MEDILINK-ONE HEALTH ONE CARD

DSN4096-CAPSTONE PROJECT

Report

Submitted by

Aswathy Stalin B (20BAI10035)

in partial fulfillment for the award of the degree
of
BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING
(ARTIFICIAL INTELLIGENCE AND MACHINELEARNING)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

SEHORE, MADHYA PRADESH - 466114

MAY 2024

VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE
MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled **"MEDILINK-ONE HEALTH ONE CARD"** is the bonafide work of "**Aswathy Stalin B (20BAI10035),"** who carried out the project work (DSN4096- Capstone Project) under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR
Dr Pradeep Kumar Mishra,
Senior Assistant Professor (Gr-1),
School of Computing Science and Engineering,
VIT BHOPAL UNIVERSITY.

PROJECT GUIDE
Mr. Jitendra Parmar,
Assistant Professor Junior,
School of Computing Science and Engineering,
VIT BHOPAL UNIVERSITY.

The DSN4096-Capstone Project phase-II Viva Voce Examination is held on _____

ACKNOWLEDGEMENT

First and foremost, we would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

We would like to thank Mr. Jitendra Parmar, for continually guiding and actively participating in our project, and giving valuable suggestions to complete the project work.

We wish to express our heartfelt gratitude to Dr Pradeep Kumar Mishra, Head of the Department, School of Computing Science and Engineering for much of his valuable support and encouragement in carrying out this work.

We wish to express our heartfelt gratitude to Dr. S. Poonkuntran, Professor and Dean, School of Computing Science and Engineering for much of his valuable support and encouragement in carrying out this work.

We would like to thank all the technical and teaching staff of the School of Computing Science and Engineering, who extended directly or indirectly all support.

Last, but not least, we are deeply indebted to our parents who have been the greatest support while we worked day and night for the project to make it a success.

LIST OF FIGURES

ABSTRACT

MEDILINK, a comprehensive healthcare platform, addresses critical challenges prevalent in the healthcare sector. The platform confronts issues such as breaches in health records, human errors in diagnosis, and overcrowded appointment scheduling in hospitals. To overcome these obstacles, MEDILINK emphasizes robust data security protocols, harnesses the power of AI for accurate diagnosis, predicts diseases, and streamlines appointment management processes. By prioritizing these initiatives, MEDILINK aims to redefine healthcare delivery, improve patient outcomes, enhance operational efficiency, and elevate the overall healthcare experience for patients and healthcare providers alike.

TABLE OF CONTENTS

CHAPTER 1
PROJECT DESCRIPTION AND OUTLINE

1.1 INTRODUCTION

In healthcare, efficient access to medical services and streamlined management of patient information are paramount. However, traditional healthcare systems often face challenges such as breaches in patient records, errors in diagnosis, and cumbersome appointment scheduling processes. In response to these obstacles, this report proposes the development of a solution: the "Medilink" platform. Medilink aims to revolutionize healthcare delivery by offering a comprehensive online platform that addresses these challenges head-on. By analyzing the current shortcomings of healthcare systems and understanding the needs and expectations of both patients and healthcare providers, this report will outline a design and implementation plan for the Medilink platform. Through robust data security measures, AI-driven diagnostic capabilities, disease prediction algorithms, and streamlined appointment scheduling functionalities, Medilink seeks to enhance the efficiency and effectiveness of healthcare services. Moreover, this report will discuss the potential benefits of Medilink, including improved patient outcomes, enhanced operational efficiency for healthcare providers, and a more seamless healthcare experience for all stakeholders. Through ongoing evaluation and feedback mechanisms, the success of Medilink will be assessed, and recommendations for further development and expansion will be provided. Ultimately, the proposed Medilink platform endeavors to redefine the standards of healthcare delivery, ushering in an era of enhanced accessibility, efficiency, and quality of care for patients and healthcare providers alike.

1.2 MOTIVATION FOR WORK

The motivation behind this project stems from several critical issues plaguing the current healthcare system. Firstly, the persistent challenge of health records getting lost or misplaced underscores the necessity for a robust and centralized system to ensure the continuity and accuracy of patient care. Secondly, the frustration and inconvenience faced by patients due to long queues and cumbersome appointment booking processes highlight the urgent need for streamlined healthcare services. Thirdly, the potential for

human error in diagnosis poses significant risks to patient safety and underscores the importance of leveraging advanced technologies, such as artificial intelligence, to enhance diagnostic accuracy and efficiency. Lastly, the increasing commercialization of hospitals raises concerns about prioritizing profit over patient care, emphasizing the need for solutions that prioritize patient well-being and quality of care. By addressing these pressing issues, our project aims to improve patient experiences, optimize healthcare operations, and enhance the overall quality and accessibility of healthcare services.

1.3 PROBLEM STATEMENT

In the current healthcare ecosystem, the management of patient health records across various institutions remains fragmented, leading to inefficiencies in care coordination and potential risks to patient safety and privacy. Moreover, the reliance on traditional diagnostic methods introduces the possibility of human error and inconsistency in diagnosis. Integrating artificial intelligence (AI) into the diagnostic process holds promise for enhancing accuracy and efficiency by continuously learning from new data. However, the lack of a unified platform for managing health records and diagnostic data exacerbates these challenges. Additionally, the absence of streamlined appointment booking systems further compounds the issue, resulting in patient dissatisfaction and operational inefficiencies. Furthermore, the disparate databases maintained by different hospitals raise concerns about data integrity and security, with potential vulnerabilities to unauthorized alterations. Addressing these multifaceted challenges requires the development of a comprehensive solution that seamlessly integrates health records management, AI-based diagnosis, and appointment scheduling while ensuring data integrity, privacy, security, and mitigating issues of commercialization that may arise within the healthcare sector.

1.4 OBJECTIVE OF THE WORK

1. **Secure Health Records Management:** Implement a robust system for securely storing and managing patient health records, ensuring data integrity, confidentiality, and accessibility across healthcare institutions.

2. **Integration of AI for Early Discovery:**

   Integrate machine learning (ML) models to detect case records for early discovery of implicit health pitfalls or conditions, allowing for visionary healthcare interventions.

3. **AI-Powered Disease Detection:** Integrate advanced artificial intelligence (AI) algorithms to enhance diagnostic accuracy by detecting various diseases and tumors through automated analysis of medical data, including imaging scans and patient records.

4. **Streamlined Appointment Booking:** Design and implement an efficient appointment booking system to facilitate seamless scheduling and management of patient appointments, reducing wait times, and optimizing healthcare facility operations.

5. **Enhanced Patient-Doctor Communication:** Develop user-friendly interfaces and communication tools within the Medilink platform to bridge the gap between patients and healthcare providers, facilitating better communication, information sharing, and collaboration for improved healthcare delivery.

1.5 SUMMARY

The proposed project entails developing and launching MEDILINK, a website aimed at streamlining healthcare services for both doctors and patients. MEDILINK aims to innovate healthcare provision by offering a comprehensive online platform that addresses persistent industry challenges, such as overcrowding, lengthy wait times, and ineffective management of patient records and medical conditions. Through centralized record-keeping and AI-driven analytics, MEDILINK optimizes care delivery, facilitating data-driven decisions and personalized treatment strategies. MEDILINK strives to create a more efficient and secure healthcare environment, promoting preventive care, enhancing patient outcomes, and revolutionizing the overall healthcare experience for all involved parties.

CHAPTER 2
RELATED WORK INVESTIGATION

2.1 EXISTING APPROACHES/METHODS

1. **Electronic Health Records (EHR) Systems:** Companies like Epic Systems Corporation and Cerner Corporation provide advanced EHR solutions, such as Epic EHR and Cerner Millennium EHR, which consolidate patient health records across healthcare facilities. These platforms enable secure access to patient data for healthcare providers, facilitating seamless care coordination and continuity across diverse healthcare settings.
2. **Artificial Intelligence (AI) in Healthcare Diagnosis:** IBM's Watson Health platform employs AI algorithms to analyze medical data and aid healthcare professionals in diagnosis and treatment decision-making. For instance, Watson for Oncology delivers evidence-based treatment recommendations for cancer patients by analyzing patient data, medical literature, and expert guidelines, thereby ensuring personalized and efficacious care.
3. **Appointment Booking Platforms:** Platforms like Zocdoc offer online appointment booking services, allowing patients to schedule appointments with healthcare providers based on factors like specialty, location, and availability.

By leveraging and expanding upon these established methodologies and technologies, your project can develop a comprehensive solution that addresses challenges related to fragmented health records, enhances diagnostic precision through AI integration, and optimizes appointment booking processes to elevate patient care and satisfaction.

2.2 PROS AND CONS OF THE STATED APPROACHES/ METHODS

When it comes to the existing approaches/methods for healthcare solutions, there are several pros and cons to consider. Here are some of them:

Pros:

1. **Convenience:** Users can access healthcare services easily and conveniently through the platform, streamlining the process for both patients and healthcare providers.
2. **Timesaving:** The platform saves time by automating tasks such as appointment booking and medical record retrieval, allowing users to focus on patient care.
3. **Transparency:** Patients have access to their medical records and treatment plans, fostering transparency and trust between patients and healthcare providers.
4. **Report tracking:** The platform enables efficient tracking of medical reports and test

results, ensuring timely access to valuable information for informed decision-making.

5. **Reduced errors:** By leveraging AI-driven diagnosis and standardized processes, the platform minimizes errors in medical assessments and treatment plans.

6. **Integration with Medical Insurer:** Seamless integration with medical insurers facilitates hassle-free billing and insurance claim processing for patients and healthcare providers.
7. **Data Analytics:** The platform utilizes data analytics to derive insights from patient health records, enabling evidence-based decision-making and continuous improvement in healthcare delivery.

Cons:

1. **Technical issues:** Despite efforts to maintain stability, the website may experience occasional technical glitches or downtime, impacting user experience.
2. **Internet dependency:** Users rely heavily on internet connectivity to access and utilize the platform's services, potentially limiting accessibility in areas with poor connectivity.
3. **Learning curve:** Users may face a learning curve in navigating the platform's features and functionalities, requiring time and effort for adaptation.
4. **Security concerns:** Ensuring the security of patient’s personal information is paramount, with online platforms needing robust measures to safeguard against potential data breaches or unauthorized access.
5. It is important to assess these pros and cons based on the specific context of the healthcare solution and consider the needs and preferences of the patients and doctors when deciding on the approach/method for the healthcare platform.

CHAPTER-3
REQUIREMENT ARTIFACTS

3.1 INTRODUCTION

The requirements for our healthcare platform are paramount in ensuring its effectiveness, security, and user-friendliness. These requirements are essential to provide patients and healthcare providers with a seamless and reliable platform for managing appointments, accessing AI-powered diagnosis, and maintaining health records. The requirements for our healthcare platform can be categorized into three key areas: functional, non-functional, and usability requirements. Functional requirements delineate the features and capabilities of the platform, including appointment booking, AI-driven diagnosis, and health record management. Non-functional requirements encompass performance, security, and scalability considerations, ensuring the platform operates efficiently, securely, and can accommodate future growth. Usability requirements focus on enhancing the user experience, ensuring the platform is intuitive, accessible, and easy to navigate for both patients and healthcare providers. Adhering to these requirements is essential in creating a successful healthcare platform that improves patient care, enhances operational efficiency, and facilitates collaboration between patients and healthcare providers.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

- **Server:** A reliable server infrastructure is essential to host Medilink's website and efficiently handle incoming user requests, ensuring seamless accessibility to its services.
- **Storage:** The website requires a robust database system to securely store crucial user information, order histories, and menu data, facilitating efficient data management and retrieval.
- **Network:** A stable and dependable network connection is imperative to maintain quick response times and prevent downtime, guaranteeing uninterrupted access to the website's services.
- **Load Balancer:** In anticipation of potential traffic spikes, a load balancer may be necessary to evenly distribute incoming requests across multiple servers, optimizing

performance and preventing overload-induced disruptions.
These hardware requirements are crucial to uphold the website's performance, stability, and security, thereby ensuring a seamless and secure user experience. Additionally, scalability considerations are paramount to accommodate future growth and increased traffic.

Software Requirements:

- **Python, Flask, NumPy, Scikit-Learn, TensorFlow:** Leveraging a comprehensive software stack comprising Python-based frameworks and libraries such as Flask, NumPy, Scikit-Learn, and TensorFlow is fundamental in building robust AI-driven functionalities for Medilink's website, enabling advanced medical data analysis and decision-making support.

- **Programming Languages (JavaScript, CSS, HTML):** Incorporating essential web development languages like JavaScript, CSS, and HTML is imperative for crafting an intuitive and visually appealing user interface, ensuring seamless navigation and interaction for users across diverse devices and platforms.

These software requirements are essential for developing a feature-rich, secure, and user-friendly platform tailored to Medilink's healthcare solutions. By meeting these requirements, the website can deliver a seamless and secure experience for users while facilitating efficient data management and maintenance.

3.3 SPECIFIC PROJECT REQUIREMENTS

3.3.1 Data Requirements:

- The website must store users' profiles, including name, email, and delivery address.
- The website must secure patients' medical history that includes diagnosis given by doctors.
- The website has AI diagnosis in doctors screen to detect the diseases which eventually gets stored in medical history of patients.
- The website has doctors detail stored which helps the patient to choose the doctors according to their need.

3.3.2 Functionality Requirements:

- User Authentication and Account Creation: Enable users to create accounts and log in for accessing medical history and appointment booking.
- Doctor Selection Capability: Allow patients to choose preferred healthcare providers when scheduling appointments.
- Integration of AI-driven Disease Detection: Seamlessly integrate advanced AI for

accurate disease detection by healthcare providers.

- Secure Storage of Medical History: Ensure secure storage and management of patient medical records to prevent data breaches.

3.3.3 Performance Requirements:

- The website must be responsive and load quickly on the desktop.
- The website must be able to handle multiple concurrent users without slowing down.
- The website must be able to handle a high volume of appointments.

3.3.4 Security Requirements:

- The website must use secure authentication protocols to prevent unauthorized access to user accounts.
- The website must use measures to prevent SQL injection attacks and other common security threats.

3.3.5 Looks and Feel Requirements:

- The website must have a clean and modern design that is visually appealing.
- The website must be easy to navigate and use, with clear labels and instructions.

These requirements are critical to ensuring the functionality, reliability, and user-friendliness of our healthcare platform, while also upholding stringent standards for security and performance. By meeting these requirements, the platform can deliver a seamless and positive experience for both patients and healthcare providers, thereby bolstering our brand reputation and solidifying our position as a trusted provider in the healthcare industry.

