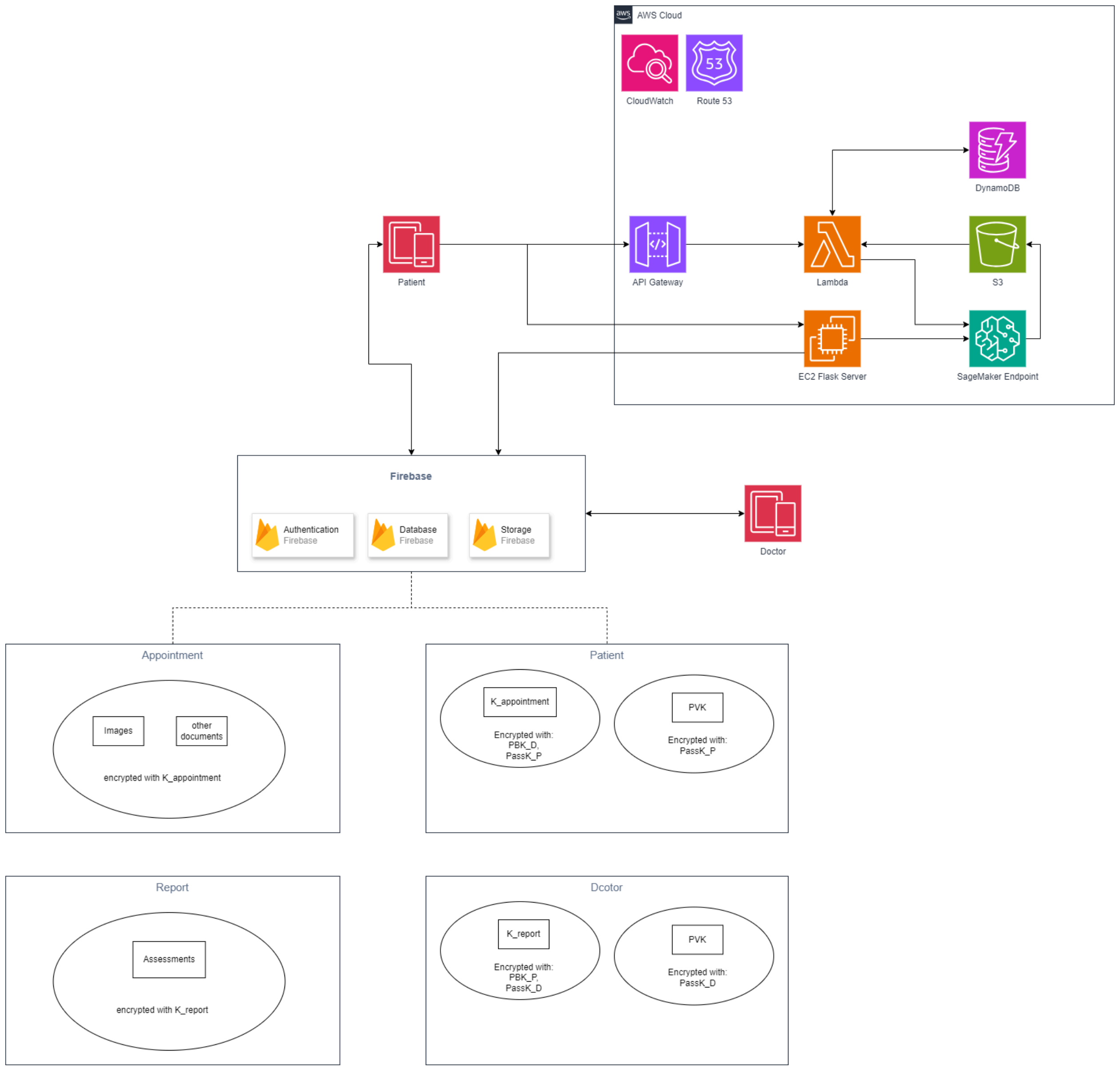


Figure: The Server Subsystem

Backend of MediXAI is connected to the Firebase and sends requests to the AWS. In AWS, the API Gateway manages SageMaker Endpoint requests which contain S3 Bucket, DynamoDB, and Lambda Requests. Authentication, verification, and messaging are managed via built-in services in Firebase.

5 Test Cases

Test ID Format (from left to right):
First two digit: 00 (Functional), 01 (Non-functional)
Third digit: 0 (mobile app), 1 (web app), 2 (XAI Model)
Last two digit: incremental

Patient Signup

Test ID	00001	**Category**	Functional	**Severity**	Critical
Objective	1-To verify that patients can successfully sign up for the mobile app. 2-To verify that patients cannot sign up and get an error message when they enter invalid information for the mobile app.				
Steps	1. Open the mobile app. 2. Click on the signup button. 3. Enter valid/invalid patient details (name, email, password). 4. Click on the signup button.				
Expected Result	1-The patient should be registered successfully and redirected to the login page. 2-The patient should not be registered and should get an error message.				
Date Tested & Test Result	Tested on 27/04/2024. Test passed.				

Patient Login

Test ID	00002	**Category**	Functional	**Severity**	Critical
Objective	1-To verify that patients can successfully login to the mobile app. 2-To verify that patients cannot login and get an error message when they enter invalid information for the mobile app.				

Steps	1. Open the mobile app. 2. Tap on the login button. 3. Enter valid/invalid patient credentials (email and password). 4. Tap on the login button.
Expected Result	1-Patient should be successfully logged in and redirected to the home page. 2-The patient should not be logged in and should get an error message.
Date Tested & Test Result	Tested on 27/04/2024. Test passed.

Doctor Signup

Test ID	00101	**Category**	Functional	**Severity**	Critical
Objective	1-To verify that doctors can successfully sign up for the web app. 2-To verify that doctors cannot sign up and get an error message when they enter invalid information for the web app.				
Steps	1. Open the web app. 2. Click on the signup button. 3. Fill in the required doctor information (name, email, password, doctor verification info). 4. Click on the signup button.				
Expected Result	1-Doctor should be successfully registered and redirected to the login page. 2-The doctor should not be registered and should get an error message.				
Date Tested & Test Result	Tested on 27/04/2024. Test passed.				

Doctor Login

Test ID	00102	**Category**	Functional	**Severity**	Critical
Objective	1-To verify that doctors can successfully login to the web app. 2-To verify that doctors cannot login and get an error message when they enter invalid information for the web app.				
Steps	1. Open the web app. 2. Enter valid doctor credentials (email and password). 3. Click on the login button.				
Expected Result	1-Doctor should be successfully logged in and redirected to the dashboard. 2-The doctor should not be logged in and should get an error message.				
Date Tested & Test Result	Tested on 27/04/2024. Test passed.				

Start a Check-Up

Test ID	00003	**Category**	Functional	**Severity**	Critical
Objective	To confirm patients can start a check-up.				
Steps	1. Log in to the patient app. 2. Navigate to the Check-Up section. 3. Click on "Start Check-Up" button.				
Expected Result	Check-up request recorded, waiting for user medical data.				
Date Tested & Test Result	Tested on 28/04/2024.				

	Test passed.

Upload Medical Data

Test ID	00004	**Category**	Functional	**Severity**	Critical
Objective	To verify that patients can upload their medical data.				
Steps	1. Login to the mobile app. 2. Navigate to the Check-up section. 3. Click on the Start Check-up button. 4. Click on the Upload Data button. 5. Select a medical data file (mammogram images). 6. Click on the upload button.				
Expected Result	Files should be successfully uploaded to the cloud storage and their path to the database.				
Date Tested & Test Result	Tested on 28/04/2024. Test passed.				

View Patient's Medical Data

Test ID	00103	**Category**	Functional	**Severity**	Critical
Objective	To verify doctors can view patient's medical data.				
Steps	1. Login to the web app as a doctor. 2. Navigate to the patient list. 3. Click on a patient's check-up. 4. View the uploaded medical data.				

Expected Result	Doctor should be able to view patient's uploaded medical data.
Date Tested & Test Result	Tested on 28/04/2024. Test passed.