3.4 SUMMARY

By carefully crafting and following the defined requirement artifacts, we guarantee that our healthcare platform, "MEDILINK," is precisely designed to address the complex needs of the healthcare industry. This meticulous strategy prioritizes delivering outstanding performance, robust security measures, and user-friendly interfaces, creating a smooth and rewarding healthcare experience for all stakeholders.

CHAPTER-4
DESIGN METHODOLOGY AND ITS NOVELTY

4.1 METHODOLOGY AND GOAL

The Medilink-One Health One Card project adopts an integrated methodology that combines Agile Development and User-Centered Design principles to create a groundbreaking healthcare solution. Agile methodologies facilitate iterative development, allowing for continuous adaptation to evolving requirements and user feedback, ensuring that the final product effectively meets user needs while remaining responsive to changes throughout the development process.

Aligned with the goals outlined in earlier chapters, the project's primary objectives are centered around centralizing patient records, integrating AI algorithms, enhancing communication within the healthcare ecosystem, and streamlining medical processes. Centralizing patient records enables easy access to comprehensive information, while AI-driven analysis aids in proactive disease detection, including early disease detection, kidney tumor detection, oral tumor detection, and glioma detection. Improved communication bridges gaps between patients and healthcare providers, fostering collaboration and informed decision-making. By streamlining medical processes, the project aims to optimize healthcare delivery and improve patient outcomes, while ensuring better data security.

Combining Agile Development and User-Centered Design methodologies allows the project to redefine healthcare delivery by promoting proactive interventions and enhancing patient care. The iterative nature of Agile Development ensures that the solution evolves to meet user needs effectively, while User-Centered Design principles prioritize the creation of an intuitive and user-friendly platform. This methodology ensures that the Medilink-One Health One Card solution is not only technologically advanced but also user-centric, contributing to the advancement of public health and patient care.

In summary, the methodology employed by the Medilink-One Health One Card project emphasizes flexibility, responsiveness, and user-centricity. By integrating Agile Development and User-Centered Design principles, the project aims to create an innovative healthcare solution that addresses the complex challenges of the healthcare landscape while prioritizing the needs and preferences of users.

4.2 FUNCTIONAL MODULES DESIGN AND ANALYSIS

Functional Modules for the MEDILINK-ONE HEALTH ONE CARD project include:

1. **User Authentication Module:**
- Enables patients and doctors to sign up and log in securely, ensuring authorized access to the platform.
2. **Doctor's Dashboard Module:**
- Provides doctors with a comprehensive dashboard displaying daily schedules and appointments, enhancing their schedule management efficiency.
3. **AI Modules for analyzing images:**
- Integrates Convolutional Neural Network (CNN) algorithms for analyzing patient records and medical images, including early disease detection, kidney tumor detection, oral tumor detection, and glioma detection.
4. **Prescription Generation Module:**
- Allows doctors to generate prescriptions within the application following patient consultations, with details securely stored in the database for reference.
5. **Patient's Dashboard Module:**
- Empowers patients to access a user-friendly dashboard displaying available doctors and schedule appointments conveniently.
6. **Appointment Scheduling Module:**
- Enables patients to schedule appointments with preferred doctors, enhancing accessibility and timely healthcare services.
7. **Prescription Access Module:**
- Provides patients with online access to their prescriptions, facilitating continuity of care and active management of health.
8. **System Initialization Module:**
- Initiates training for AI modules upon system startup, ensuring continuous optimization and accuracy in disease predictions.
9. **User Interaction Module:**
- Facilitates seamless interaction between patients and doctors, allowing for active management of healthcare journeys.

10. Logout Module:

- Allows users to securely log out after completing tasks within the application, ensuring data security and privacy.

4.3 SOFTWARE ARCHITECTURAL DESIGNS

The architecture of the Medilink-One Health One Card project encompasses a blend of foundational web development technologies and specialized frameworks, ensuring a robust and efficient system.

Client-Side:

- **HTML, CSS, JavaScript (JS):** These foundational technologies are utilized for building the user interface, providing structure, styling, and interactivity across diverse devices and platforms.
- **Material UI:** Integrated for crafting visually appealing and consistent UI elements, augmenting the user experience with its modern design principles.
- **State Management:** Managed efficiently using JavaScript lightweight libraries, ensuring consistent data handling and synchronization across various components.
- **Routing**: Implemented using server-side routing frameworks like Flask, facilitating seamless navigation between different views within the application.

Server-Side:

- **Flask**: Empowers the backend logic and API endpoints, enabling smooth communication between the client-side and the database.
- **RESTful API:** Offers standardized endpoints for client-server interaction, facilitating data retrieval and manipulation with ease.
- **SQL Database**: Stores critical data including patient records, prescriptions, and AI model training data securely, maintaining data integrity and accessibility.
- **Authentication**: Implemented using secure mechanisms provided by Flask, ensuring authorized access to system resources, and safeguarding sensitive information.

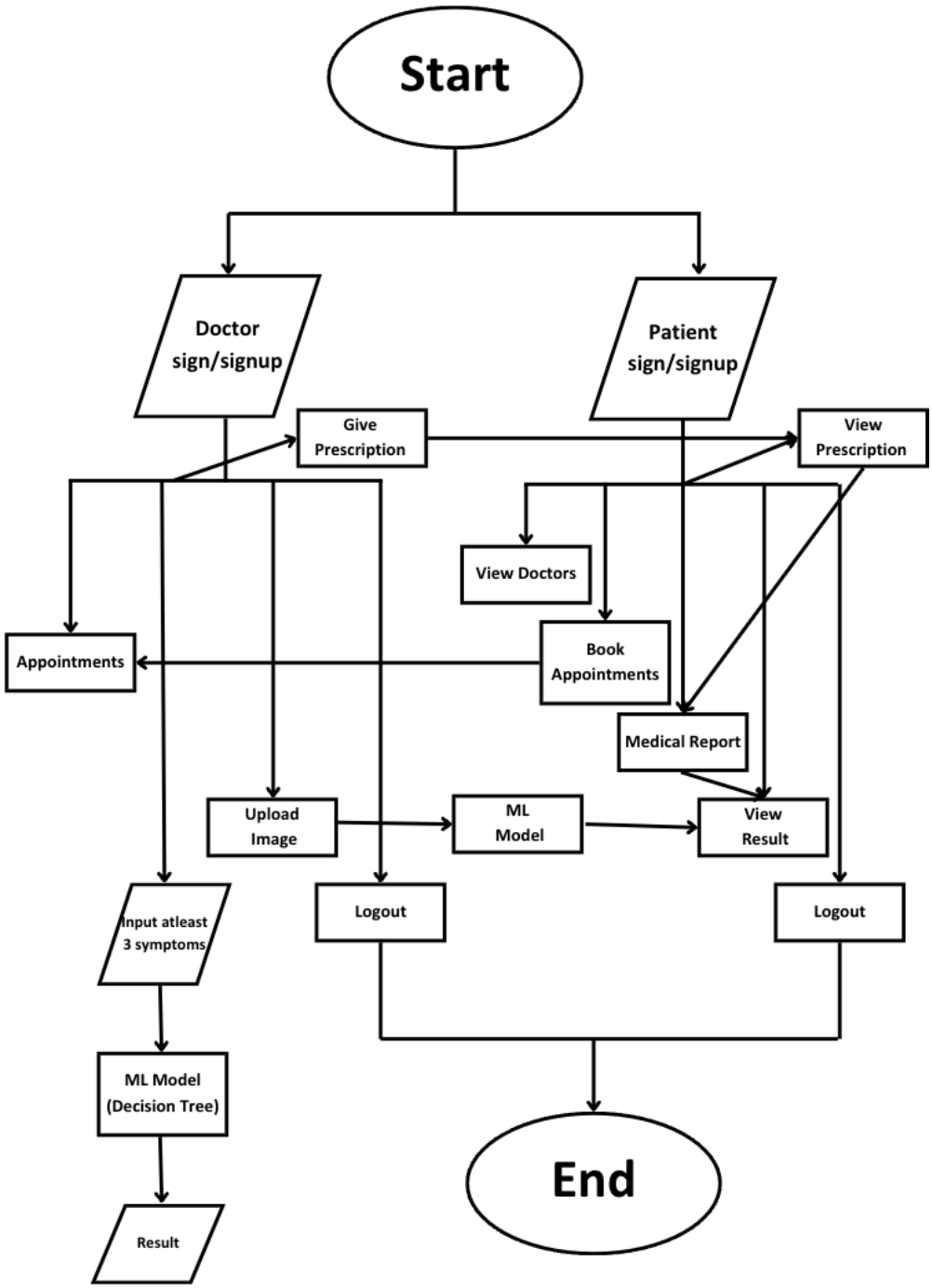


Figure 4.1: Architecture of Proposed Work

Development Tools:

- **Visual Studio Code (VS Code):** Serves as the primary code editor, providing developers with a lightweight and customizable environment for efficient development tasks.
- **Git**: Utilized as the version control system, enabling tracking of changes and seamless collaboration among team members, ensuring code integrity and consistency.
- **Jupyter Notebook:** Employed for data analysis, model training, and evaluation during AI development, offering an interactive and exploratory environment for enhanced productivity.

This architecture harmonizes traditional web development technologies with specialized frameworks and tools, ensuring scalability, maintainability, and performance of the Medilink-One Health One Card system. By leveraging this diverse set of technologies, the project lays a solid foundation for the advancement of healthcare technology, fostering innovation and efficiency in healthcare delivery.

4.4 USER-INTERFACE DESIGNS

In addition to the fundamental principles of simplicity, functionality, and aesthetics, the user interface design for MEDILINK-ONE HEALTH ONE CARD incorporates several key elements to enhance user experience and facilitate seamless interaction:

1. Intuitive Navigation: The user interface features intuitive navigation menus and buttons, strategically placed to guide users through the platform effortlessly. Clear and concise labeling ensures that users can easily find the desired features and functionalities without confusion.
2. Responsive Layouts: The platform adopts responsive design principles, ensuring optimal viewing and interaction experiences across various devices and screen sizes. Whether accessed on desktops, laptops, tablets, or smartphones, the interface adapts dynamically to provide a consistent and visually appealing layout.
3. Interactive Elements: Interactive elements such as buttons, dropdown menus, and sliders are implemented to enable user interaction and facilitate task completion. These elements are designed with user feedback in mind, providing visual cues and feedback to enhance usability and engagement.
4. Error Handling: Clear error messages and prompts are displayed when users encounter input validation errors or other issues. These messages are presented in a user-friendly manner, offering guidance on resolving the issue and continuing with the task at hand.

5. Consistent Branding: The interface maintains consistent branding elements, including color schemes, typography, and logo placement, to reinforce the platform's identity and establish visual continuity. This cohesive branding helps build trust and familiarity with users, enhancing their overall experience.