Generate Report

Test ID	00104	**Category**	Functional	**Severity**	Critical
Objective	To verify the report generation process.				
Steps	1. Make a doctor analyze patient's medical data. 2. Make XAI model analyze patient's medical data. 3. Trigger the report generation process.				
Expected Result	A comprehensive report should be generated based on the medical data analysis by doctor and XAI model.				
Date Tested & Test Result	Tested on 28/04/2024. Test passed.				

View Report

Test ID	00005	**Category**	Functional	**Severity**	Critical
Objective	To verify that patients can view their check-up reports.				
Steps	1. Login to the mobile app as a patient. 2. Navigate to the My Check-ups section. 3. Click on the View Report button.				

Expected Result	Patient should be able to view the generated check-up report.
Date Tested & Test Result	Tested on 28/04/2024. Test passed.

XAI Model Training & Test

Test ID	00201	**Category**	Functional	**Severity**	Critical
Objective	To verify the training and testing of the XAI model.				
Steps	1. Prepare medical image dataset. 2. Train model on that dataset with tuning hyperparameters. 3. Get the test result. 4. Evaluate the result.				
Expected Result	XAI model should achieve good scores based on determined metrics (e.g. accuracy, f1-score)				
Date Tested & Test Result	Tested on 05/05/2024. Test failed. Additional steps are taken to improve the model performance.				

View Waiting Assessments

Test ID	00105	**Category**	Functional	**Severity**	Critical
Objective	To verify doctors can see pending assessments.				

Steps	1. Log in to the doctor app. 2. Navigate to the "My Patients" section.
Expected Result	Doctors can see pending assessments for their patients.
Date Tested & Test Result	Tested on 01/05/2024. Test passed.

Assess Medical Data

Test ID	00106	**Category**	Functional	**Severity**	Critical
Objective	To confirm doctors can assess medical data.				
Steps	1. Log in to the doctor app. 2. Select a pending assessment. 3. Review uploaded image and other medical data. 4. Give a BI-RADS score for mammograms. 5. Add notes and suggestions. 6. Click on "Submit Assessment" button.				
Expected Result	Analysis provided with optional notes.				
Date Tested & Test Result	Tested on 28/04/2024. Test passed.				

View First Doctor’s Diagnosis

Test ID	00107	**Category**	Functional	**Severity**	Critical

Objective	To ensure the second doctors can access the first doctor’s diagnosis.
Steps	1. Log in to the doctor app. 2. Select a pending assessment. 3. Click on “First Doctor Analysis” button.
Expected Result	The first doctor’s diagnosis displayed to the second doctor.
Date Tested & Test Result	Tested on 01/05/2024. Test passed.

View XAI Diagnosis

Test ID	00108	**Category**	Functional	**Severity**	Critical
Objective	To ensure the second doctors can access XAI diagnosis.				
Steps	4. Log in to the doctor app. 5. Select a pending assessment. 6. Click on “XAI Analysis” button.				
Expected Result	XAI diagnosis displayed to the second doctor.				
Date Tested & Test Result	Tested on 01/05/2024. Test passed.				

Edit Profile for Patients

Test ID	00006	**Category**	Functional	**Severity**	Minor
Objective	To verify patients can edit their profiles.				

Steps	1. Log in to the mobile app. 2. Navigate to the “My Profile” section. 3. Click on the "Edit Profile" button. 4. Update necessary information. 5. Click on the "Save Changes" button.
Expected Result	Patient profile updated successfully.
Date Tested & Test Result	Tested on 28/04/2024. Test passed.

Edit Profile for Doctors

Test ID	00109	**Category**	Functional	**Severity**	Minor
Objective	To verify doctors can edit their profiles.				
Steps	1. Log in to the web app. 2. Navigate to the “My Profile” section. 3. Click on the "Edit Profile" button. 4. Update necessary information. 5. Click on the "Save Changes" button.				
Expected Result	Doctor profile updated successfully.				
Date Tested & Test Result	Tested on 28/04/2024. Test passed.				

Notification for Pending Assessments

Test ID	00110	Category	Functional	Severity	Major
Objective	To ensure doctors receive notifications for pending assessments.				
Steps	1. Log in as a doctor. 2. Wait for a new check-up request.				
Expected Result	Doctor receives a notification indicating a new assessment is pending.				
Date Tested & Test Result	Tested on 08/05/2024. Test passed.				