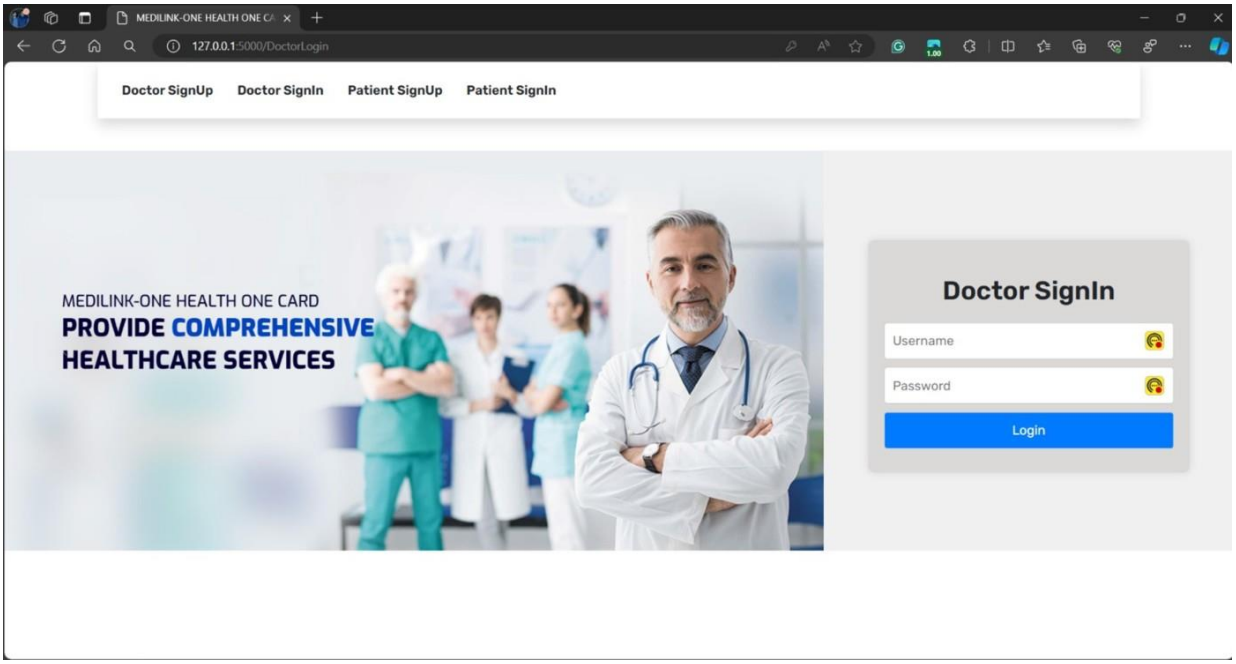


Figure 4.2: Doctor Sign in Module

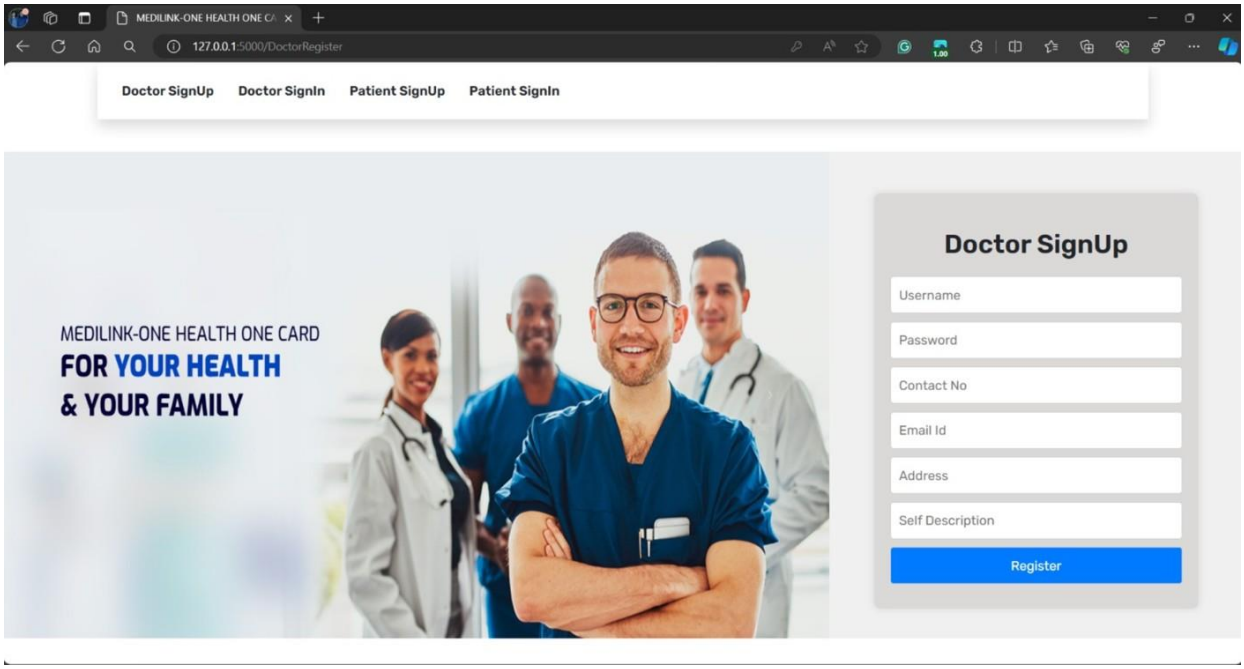


Figure4.3: Doctor Signup page

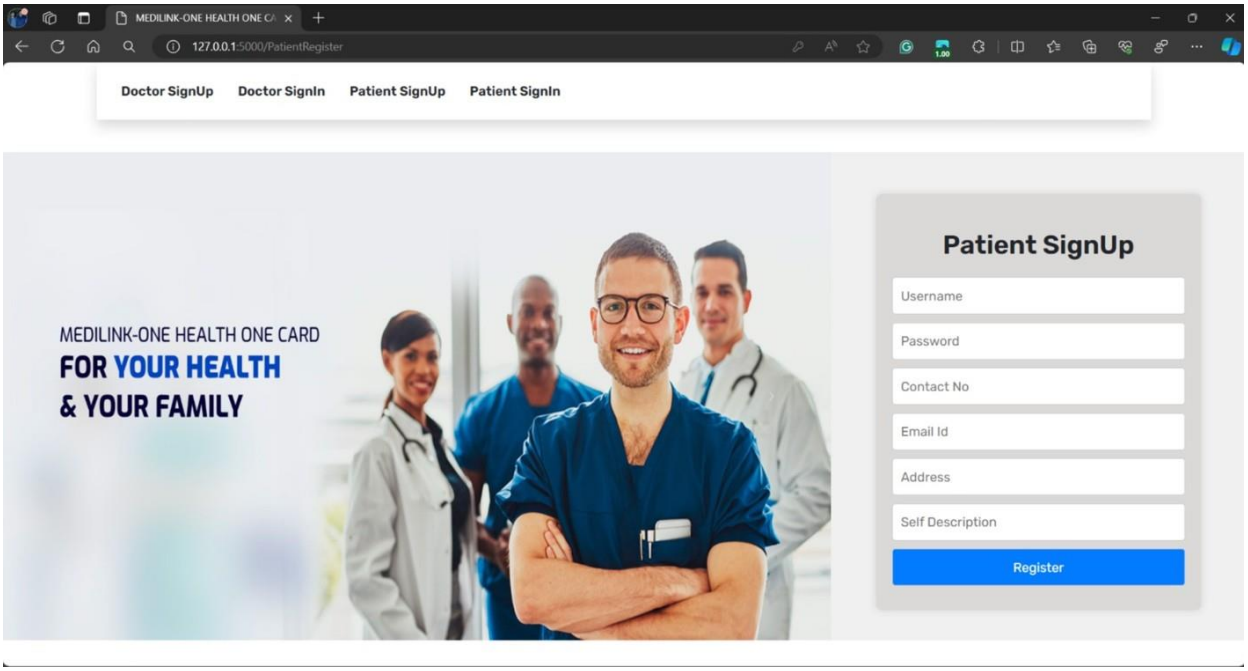


Figure 4.4: Patient Sign up Page.

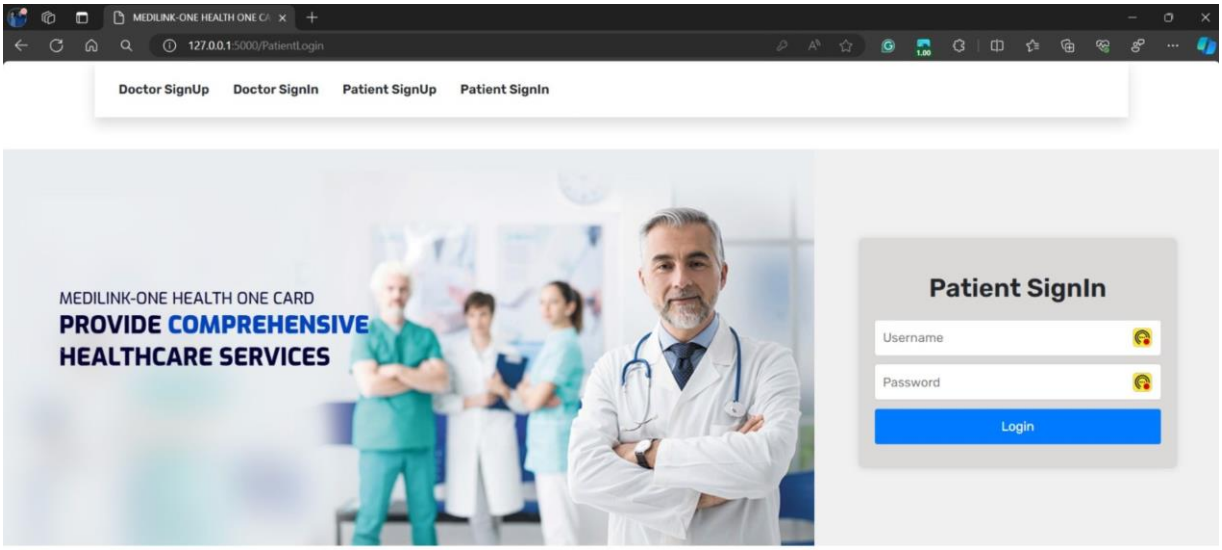


Figure 4.5: Patient Sign in Page

Overall, the user interface design for MEDILINK-ONE HEALTH ONE CARD is meticulously crafted to prioritize user experience, functionality, and accessibility. By incorporating intuitive navigation, responsive layouts, interactive elements, and visual feedback, the design aims to provide a seamless and engaging experience for users across all touchpoints of the platform.

4.5 SUMMARY

The Medilink-One Health One Card project represents a significant advancement in healthcare technology, integrating Agile Development and User-Centered Design principles to create an innovative solution. By blending these methodologies, the project aims to redefine healthcare delivery, emphasizing proactive interventions and enhanced patient care.

Aligned with the goals outlined in earlier chapters, the project's primary objectives include centralizing patient records, integrating AI algorithms for disease detection, enhancing communication within the healthcare ecosystem, and streamlining medical processes. Centralizing patient records enables easy access to comprehensive information, while AI-driven analysis aids in disease detection. Improved communication bridges gaps between patients and healthcare providers, fostering collaboration and informed decision-making. By streamlining medical processes, the project aims to optimize healthcare delivery and improve patient outcomes.

The software architecture of the Medilink-One Health One Card project comprises a blend of traditional web development technologies and lightweight frameworks. On the client-side, HTML, CSS, and JavaScript form the foundation for the user interface, supplemented by Material UI for visually appealing UI elements. Server-side logic is powered by Flask, with RESTful APIs facilitating communication with the database. Data storage is managed using a SQL database, ensuring secure storage of patient records and AI model training data. Development tools such as Visual Studio Code, Git, and Jupyter Notebook are utilized for efficient development and collaboration.

Overall, the Medilink-One Health One Card project aims to contribute to a more efficient and effective healthcare system, where disease detection, comprehensive record-keeping, and AI-driven insights lead to better patient outcomes and improved public health. By leveraging modern technologies and methodologies, the project lays a solid foundation for the advancement of healthcare technology, ensuring scalability, maintainability, and performance in the delivery of healthcare services.

CHAPTER-5
TECHNICAL IMPLEMENTATION & ANALYSIS

5.1 OUTLINE

I. Front-End Implementation

A. JavaScript

B. CSS

C. HTML

II. Back-End Implementation

A. Python, Flask, NumPy, Scikit-Learn, TensorFlow: Integrated a comprehensive software stack featuring Python-based frameworks and libraries, including Flask for web development.

B. Developed AI-driven functionalities: Leveraged Flask to build robust backend APIs capable of handling advanced medical data analysis.

C. Utilized NumPy and Scikit-Learn for data processing and machine learning tasks: Employed NumPy for efficient handling of data arrays and matrices, while leveraging Scikit-Learn for implementing machine learning algorithms.

D. Integrated TensorFlow for deep learning: Implemented TensorFlow to build and train deep learning models for tasks such as image recognition. Implementation

III. Testing and Validation

A. CNN

B. EXCEPTION

C. RESNET

5.2 TECHNICAL CODING AND CODE SOLUTIONS

1. **Setting up the Development Environment:**

- Install Python and Flask using pip: pip install Flask.
- Install required libraries such as NumPy (pip install numpy), scikit-learn (pip install scikit-learn), and TensorFlow (pip install tensorflow).
- Set up a virtual environment to manage dependencies.
- Install necessary front-end tools like Node.js and npm for package management.
- Choose a text editor or IDE suitable for Python and web development.

2. **Front-end Development:**

- Design the user interface using HTML for structure, CSS for styling, and JavaScript for interactivity.
- Utilize front-end frameworks like Bootstrap or Materialize for rapid development and responsive design.
- Implement features for patient registration, medical record display, prescription management, and interaction with the AI model using JavaScript.
- Ensure accessibility and usability by following UX/UI best practices.
- Test the front-end components across different browsers and devices.

3. **Back-end Development:**

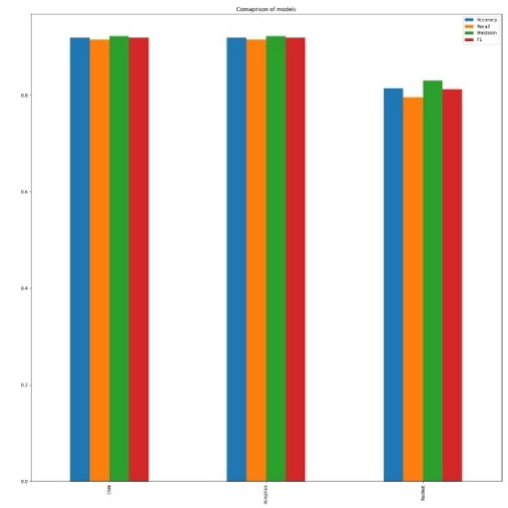
- Set up Flask project structure with routes, templates, static files, and configuration.
- Develop RESTful APIs using Flask to handle client-server communication.
- Integrate NumPy and scikit-learn for data processing and analysis in the back end.
- Implement server-side logic for user authentication, data validation, and error handling.
- Define endpoints for interacting with the TensorFlow model for disease detection, considering exception handling for model building.
- Test the back-end APIs using tools like Postman or curl.

4. **Database Management:**

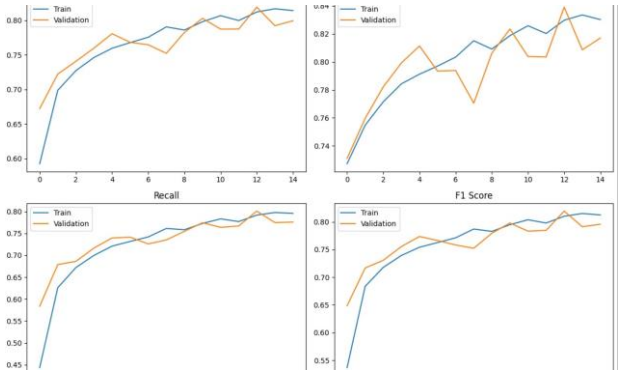
- Choose a database system compatible with Flask such as SQLite, PostgreSQL, or MySQL.
- Design the database schema to store patient records, prescriptions, medical history, and other relevant information.
- Use Flask-SQLAlchemy or Flask-MySQL to interact with the database from the Flask application.
- Implement CRUD operations for managing data persistence.
- Ensure data security and privacy by encrypting sensitive information and implementing proper access control.

5. **Deployment:**

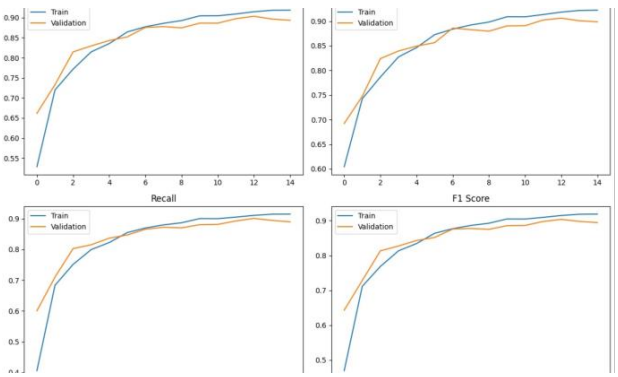
- Choose a deployment platform like Heroku, AWS Elastic Beanstalk, or DigitalOcean.
- Containerize the application using Docker for portability and consistency across environments.
- Set up CI/CD pipelines with tools like GitHub Actions or Jenkins for automated testing and deployment.
- Configure server environments for development, testing, and production stages.
- Monitor the deployed application for performance, security, and availability using logging and monitoring tools.



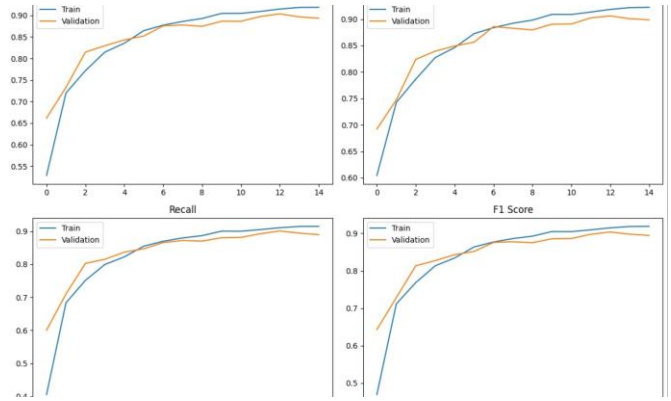
(a)



(b)



(c)



(d)

Figure 5.1: (a) Comparison between CNN, Xception, Resnet

(b) Resnet (c) Xception (d) CNN

5.3 PROTOTYPE SUBMISSION

Implemented patient and doctor login functionalities using HTML, CSS, and JavaScript in the frontend, ensuring secure access to the platform for both user types.

Enabled patients to schedule appointments by providing their symptoms, facilitating efficient management of healthcare appointments and patient-doctor interactions.

Integrated SQL for data storage, enabling robust and structured storage of patient health records, appointment details, and diagnostic information within the platform's database.

Leveraged AI algorithms enable doctors to scan Brain tumor, pituitary tumor, kidney tumor, oral tumor, breast tumor, fracture and disease prediction enhancing diagnostic accuracy and enabling timely treatment interventions for patients.

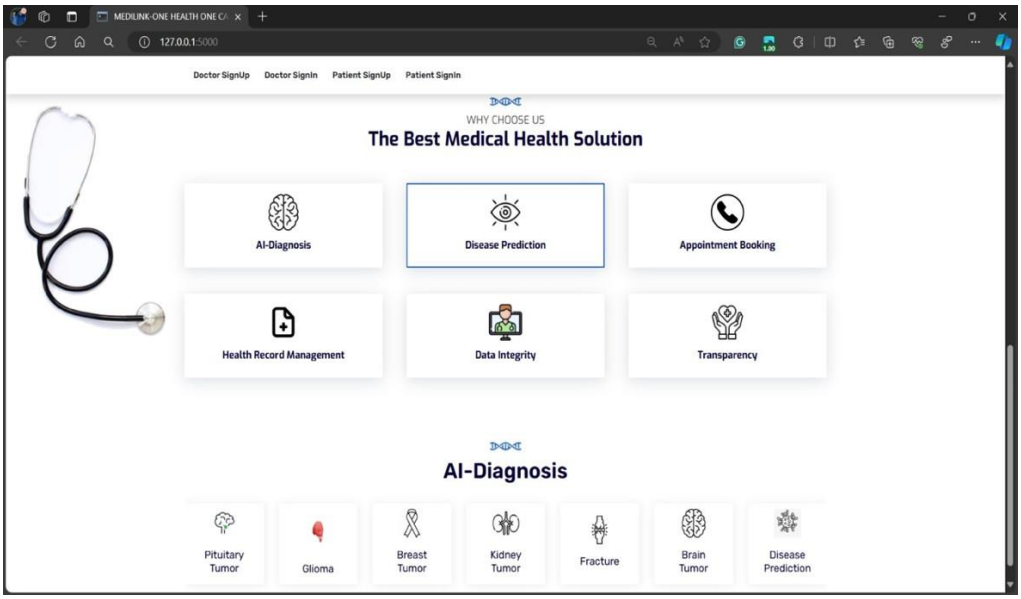


Figure 5.2: Home Page

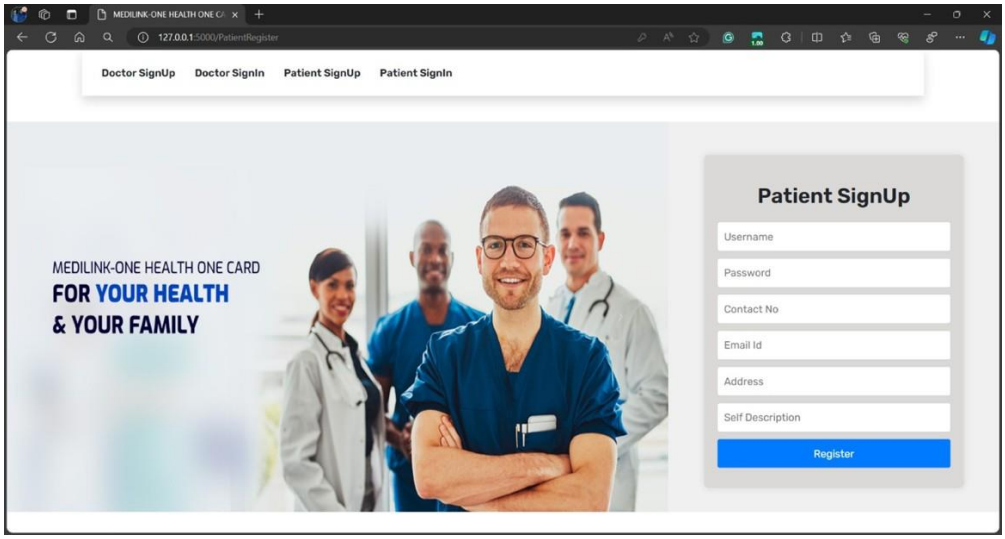


Figure 5.3: Patient Signup Page

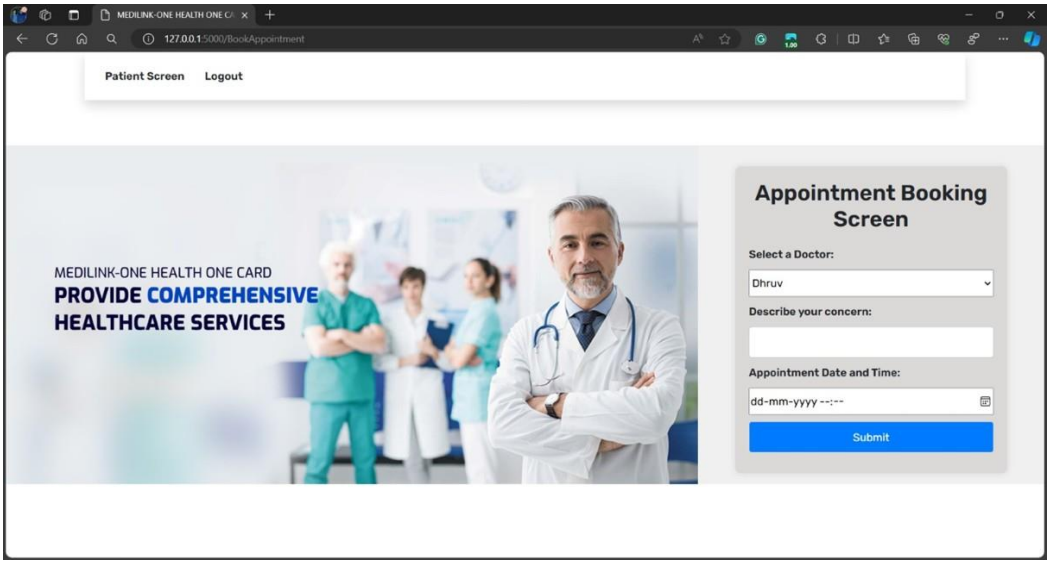


Figure 5.4: Appointment Booking Page

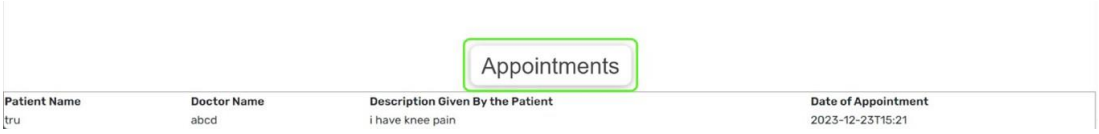


Figure 5.5: Appointments

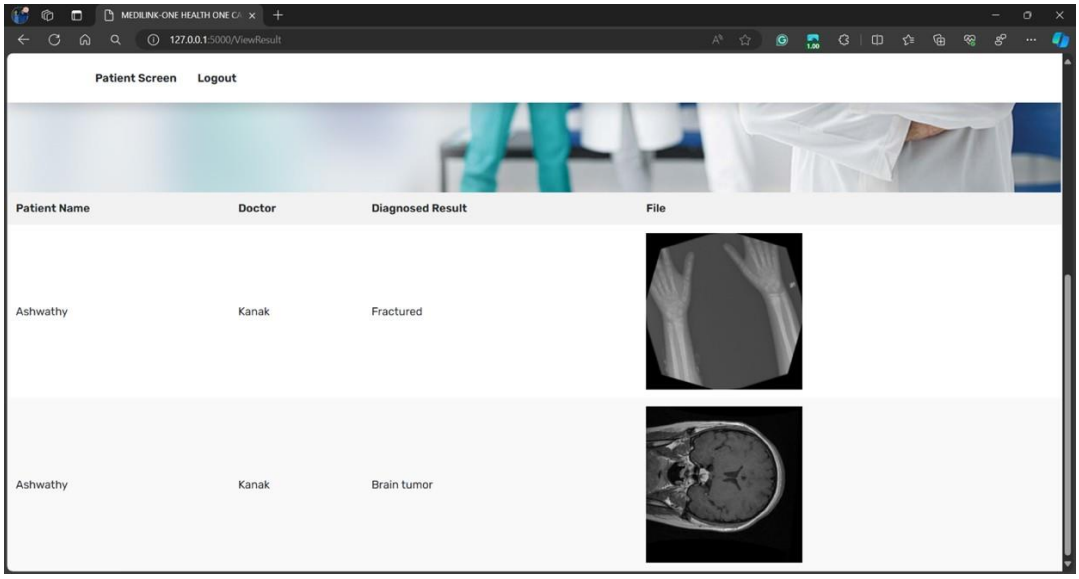


Figure 5.6: Medical Report

5.4 SUMMARY

The development of MEDILINK involved a comprehensive approach, utilizing a combination of front-end and back-end technologies to create a robust healthcare platform. Frontend development relied on HTML, CSS, and JavaScript to craft a visually appealing and interactive user interface. On the backend, Python served as the primary programming language, with Flask providing the framework for building APIs to handle medical data analysis. Advanced AI-driven functionalities were implemented using libraries such as NumPy, Scikit-Learn, and TensorFlow, enabling tasks like data processing, machine learning, and deep learning model training for tasks such as image recognition. Testing and validation were performed using Convolutional Neural Networks (CNN), RestNet and the Xception model, ensuring the accuracy and reliability of medical image analysis and diagnosis within the platform. This comprehensive approach resulted in the creation of MEDILINK, a sophisticated healthcare solution with advanced analytical capabilities and robust performance.

CHAPTER-6
PROJECT OUTCOME AND APPLICABILITY

6.1 KEY IMPLEMENTATION OUTLINES OF THE SYSTEM

- **User Management:**

A. Implement user registration and authentication using a secure framework like Flask and a robust database like SQLite to ensure user privacy and data security.
B. Create comprehensive user profiles for patients and healthcare providers.
C. Allow users to manage their personal information, preferences, and medical history (for patients) securely within the platform.

- **Medical Records Management:**

A. Establish a secure and centralized database to store patient medical records efficiently.
B. Only authorized healthcare providers with the appropriate permissions should be able to access relevant medical data.
C. Design functionalities for secure archiving, efficient retrieval, and controlled sharing of medical records with authorized personnel across healthcare institutions.

- **Prescription Management:**

A. Enable doctors to upload prescriptions securely within the patient's medical record.

B. Allow both patients and doctors to view current and past prescriptions easily within the platform. This facilitates continuity of care and informed decision-making.

C. Ensure prescriptions are retrievable from anywhere, empowering patients to access their medication history and reducing the risk of missing refills.

- **Appointment Scheduling and Management:**

A. Develop a user-friendly online appointment scheduling system that allows patients to easily search for available appointments, book appointments with preferred providers, and manage their appointments conveniently.
B. Provide healthcare providers with tools to view, manage, and update appointments efficiently. This may include functionalities like viewing appointments and managing appointment notes.

- **Symptom-Based Disease Prediction:**

A. Allow doctors to enter up to five symptoms patients are experiencing.

B. Integrate a machine learning model trained to analyze symptom combinations.

C. Based on the entered symptoms (minimum of 3 required), the model will predict potential disease matches with a confidence score. This empowers doctors to seek professional medical advice with a starting point for discussion.

- **AI-powered Medical Image Analysis**

A. Enable doctors to securely upload scanned medical images (e.g., MRI, X-ray) for analysis.

B. Integrate a separate machine learning model trained on medical image datasets.

C. The model will analyze uploaded images to identify potential abnormalities or signs of disease.

D. The system will provide a risk indication based on the analysis, highlighting the need for further evaluation by a qualified healthcare professional. This serves as a preliminary screening tool, not a definitive diagnosis.

By integrating these functionalities, Medilink seeks to enhance patient experiences, optimize healthcare operations, and improve the overall quality and accessibility of healthcare services.

6.2 SIGNIFICANT PROJECT OUTCOMES

Medilink goes beyond the typical functionalities of a healthcare platform, offering a comprehensive suite of features designed to revolutionize healthcare delivery. Here is a glimpse into the significant outcomes Medilink aims to achieve:

- **Increased Efficiency:** Medilink streamlines healthcare processes, reducing time and effort for both patients and providers. Online appointment scheduling eliminates wait times and simplifies booking, while centralized medical records minimize administrative tasks.

- **Improved Accuracy:** By leveraging electronic health records and AI-powered diagnostics, Medilink reduces the potential for human error in diagnoses and data management. This leads to more accurate assessments and treatment plans, improving patient outcomes.

- **Unified Patient Records:** Medilink establishes a secure and centralized database to eliminate the risk of lost or misplaced records. This ensures seamless information sharing and continuity of care across healthcare institutions. With a complete medical history readily available, providers can make informed decisions for optimal treatment plans.

- **Empowered Patients:** Medilink empowers patients to take charge of their health. The platform allows them to securely manage their medical profiles, access medical history, and view prescriptions.

- **Effortless Appointment Scheduling:** Medilink streamlines appointment booking with its user-friendly online system. Patients can conveniently search for available appointments with preferred providers, book appointments, and manage them easily. Healthcare providers benefit from efficient appointment management tools that reduce

wait times and optimize clinic operations.

- **AI-powered Diagnostics:** Medilink goes beyond traditional diagnostics by integrating machine learning models. These AI tools analyze medical data (imaging scans, patient records) to assist in early disease detection, improve diagnostic accuracy, and recommend potential treatment plans. This empowers healthcare professionals to deliver better patient care and outcomes.

- **Reduced Costs:** By streamlining processes and improving operational efficiency, Medilink has the potential to reduce healthcare costs. This can benefit both patients and healthcare institutions by minimizing unnecessary tests and optimizing resource allocation.

Medilink strives to be more than just a platform; it aspires to be a catalyst for positive change in the healthcare landscape. By offering a comprehensive suite of features and functionalities, Medilink aims to improve patient outcomes, enhance operational efficiency, and empower both patients and healthcare providers.