Notifications for Patients

Test ID	00007	Category	Functional	Severity	Major
Objective	To ensure patients can get notifications.				
Steps	1. Click on “Notifications” button. 2. Get for different events (e.g., new diagnosis, chat messages).				
Expected Result	Patients receive notifications successfully.				
Date Tested & Test Result	Tested on 28/04/2024. Test passed.				

Message Sending for Patients

Test ID	00008	Category	Functional	Severity	Major
Objective	To verify that patients can send messages to doctors.				

Steps	1. Log in as a patient. 2. Navigate to the chat section. 3. Select a doctor and compose a message. 4. Click send.
Expected Result	Message is sent successfully to the selected doctor.
Date Tested & Test Result	Tested on 28/04/2024. Test passed.

Message Receiving for Patients

Test ID	00009	**Category**	Functional	**Severity**	Major
Objective	To confirm that patients receive messages from doctors.				
Steps	1. Log in as a patient. 2. Check the chat for new messages from doctors.				
Expected Result	Patient receives the message sent by doctors.				
Date Tested & Test Result	Tested on 28/04/2024. Test passed.				

Message Sending for Doctors

Test ID	00111	**Category**	Functional	**Severity**	Major
Objective	To verify that doctors can send messages to patients.				

Steps	5. Log in as a doctor. 6. Navigate to the chat section. 7. Select a patient and compose a message. 8. Click send.
Expected Result	Message is sent successfully to the selected patient.
Date Tested & Test Result	Tested on 28/04/2024. Test passed.

Message Receiving for Doctors

Test ID	00112	**Category**	Functional	**Severity**	Major
Objective	To confirm that doctors receive messages from patients.				
Steps	3. Log in as a doctor. 4. Check the chat for new messages from patients.				
Expected Result	Doctor receives the message sent by patients.				
Date Tested & Test Result	Tested on 28/04/2024. Test passed.				

Successful Payment

Test ID	00010	**Category**	Functional	**Severity**	Critical
Objective	To verify that patients can successfully make payments for check-up.				

Steps	1. Log in as a patient. 2. Navigate to the payment section. 3. Select an assessment to pay for. 4. Enter payment details and confirm payment.
Expected Result	Payment is processed successfully, and the check-up is marked as paid.
Date Tested & Test Result	Tested on 27/04/2024. Test passed.

XAI Model Integration

Test ID	01201	**Category**	Non-Functional	**Severity**	Critical
Objective	To verify the integration of the XAI model.				
Steps	1. Upload a sample medical data. 2. Verify the XAI model's prediction results on the database.				
Expected Result	XAI model should accurately analyze the medical data, provide relevant insights and write the prediction to the database.				
Date Tested & Test Result	Tested on 01/05/2024. Test passed.				

Device Compatibility

Test ID	01001	**Category**	Non-Functional	**Severity**	Major
Objective	To verify that the mobile application work across different devices.				
Steps	1. Use the mobile app on various devices (Android, iOS).				

Expected Result	App should function properly across different devices without any issues.
Date Tested & Test Result	Tested on 05/05/2024. Test passed.

Browser Compatibility

Test ID	01101	**Category**	Non-Functional	**Severity**	Major
Objective	To verify that the web application work across different browsers.				
Steps	1. Access the web app from different browsers (Chrome, Firefox).				
Expected Result	App should function properly across different browsers without any issues.				
Date Tested & Test Result	Tested on 05/05/2024. Test passed.				

Data Encryption

Test ID	01002	**Category**	Non-Functional	**Severity**	Critical
Objective	To verify data security through encryption.				
Steps	1. Review the data transmission process. 2. Verify data encryption protocols are implemented (for password or other sensitive info).				
Expected Result	Patient's data should be encrypted during transmission and storage.				

Date Tested & Test Result	Tested on 27/04/2024. Test passed.

Response Time

Test ID	01003	**Category**	Non-Functional	**Severity**	Major
Objective	To test the system's response time.				
Steps	1. Upload medical data. 2. Analyze the data and generate reports.				
Expected Result	System should respond within acceptable time limits for all operations.				
Date Tested & Test Result	Tested on 05/05/2024. Test passed.				