6.3 PROJECT APPLICABILITY ON REAL-WORLD APPLICATIONS

While Medilink excels at its core functionalities – establishing a centralized system for secure medical records and leveraging AI for improved diagnostics – its potential extends far beyond these initial applications. Here's a glimpse into how Medilink can be implemented in various real-world settings to create a more comprehensive and accessible healthcare ecosystem:

- **Healthcare Industry:** Medilink fosters a more collaborative environment for healthcare providers. The platform's centralized health record system ensures secure access to patient data across different facilities, enabling comprehensive care coordination and informed decision-making. Features like secure messaging and appointment scheduling streamline communication and improve patient flow, leading to better quality care.

- **Diagnostic Centre:** Medilink leverages artificial intelligence to improve diagnostic accuracy and treatment effectiveness. AI-powered tools like symptom-based prediction and medical image analysis can assist healthcare professionals in identifying potential illnesses and abnormalities, prompting further investigation, and potentially leading to earlier diagnoses and more targeted treatment plans.

- **Healthcare Administration:** Medilink streamlines administrative processes, leading to cost savings and improved operational efficiency. The user-friendly appointment booking system reduces wait times and optimizes resource allocation for healthcare facilities. Secure electronic health records eliminate the need for paper-based systems, minimizing administrative burdens. Data-driven insights can inform better decision-making regarding resource allocation and service delivery.

- **Research and Development:** Medilink's anonymized data collection offers valuable insights for medical research and development. By analyzing trends and patterns in patient data, researchers can gain crucial knowledge for developing new diagnostic

tools, treatments, and preventative measures. The platform can facilitate secure recruitment and data collection for clinical trials, accelerating medical advancements.

- **Public Health Initiatives:** Medilink can transform public health efforts. Real-time data collection and analysis capabilities allow public health agencies to effectively monitor disease outbreaks and identify areas requiring immediate intervention. The platform can be used to design and implement targeted outreach programs based on real-world data insights, empowering individuals to take an active role in safeguarding their health.

- **Chronic Disease Management Centres:** Medilink leverages machine learning to detect chronic illnesses earlier. This allows doctors to intervene sooner, preventing complications and improving treatment effectiveness. By identifying at-risk patients and tailoring preventative measures, Medilink helps reduce the overall risk of chronic diseases, not delay treatment.

These are just a few examples of how the core concepts behind Medilink- One Health One Card can be adapted and applied in various real-world healthcare settings. By fostering collaboration, optimizing workflows, and empowering patients and providers, this project can create a more accessible, efficient, and data-driven healthcare ecosystem globally.

6.4 INFERENCE

Based on its comprehensive functionalities, innovative applications of technology, and potential benefits for various stakeholders, Medilink presents a compelling vision for the future of healthcare. The platform addresses a critical need for secure and centralized management of patient medical records, empowering collaboration among healthcare providers and fostering a more coordinated care experience.

Leveraging AI technologies, Medilink goes beyond traditional record-keeping. Symptom-based prediction and medical image analysis tools assist healthcare professionals in earlier diagnoses and potentially more effective treatment plans. Additionally, the platform streamlines administrative processes through user-friendly appointment scheduling, leading to improved operational efficiency for healthcare facilities.

The platform extends its reach beyond the walls of hospitals, empowering patients through educational resources and fostering preventative healthcare measures. Ultimately, Medilink's adaptability and focus on preventative measures have the potential to revolutionize healthcare delivery, promoting cost-effectiveness and a more patient-centered approach for everyone.

CHAPTER-7

CONCLUSIONS AND RECOMMENDATION

7.1 OUTLINE

Medilink- One Health One Card project, takes patient data to the forefront, fundamentally transforming healthcare delivery. The key outcomes and achievements of this project include:

- Medilink establishes a centralized database for patient medical records. This eliminates the challenges of scattered healthcare information, enabling seamless access and analysis across different healthcare facilities. This fosters collaboration among providers and ensures a complete picture of a patient's health history for informed decision-making.

- The platform leverages artificial intelligence to revolutionize preventive healthcare. AI models analyze patient data to identify patterns and potential risks for diseases. This allows for early disease detection, enabling proactive interventions before illnesses progress. Additionally, AI can identify individuals with high health risks, allowing healthcare providers to prioritize care and implement preventative measures.

- Medilink fosters a more collaborative patient-doctor relationship. With a shared electronic health record readily accessible to both parties, patients can actively participate in their healthcare decisions. Doctors can gain a deeper understanding of a patient's medical history, leading to more personalized treatment plans and improved communication.

- Medilink provide a secure platform for managing prescriptions. Patients can view their current medications, track refills, and receive electronic prescriptions directly from their doctors within the platform. This eliminates the need for paper prescriptions and improves communication regarding medication management.

- Medilink prioritizes a user-friendly interface designed specifically for patients. This ensures clear navigation, easy access to information, and a seamless experience when managing personal health data or scheduling appointments.

Throughout the course of this project, we learned several important lessons, including:

- Planning and organization are key to project success. We found that breaking down the project into manageable tasks and setting clear timelines helped us to stay on track and meet our goals.

- Communication is critical. Effective communication among team members ensured that everyone was on the same page and that issues were resolved quickly. Open communication and a shared vision are crucial for successful implementation and user adoption.

- Prioritizing a user-friendly interface that caters to diverse needs is critical. Medilink's design should ensure intuitive navigation and clear communication for both patients and healthcare professionals.

- Thorough testing throughout the development process is essential to guarantee Medilink's functionality, reliability, and security. This ensures a positive user experience and minimizes potential disruptions in healthcare delivery.

While Medilink has demonstrably achieved significant milestones, there's always room for improvement. Here are some key areas for further development:

- Focus on continuously improving the accuracy and effectiveness of AI models for disease prediction, risk stratification, and treatment recommendations.

- Ensure Medilink can exchange data securely with other healthcare information systems, fostering a more connected healthcare ecosystem.

- Develop a sustainable development model to ensure ongoing maintenance, updates, and integration of modern technologies as healthcare needs evolve.

- Need a scalable infrastructure to accommodate the ever-growing volume of patient data and future system enhancements.

By focusing on these areas, Medilink can continuously improve its functionalities, solidify its position as a transformative force in healthcare delivery, and empower patients, healthcare providers, and public health agencies to achieve even better health outcomes.

7.2 LIMITATIONS/ CONSTRAINTS OF THE SYSTEM

While Medilink offers a compelling vision for healthcare transformation, it is crucial to acknowledge and address potential challenges to ensure its success:

- **Data Breach Risks:** Medilink's centralized data storage system concentrates a vast amount of sensitive patient information in one place. This makes it a prime target for hackers seeking to hack personal data or disrupt healthcare operations. Mitigating this risk requires robust security measures like regular security audits, encryption of sensitive data, and multi-factor authentication.

- **Data Ownership and Access Control:** Clear and transparent data ownership policies are essential to establish trust with users. Patients should have complete control over their data, with the right to access, modify, or delete it as needed. Furthermore, Medilink should implement granular access permission controls. This ensures that only authorized healthcare providers, such as doctors or specialists directly involved in a patient's care, can access relevant data based on the principle of least privilege.

- **Seamless Integration with Existing Systems:** Medilink's success hinges on seamless integration with existing hospital information systems (HIS). This eliminates data silos, reduces manual data entry, and ensures a smooth workflow for healthcare professionals.

Standardized data formats and open APIs are crucial for facilitating this integration.

- **Server Downtime:** Since Medilink relies heavily on server access, unexpected server outages or maintenance can disrupt critical services. This can prevent users from accessing valuable information like prescriptions. Mitigating this risk requires robust server management practices, including redundancy plans and disaster recovery protocols to minimize downtime and ensure service continuity.
- **Reliance on Connectivity:** A stable internet connection is crucial for using Medilink. Without it, tasks like appointment booking, AI diagnosis, and prescription details access become unavailable.
- **Server Dependence:** Medilink's core functionality hinges on server access. Server outages, maintenance, or unforeseen management issues can disrupt services.
- **Potential for Error:** Errors can still creep in due to factors like incorrect data entry or limitations in how AI understands complex medical situations.

By proactively addressing these challenges, Medilink can build a secure, ethical, and sustainable platform that fosters trust among patients and healthcare providers. This empowers all stakeholders to leverage the power of data for improved healthcare delivery and, better health outcomes.

7.3 FUTURE ENHANCEMENTS

Building upon its core functionalities, Medilink can reach even greater heights through continuous development and integration of innovative features. Here is a glimpse into some potential future enhancements:

- **Mobile App Development:** A mobile application would provide patients with on-the-go access to Medilink's features. They could easily manage appointments, access health records, refill prescriptions, and even connect with healthcare providers remotely.
- **Gamification and Reward Systems:** Implementing gamified elements or loyalty programs can encourage positive health behaviors and incentivize patients to utilize Medilink's preventative care functionalities.
- **Advanced Analytics and Reporting:** By leveraging AI and big data analytics, Medilink can generate comprehensive reports and insights. This allows healthcare institutions to identify trends, optimize resource allocation, and improve operational efficiency.
- **Telehealth Integration:** Seamless integration with telehealth platforms can expand access to care, especially in geographically dispersed areas. Patients can have virtual consultations with healthcare providers remotely, reducing travel time and increasing convenience.
- **Clinical Decision Support Tools:** AI-powered clinical decision support tools can provide real-time recommendations to healthcare professionals at the point of care. This can improve diagnosis accuracy, personalize treatment plans, and lead to better patient outcomes.

- **Multilingual Support:** Providing support for multiple languages can cater to a wider patient population and improve accessibility for international patients or those with limited language proficiency.

- **Integration with Public Health Agencies:** Real-time data sharing with public health agencies allows for better disease surveillance, outbreak detection, and implementation of preventative measures to safeguard population health.

- **Open APIs and Interoperability Standards:** Adopting open APIs and adhering to interoperability standards allows Medilink to connect seamlessly with various healthcare information systems, fostering a more connected healthcare ecosystem.

  By prioritizing these future enhancements, Medilink can solidify its position as a transformative force in healthcare delivery. It can empower patients, optimize healthcare operations, and contribute to improved health outcomes for all.

7.4 INFERENCE

Medilink presents a compelling vision for the future of healthcare. By establishing a centralized patient data platform and leveraging AI capabilities, Medilink empowers patients, optimizes healthcare delivery, and fuels advancements in public health research. Project outcomes and achievements have been significant, demonstrably improving patient care coordination, fostering preventative measures.

Looking ahead, continuous improvement remains paramount. Future enhancements like mobile app development, advanced analytics, and telehealth integration can further enhance functionality and accessibility. By prioritizing user-centric design, ethical AI integration, and robust data security measures, Medilink can build trust and achieve long-term sustainability.

Ultimately, Medilink embodies the potential of technology to revolutionize healthcare. It paves the way for a future where patients are empowered, healthcare providers are equipped with powerful tools, and public health initiatives benefit from real-time data insights. This project underscores the importance of continuous development and adaptation to meet the evolving needs of stakeholders within the healthcare ecosystem. Medilink stands as a testament to the power of technology in improving healthcare delivery and enhancing the well-being of individuals and community.

Appendix A (Screenshots)

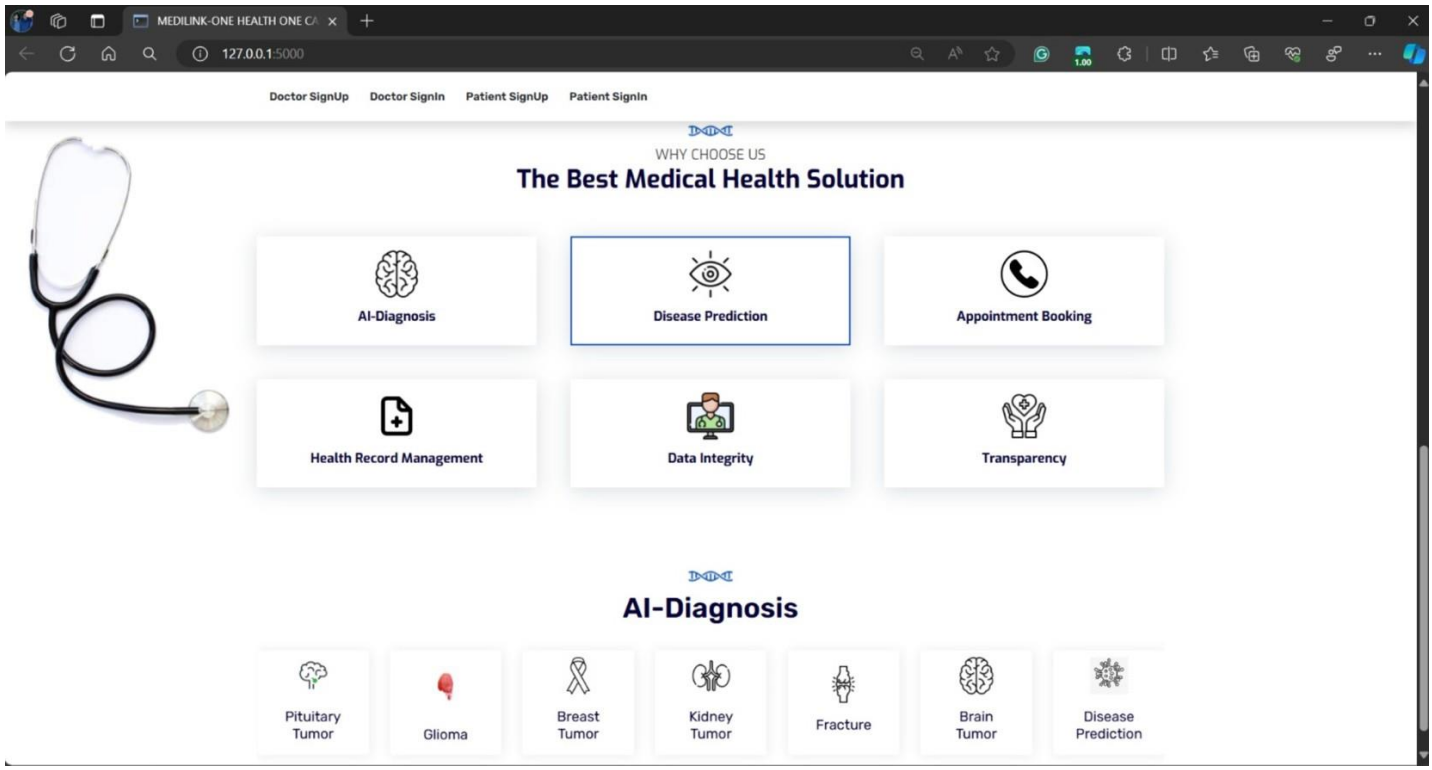
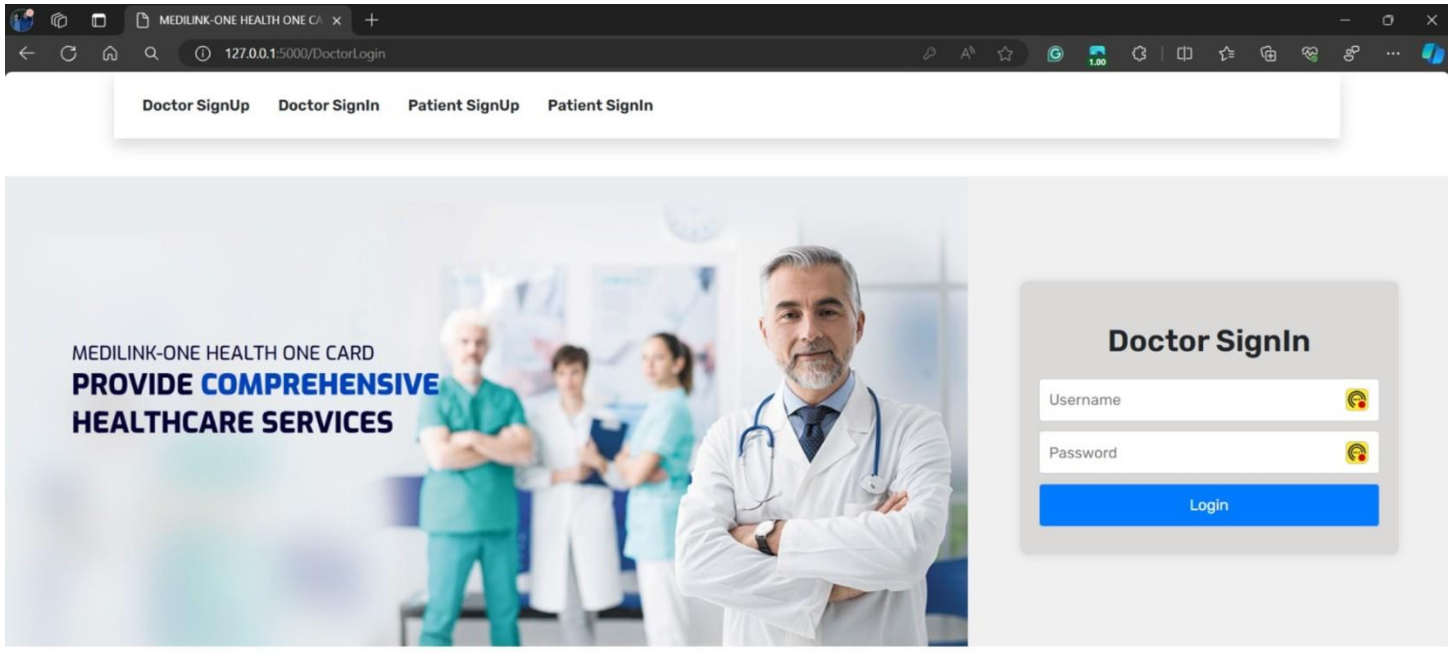