Scalability

Test ID	01004	**Category**	Non-Functional	**Severity**	Major
Objective	To test the system's ability to handle increased load.				
Steps	1. Simulate increased user activity. 2. Monitor system performance.				
Expected Result	System should maintain performance even under high loads without crashing or slowing down significantly.				
Date Tested & Test Result	Tested on 05/05/2024. Test passed.				

Reliability

Test ID	01005	**Category**	Non-Functional	**Severity**	Major
Objective	To verify the reliability of the applications.				
Steps	1. Perform tasks repeatedly. 2. Monitor for crashes or unexpected behavior.				
Expected Result	Applications should function reliably without frequent crashes or errors.				
Date Tested & Test Result	Tested on 05/05/2024. Test passed.				

Multi-device Synchronization for Patient App

Test ID	01006	**Category**	Non-Functional	**Severity**	Major
Objective	To confirm synchronization of data across multiple devices for patients.				
Steps	1. Log in as a patient on multiple devices simultaneously. 2. Upload an image from one device and check if it reflects on the other.				
Expected Result	Data is synchronized across all logged-in devices in real-time.				
Date Tested & Test Result	Tested on 05/05/2024. Test passed.				

Multi-device Synchronization for Doctor App

Test ID	01102	**Category**	Non-Functional	**Severity**	Major
Objective	To confirm synchronization of data across multiple devices for doctors.				

Steps	1. Log in as a doctor on multiple devices simultaneously. 2. Analyze an image and check if data synchronized on the other device.
Expected Result	Data is synchronized across all logged-in devices in real-time.
Date Tested & Test Result	Tested on 05/05/2024. Test passed.

Patient App Installation

Test ID	01007	**Category**	Non-Functional	**Severity**	Critical
Objective	To verify app installation process without any problem.				
Steps	1. Install app on different devices.				
Expected Result	App installs without errors.				
Date Tested & Test Result	Tested on 11/05/2024. Test passed.				

Doctor App Deployment

Test ID	01103	**Category**	Non-Functional	**Severity**	Critical
Objective	To ensure deployment of the web app without any problem.				
Steps	1. Deploy web app on a server. 2. Try to access it by providing the domain name.				

Expected Result	Web app accessible without issues.
Date Tested & Test Result	Tested on 05/05/2024. Test passed.

Usability for Patient App

Test ID	01008	**Category**	Non-Functional	**Severity**	Major
Objective	To evaluate the user-friendliness of the patient app.				
Steps	1. Provide a sample user with the patient app. 2. Ask them to perform common tasks such as check-up request and image upload.				
Expected Result	User finds the app intuitive and easy to use.				
Date Tested & Test Result	Tested on 08/05/2024. Test passed.				

Usability for Doctor App

Test ID	01104	**Category**	Non-Functional	**Severity**	Major
Objective	To evaluate the user-friendliness of the doctor app.				
Steps	1. Provide a sample doctor with the doctor app. 2. Ask them to perform common tasks such as reviewing all pending analyses and providing analysis.				
Expected Result	Doctor finds the app efficient and easy to navigate.				

Date Tested & Test Result	Tested on 08/05/2024. Test passed.

6 Maintenance Plan and Details

Our application, as all applications that deal with real world situations, requires continuous maintenance and fixes for the possible issues that might arise or that are discovered later on. Therefore we have devised a clear maintenance plan to ensure that our application holds the highest standards and performs as expected at all times. We divide our maintenance plan into five different categories. Those categories are preventive maintenance, corrective maintenance, adaptive maintenance, perfective maintenance, and operational maintenance. The details of each of those categories can be found below.

6.1 Preventive Maintenance

We will be following strict preventive maintenance plans, starting with routine checkups for our system for the early detection of any anomalies or unusual behavior. Additionally, we will be conducting regular updates for our system, making sure that we always have the latest versions of all the packages, and always stick with supported versions. We will also conduct security audits in order to be able to detect any vulnerabilities in our system, which will help to prevent any later problems.

6.2 Corrective Maintenance

The next step is to deal with issues that occur during the time when the application is in production. Any abnormal behavior will be detected immediately and whenever a bug is either reported or detected by our developers, we will work on patching it as soon as possible. We will specifically give priority to bugs related to user data security and privacy since our application will contain sensitive patient information that should not be compromised under any circumstance.

6.3 Adaptive Maintenance

Since we are working in the medical field, we have to closely follow the regulations and laws that are directly or indirectly related to our application domain. In the case of any regulation change, we have to make sure that our application still conforms with the current regulations and if it doesn't, we should adapt it so that it would be suitable with that new change. Additionally, we use advanced machine learning techniques in our application, which should be always adapted to the different requirements according to the current demand in the market, and the user expectations.

6.4 Perfective Maintenance

We will be gathering feedback from both doctors and patients and we will have as our top priority the satisfaction of our users. We will be responding to user suggestions and recommendations and we will be taking them seriously. We will also try over time to make the application easier to use, especially for elderly people and less technical users. Since this application targets patients that are of senior age, this is a crucial requirement.

6.5 Operational Maintenance

Since the patients and doctors will be relying on our platform for long-term breast cancer diagnosis and follow-ups for their medical cases, we have to make our systems as reliable as possible. For this reason, we have to make sure that we perform regular backups and that we have redundancy in our architecture.

7 Other Project Elements

7.1 Consideration of Various Factors in Engineering Design

7.1.1 Constraints

One of the primary constraints we encountered during the project was related to resources, particularly while using Amazon Web Services (AWS) for our computational needs. Despite AWS's capabilities, we faced significant memory constraints which posed challenges during the training of the Resnet152 model with high-resolution mammography images. This limitation affected the efficiency and speed of our model training phases, necessitating optimization and adjustment of our resource allocation to mitigate these issues.

Additionally, obtaining high-quality, diverse datasets for training our AI was a significant hurdle. The availability of qualified datasets that are representative of various demographics is crucial for the accuracy and reliability of AI diagnostics. Limited access to such datasets could potentially bias our model's performance and affect its generalizability across different populations.

7.1.2 Standards

In the development of MediXAI, we adhered to several key standards to ensure the reliability and safety of our system. These included HIPAA (Health Insurance Portability and Accountability Act) guidelines to ensure the privacy and security of patient data. We also followed technical standards related to AI in healthcare, such as those outlined by the IEEE (Institute of Electrical and Electronics Engineers) and ISO (International Organization for Standardization), which dictate the safety, efficacy, and ethical deployment of artificial intelligence technologies in medical settings.

7.2 Ethics and Professional Responsibilities

The ethical implications of AI in healthcare are profound. In designing MediXAI, we prioritized ethical considerations such as patient consent, transparency of AI processes, and the right to a human-mediated review in cases where the AI's diagnosis differs from that of the human doctors. Ensuring that our system enhances the diagnostic process without replacing the essential human element of medical practice was a guiding principle throughout our project development. Furthermore, by taking user data encrypted, we ensure patient privacy.

7.3 Teamwork Details

7.3.1 Contributing and Functioning Effectively on the Team

All significant decisions are made collectively following thorough discussion. Every project member participated in report-writing sessions. Each team member was readily available and approachable for communication. The work distribution in detail is the following:

Ahmet Arda: He was responsible for the machine learning part with Mustafa. He was involved in dataset finding, model implementation and explainability parts. He was also involved in preprocessing, training and deploying the model on AWS by using AWS SageMaker. Additionally, he arranged some meetings with professionals in the healthcare field.

Mahmut Furkan: He was responsible for the web application for the doctors. He implemented some functionalities specific to the doctors like report generation. He integrated the already prepared frontend with the Firebase and made some modifications as needed. He also performed general testing on the web application to verify that the system was working correctly.

Mustafa: He was responsible for the machine learning part with Ahmet. He was involved in dataset finding, model implementation and adding explainability to the model. He was also involved in training and deploying the model on AWS by using AWS SageMaker.

Omar: He was responsible for the patients' mobile application. He worked on improving the UI and integrating it with Firebase. He is also working on the AWS side of the project by setting up and integrating services like Simple Email Service (SES), Lex chat, Cloud privacy and security, S3, and EC2. He implemented the inference system and also the endpoint for the XAI model. He also implemented APIs like TC Verification and email verification that are used in the project. Moreover, he performed general testing on the mobile application to verify that the system was working correctly.