Figure A.1: Home Page



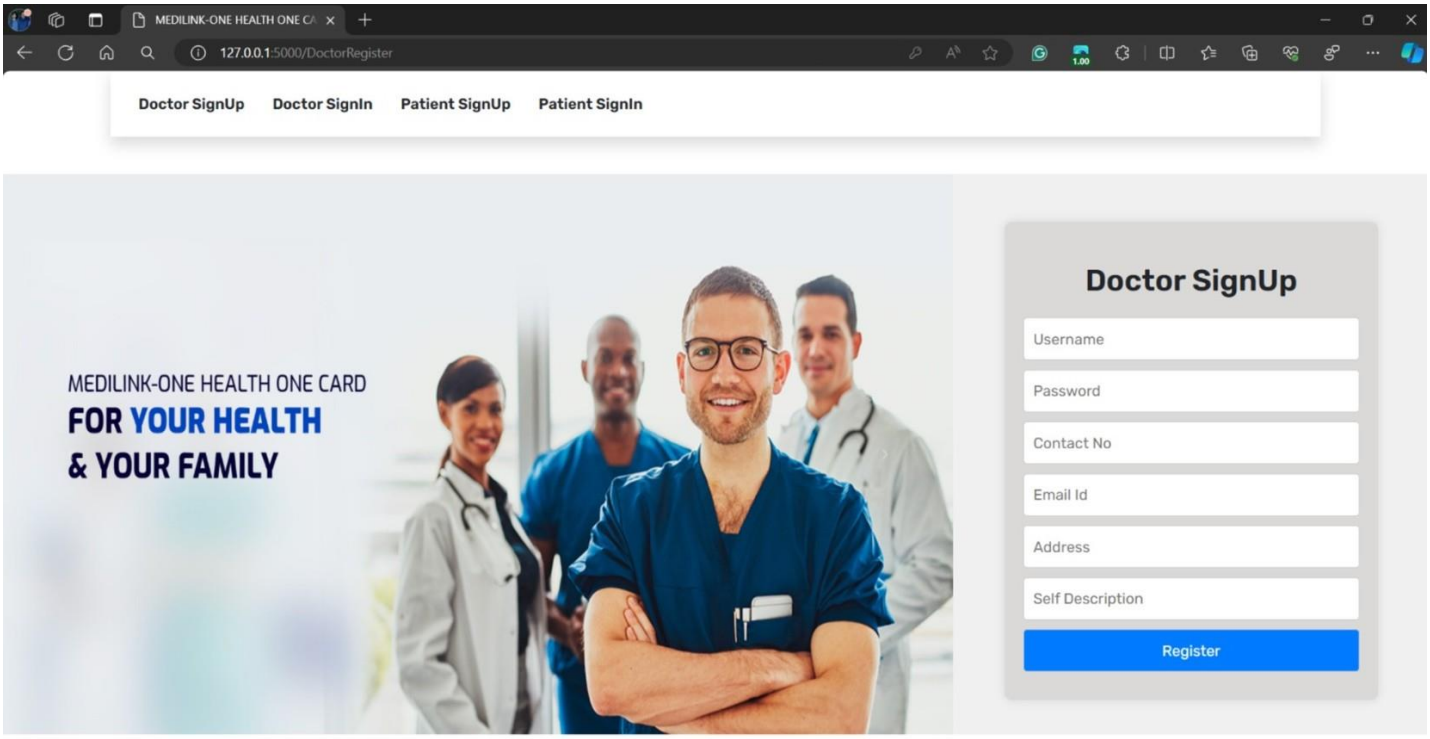
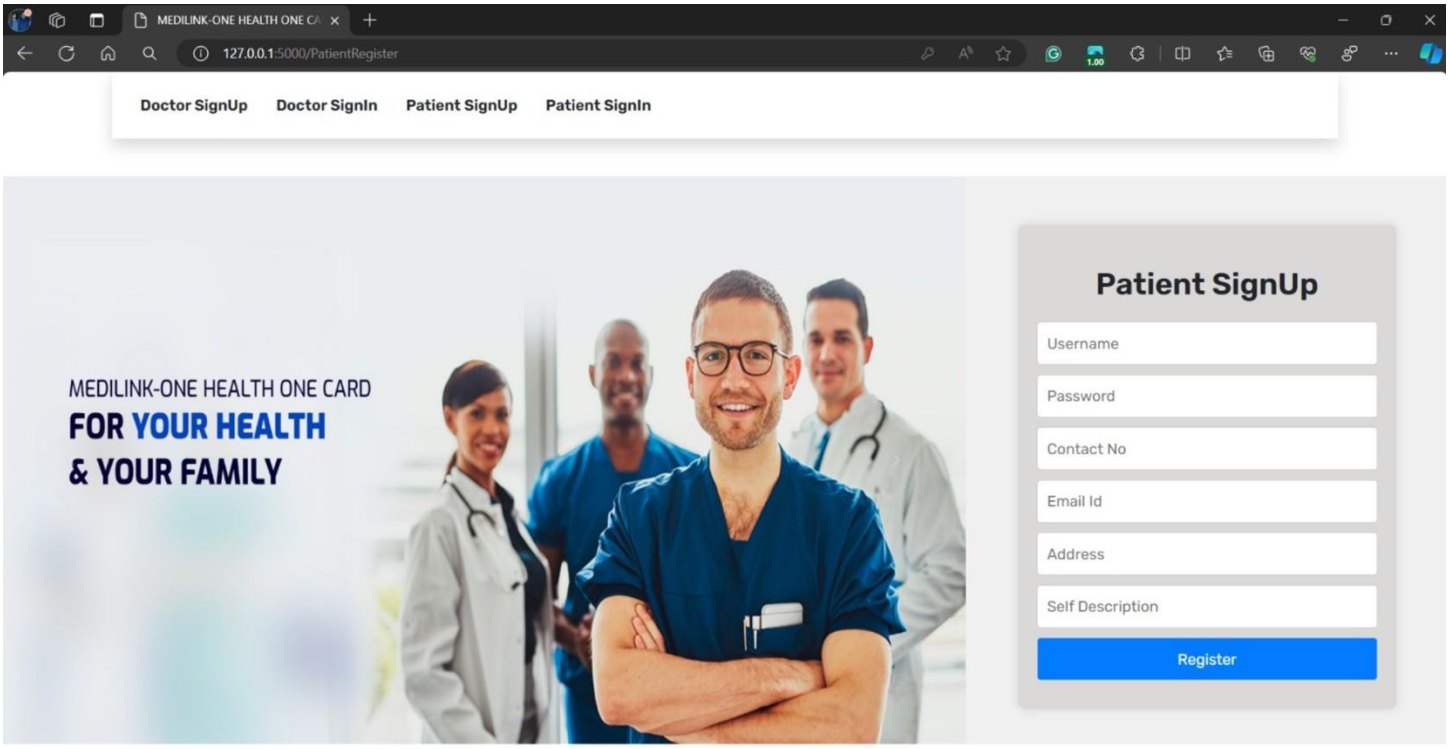
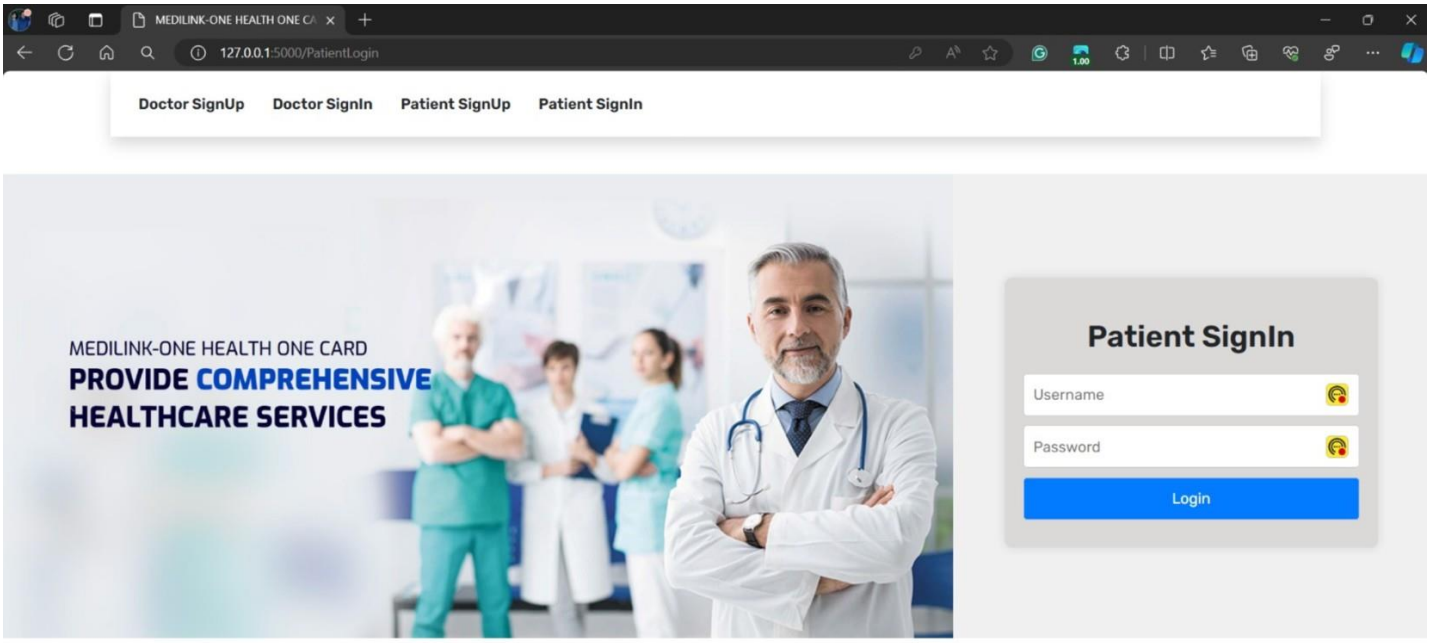
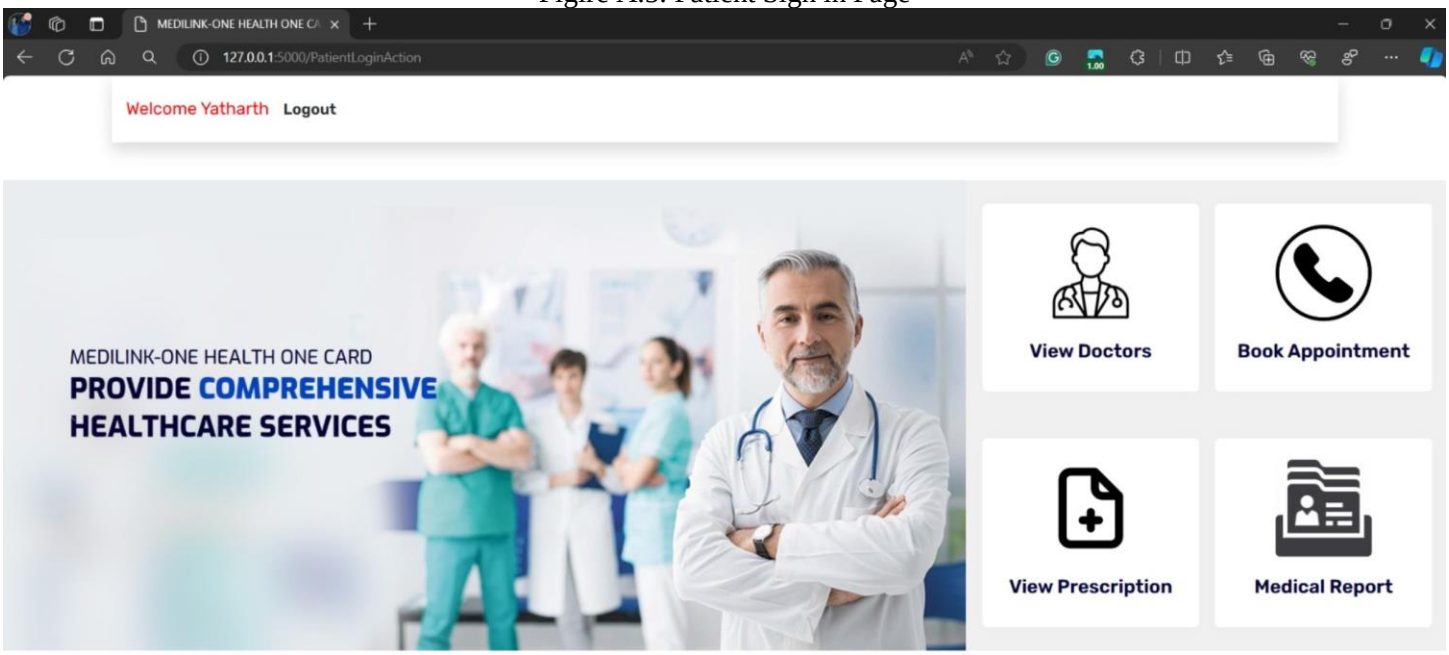


Figure A.3: Doctor Sign up Page





Figire A.5: Patient Sign in Page



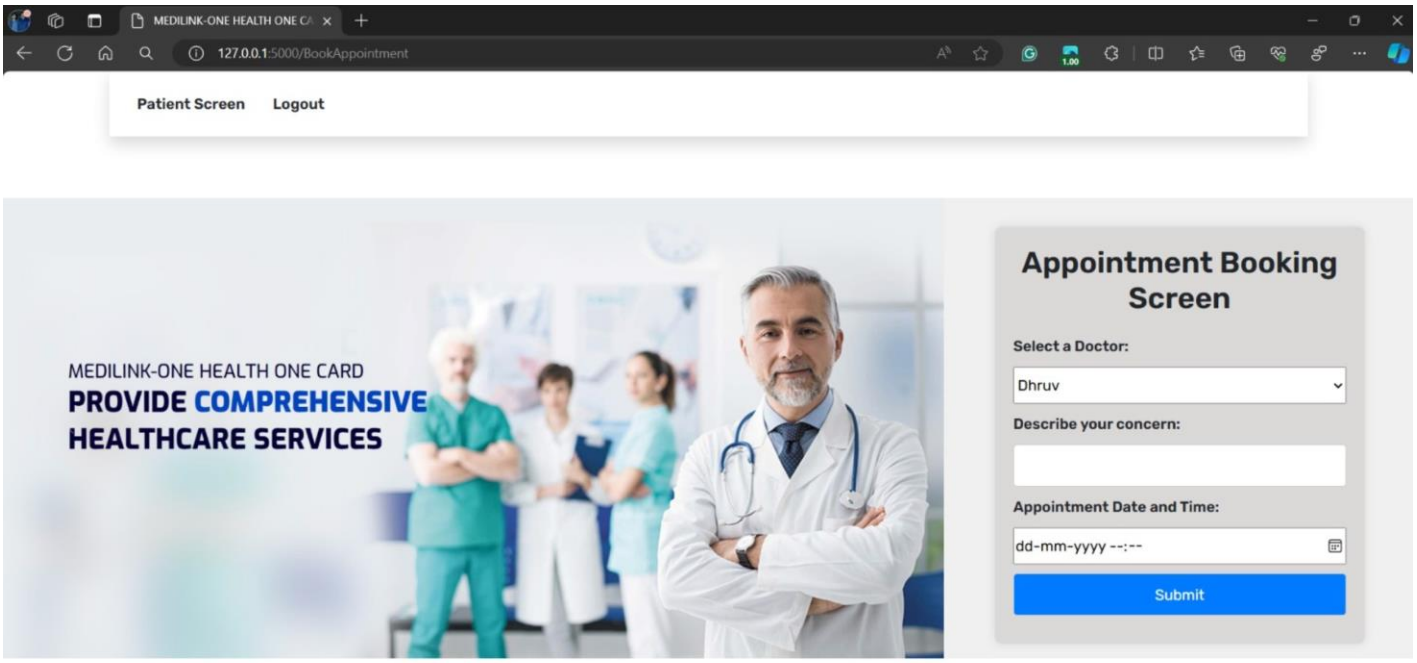
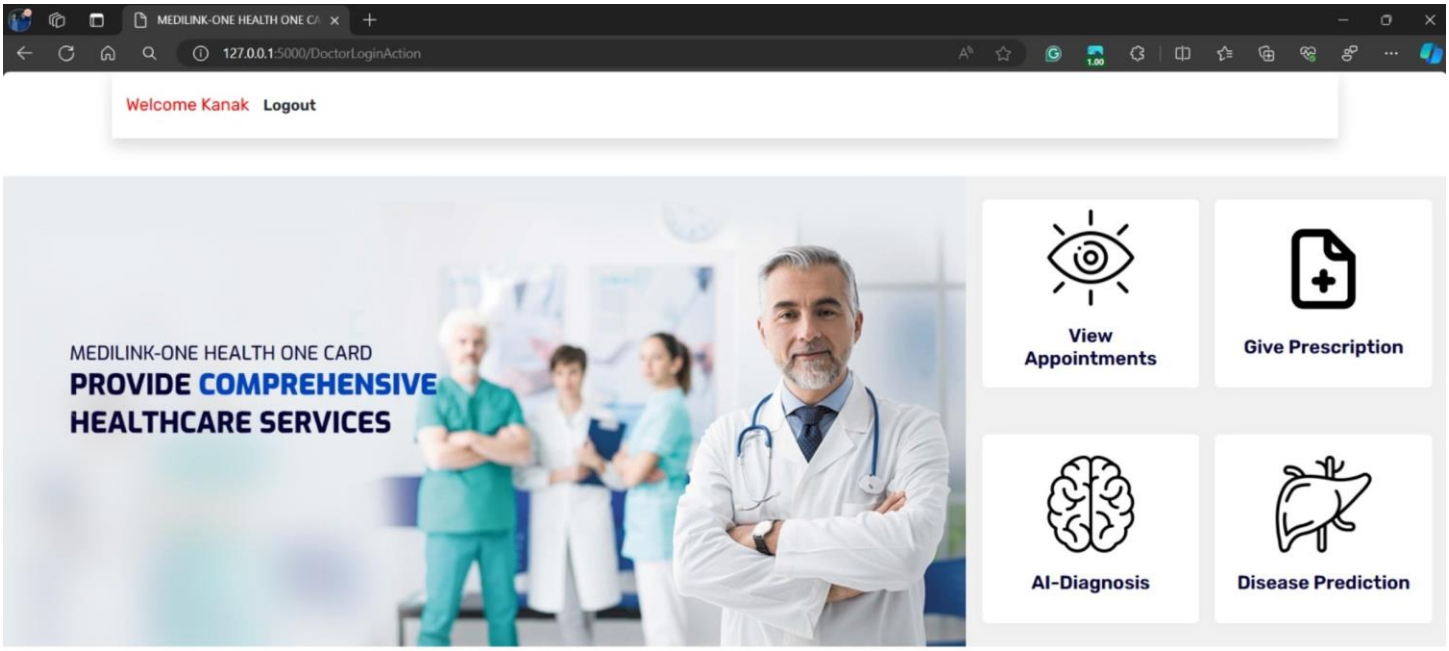


Figure A.7: Appointment Booking Screen



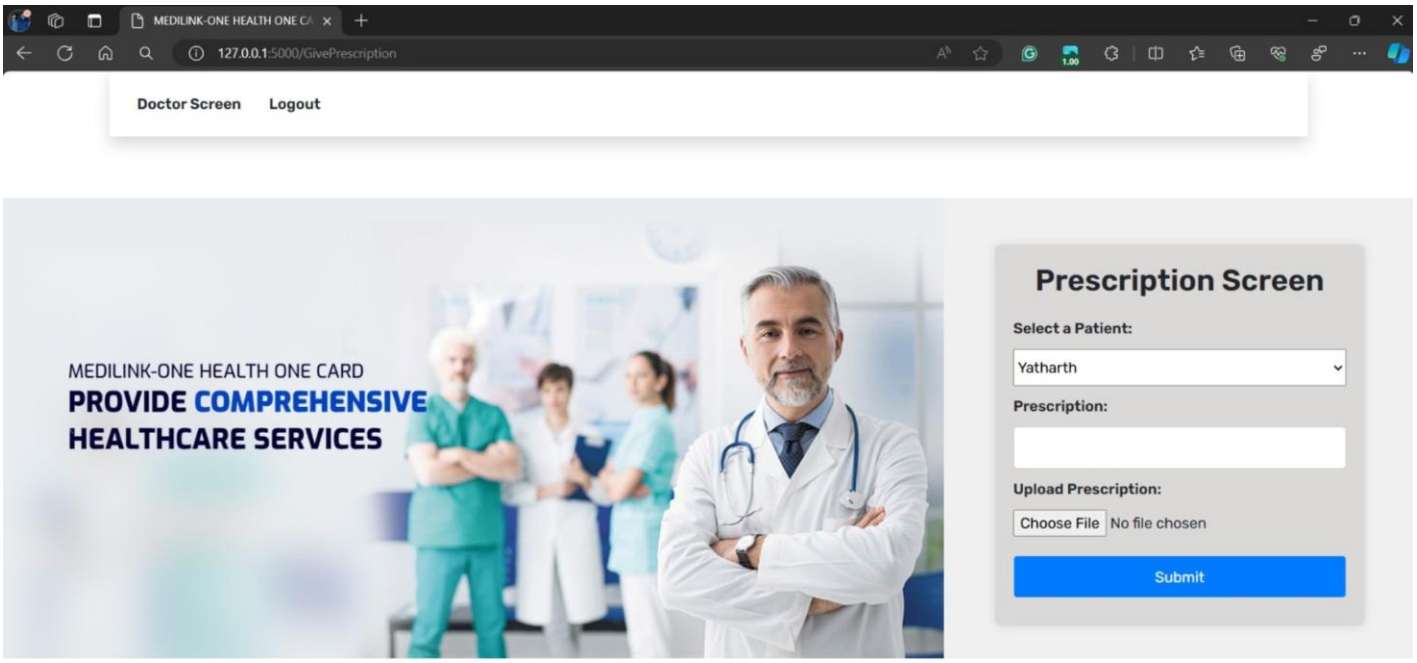
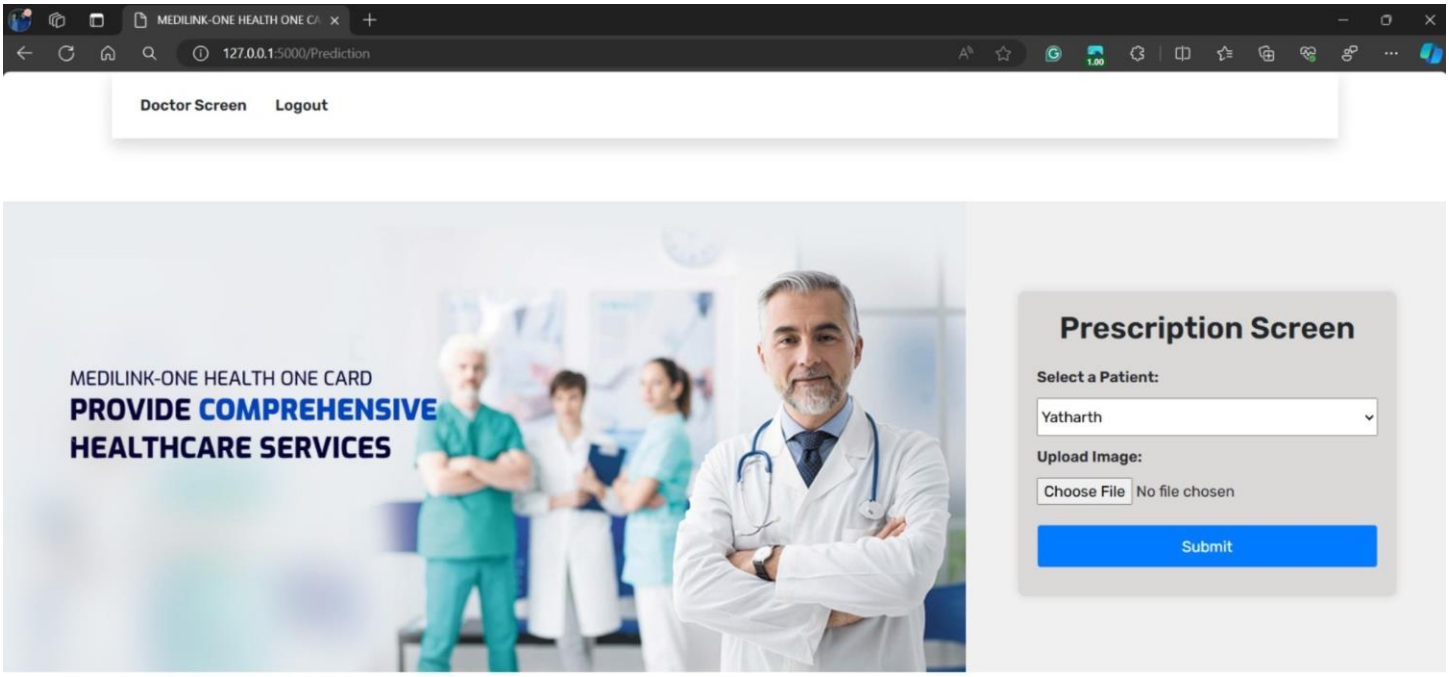


Figure A.9: Give Prescription Page (Doctor's page)