Ömer Kağan: He was responsible for the frontend part of both mobile and web applications. Additionally, he worked with Mahmut in the development of doctors' web application. He performed general testing on the web application to verify that the system was working correctly. He also worked on XAI model deployment on cloud (AWS SageMaker) and send inference to the database.

7.3.2 Helping Creating a Collaborative and Inclusive Environment

In the project, we emphasized a collaborative and inclusive work environment by strategically dividing tasks based on each team member's interests, skills and past experiences. This ensured everyone was engaged and contributed to areas where they were most passionate and knowledgeable. Our project was separated into software development and machine learning teams, fostering specialization while encouraging cross-team collaboration for shared insights and problem-solving. Weekly discussion meetings were crucial, especially in the early stages, providing a platform for everyone to freely express opinions and ideas, thereby fostering a culture of open communication and mutual respect. Thus, we shared work fairly and everyone's ideas were considered. In our project, we made sure everyone could talk openly and support one another. This helped us to form a positive space where every team member felt important.

7.3.3 Taking Lead Role and Sharing Leadership on the Team

In our team, we spread out leadership roles and we did not have a particular leader in the team. This means anyone can lead when it's their moment to express himself, it gives us all a chance to show our leadership skills. We all took charge of necessary situations, acted responsibly, and helped each other. In this mindset, every team member takes ownership of their role in the project and actively contributes to its success. This way of sharing leadership does not just bring us closer as a team, it also makes us more effective in achieving our goals as a team.

7.3.4 Meeting objectives

We held regular weekly meetings as a team throughout the development of the project. The primary objectives of our project meetings were to foster effective communication, ensure alignment with project goals and facilitate collaborative problem-solving. Throughout the project lifecycle, our meetings served as pivotal checkpoints to review progress, identify challenges and strategize solutions as a team. By establishing clear meeting agendas and objectives, we aimed to optimize time management and keep all team members engaged and informed. These meetings were instrumental in promoting transparency, accountability, and teamwork, ultimately driving us closer to achieving our project milestones and deliverables.

7.4 New Knowledge Acquired and Applied

In the realm of AI, we learned about the critical role of image preprocessing techniques in enhancing model training effectiveness. Specifically, we implemented the Contrast Limited Adaptive Histogram Equalization (CLAHE) filter to improve the contrast of mammography images, which is crucial for highlighting subtle features indicative of potential abnormalities. Additionally, we applied cropping techniques to the images to reduce the influence of the black background commonly present in mammograms. This step was vital as it helped minimize unnecessary noise and focus the AI's analysis on the relevant breast tissue, thereby improving the accuracy and reliability of our diagnostic predictions.

On the software development front, we discovered the importance and the ease of integration of Application Programming Interfaces (APIs) into our system. One of the key APIs we utilized was an identity confirmation API, which significantly bolstered the security of our app. This API ensured that all access to sensitive medical data was securely managed and that user identities were verified before they could interact with the system. Learning to implement such security measures was pivotal, not only for the integrity of MedixAI but also for complying with healthcare regulations concerning data protection and privacy.

8 Conclusion and Future Work

In conclusion, MediXAI provides effective solutions to existing system problems, such as high costs, time consumption, and low accuracy, which significantly impact both institutions and patients. By utilizing explainable machine learning models and optimizing the coordination between doctors and the models, our approach reduces costs by substituting one doctor in the current workflow and also maintains accuracy. The proposed solutions have been implemented for web and mobile platforms. For effective and healthy operation in Turkey Additional tuning should be done using Turkish data.

For future software enhancements, features such as video and voice calls between doctors and patients can be implemented. Also, informative content and self-check materials can be added. Additionally, a helper segmentation model that identifies calcifications and lesions in images could be developed for the 1. doctors' interface. This model would aid doctors for improving their diagnostic accuracy without the risk of misinterpretation.

In terms of machine learning advancements, to enhance model accuracy and facilitate its use in Turkey, acquiring a Turkish dataset is essential. This dataset should exclude any irrelevant components, such as visible parts of screening machines in the images. It has also been observed that the model often confuses between BI-RAD 4 and BI-RAD 5 categories. A potential solution could involve a two-stage classification process: initially distinguishing between three broad categories(normal, benign, malignant) and then applying a segmentation model to the malignant class to differentiate between BI-RAD 4 and 5 based on the shapes and sizes of lesions. To expand the dataset while maintaining privacy, federated learning techniques can be implemented in the future.

On the marketing side, For the marketing side, we have initiated discussions with the Tubitak Bigg program, considering a potential collaboration with them. We are planning to start our operations at Ufuk University Hospital, which will enable us to receive direct feedback and insights from the domain experts.Insights from discussions with doctors and industry professionals showed a significant demand for medical second opinions in Russia and Central Asia, suggesting an opportunity to launch this service through our application. By initially focusing on mammography, we can gradually expand our customer base.

9 Glossary

Term	Definition
XAI	Explainable Artificial Intelligence
HIPAA	Health Insurance Portability and Accountability Act
KETEM	Cancer Early Diagnosis, Screening and Education Center
BI-RADS	Breast Imaging-Reporting and Data System
API	Application Programming Interface
UI	User Interface
PBKDF	Password-Based Key Derivation Function

10 References

[1] Breast cancer facts and statistics 2023. Available at: https://www.breastcancer.org/facts-statistics (Accessed: 28 October 2023).

[2] Prof.Dr.M. (2018) Meme Kanseri belirtileri, Teşhisi Ve Tedavi yöntemleri: Anadolu Sağlık Merkezi, Anadolu Sağlık. Available at: https://www.anadolusaglik.org/blog/meme-kanseri (Accessed: 28 October 2023).

[3] Harbeck, N. *et al.* (2019) 'Breast cancer,' *Nature Reviews Disease Primers*, 5(1). doi:10.1038/s41572-019-0111-2.

[4] AI Chatbot - Amazon Lex - AWS. Available at: https://aws.amazon.com/lex/ (Accessed: 15 March 2024).

[5] DNS Service - Amazon Route 53 - AWS. Available at: https://aws.amazon.com/route53/ (Accessed: 15 March 2024).

[6] Bulk Cloud Email Service - Amazon Simple Email Service - AWS. Available at: https://aws.amazon.com/ses/ (Accessed: 15 March 2024).

[7] Cloud Object Storage - Amazon S3 - AWS. Available at: https://aws.amazon.com/s3/ (Accessed: 15 March 2024).

[8] What is Amazon EC2? - Amazon Elastic Compute Cloud. Available at: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html (Accessed: 15 March 2024).

[9] Load Balancer - Elastic Load Balancing (ELB) - AWS. Available at: https://aws.amazon.com/elasticloadbalancing/ (Accessed: 15 March 2024).

[10] What is Elastic Load Balancing? - Elastic Load Balancing. https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html (Accessed: 15 March 2024).

[11] Download Android Studio & App Tools, Android Developers. Available at: https://developer.android.com/studio.(Accessed: 12 November 2023).

[12] Visual Studio Code - Code Editing. Redefined, Microsoft. https://code.visualstudio.com/

[13] FlutterFlow - Build beautiful, modern apps incredibly fast, FlutterFlow. Available at: https://flutterflow.io/. (Accessed: 28 November 2023).

[14] Firebase console, Google. Available at: https://console.firebase.google.com. (Accessed: 02 December 2023).

[15] Google Analytics for Firebase, Google. Available at: https://firebase.google.com/docs/analytics. (Accessed: 02 December 2023).

[16] App Store, Apple. Available at: https://www.apple.com/app-store. (Accessed: 28 November 2023).

[17] Google Play, Google. Available at: https://play.google.com/. (Accessed: 28 November 2023).

[18] Git. Available at: https://git-scm.com/. (Accessed: 28 November 2023).

[19] Let's build from here, GitHub. Available at: https://github.com/. (Accessed: 28 November 2023).

[20] Jira, Issue & Project Tracking Software, Atlassian. Available at: https://www.atlassian.com/software/jira. (Accessed: 28 November 2023).

[21] Telegram Messenger. Available at: https://telegram.org/. (Accessed: 03 December 2023).