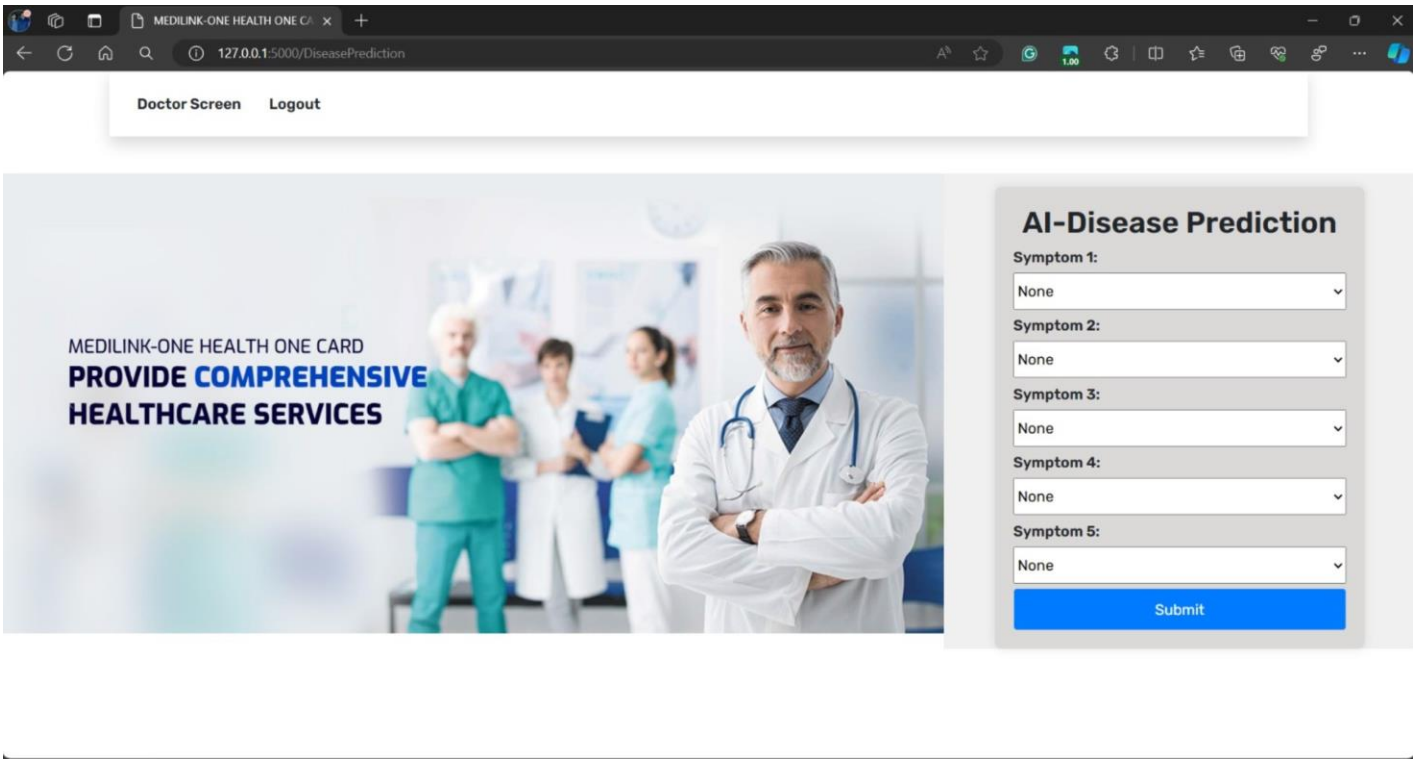
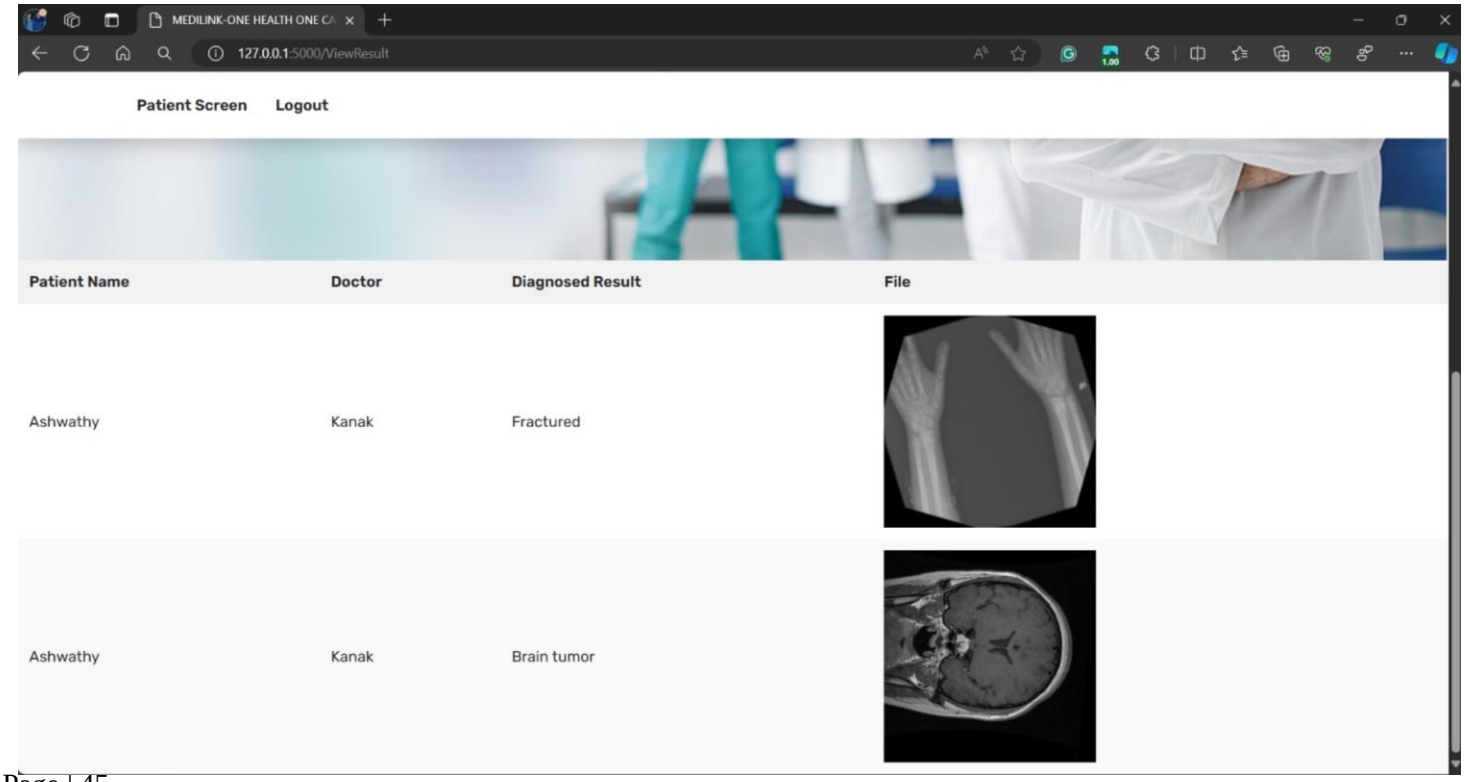


Figure A.11: AI-Disease Prediction Screen (Doctor's Page)



Appendix-B(Coding)

Importing Libraries

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import tensorflow as tf

gpus = tf.config.list_physical_devices('GPU')
if gpus:
  # Restrict TensorFlow to use only the first GPU
  try:
    tf.config.experimental.set_memory_growth(gpus[0], True)
    print("GPU found and memory growth enabled!")
  except RuntimeError as e:
    print(e)
else:
  print("No GPU detected.")


# Check for GPUs
gpus = tf.config.list_physical_devices('GPU')

if gpus:
  # Training on GPU if at least one GPU is found
  print("Training on GPU!")
else:
  # Training on CPU if no GPUs are detected
  print("Training on CPU.")
```

In [3]:
```
import pandas as pd
import pathlib
import matplotlib.pyplot as plt
import numpy as np
import os, shutil
import splitfolders
import PIL
import glob
import cv2
import time
from sklearn.metrics import confusion_matrix, confusion_matrix
import seaborn as sns
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusio
import sys
import shutil
import glob as gb
import pickle
from sklearn.svm import SVC
```

In [4]:

```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import Callback, EarlyStopping, ModelCheckpoint
from tensorflow.keras import datasets, layers, models, losses
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Activation, Flatten, Dense,
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.losses import CategoricalCrossentropy
import tensorflow.keras.backend as K
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential
import tensorflow as tf
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from tensorflow.keras.models import Sequential, Model
from sklearn.model_selection import train_test_split
```

In [5]:

```
tf.test.is_gpu_available()
```

Importing Dataset ¶

In [4]:

```
# class config:

#     data_path = 'Dataset/'

#     path_train = "./output/train"
#     path_test = "./output/test"
```

In [5]:

```
# splitfolders.ratio(config.data_path, output="output", seed=101, ratio=(
```

CNN

In [48]:
```
BATCH_SIZE = 128
IMAGE_SHAPE = (224, 224)
```

In [49]:
```
TRAIN_PATH = "output/train"
VAL_PATH = "output/val"
```

In [50]:
```
datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255)

train_gen = datagen.flow_from_directory(directory = TRAIN_PATH,
                                        class_mode="categorical",
                                        target_size = IMAGE_SHAPE,
                                        batch_size = BATCH_SIZE,
                                        color_mode='rgb',
                                        seed = 1234,
                                        shuffle = True)

val_gen = datagen.flow_from_directory(directory = VAL_PATH,
                                        class_mode="categorical",
                                        target_size = IMAGE_SHAPE,
                                        batch_size = BATCH_SIZE,
                                        color_mode='rgb',
                                        seed = 1234,
                                        shuffle = True)
```

```
Found 47658 images belonging to 15 classes.
Found 11919 images belonging to 15 classes.
```

In [51]:
```
def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_score(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)

    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

In [22]:
```
#suraj tests please ignore
import tensorflow as tf

# Check for GPU availability
if tf.config.list_physical_devices('GPU'):
    print("GPU is available!")

    # Create a large tensor on the GPU
    with tf.device('/device:GPU:0'):  # Explicitly place on GPU (optional)
        large_tensor = tf.random.normal((1024, 1024))

    # Perform computationally intensive operations (adjust for your needs)
    for _ in range(100):
        large_tensor = tf.matmul(large_tensor, large_tensor)

    # GPU memory usage (optional)
    print(f"GPU memory usage: {tf.config.experimental.list_physical_devices('GPU')}")
else:
    print("GPU is not available.")
```

```
GPU is available!
GPU memory usage: [PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

In [52]:
```
model = Sequential()
model.add(Conv2D(96, (11, 11), strides=(4, 4), activation='relu', input_shape=(224,
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
model.add(Conv2D(256, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
model.add(Conv2D(384, (3, 3), activation='relu'))
model.add(Conv2D(384, (3, 3), activation='relu'))
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2)))
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(15, activation='softmax'))
```

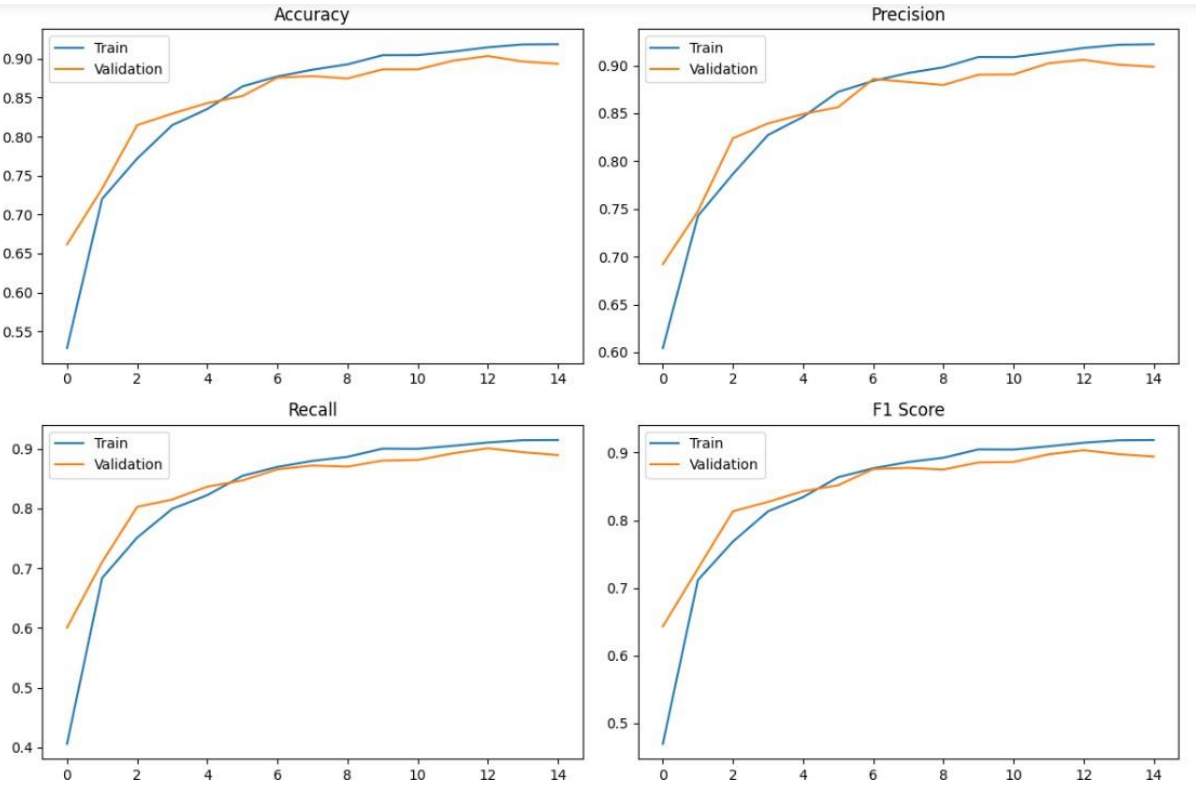
In [53]:
```
model.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy',
history = model.fit(train_gen, validation_data=val_gen, epochs=15, steps_per_epoch=1
```

```
Epoch 1/15
373/373 [==============================] - 262s 688ms/step - loss: 1.1112 - accuracy
063 - precision_m: 0.6044 - val_loss: 0.6922 - val_accuracy: 0.6617 - val_f1_score:
on_m: 0.6922
Epoch 2/15
373/373 [==============================] - 246s 659ms/step - loss: 0.5839 - accuracy
837 - precision_m: 0.7426 - val_loss: 0.5405 - val_accuracy: 0.7332 - val_f1_score:
on_m: 0.7475
Epoch 3/15
```

```
373/373 [==============================] - 262s 702ms/step - loss: 0.4796 - accuracy: 0.7
515 - precision_m: 0.7866 - val_loss: 0.4235 - val_accuracy: 0.8147 - val_f1_score: 0.81
on_m: 0.8239
Epoch 4/15
373/373 [==============================] - 242s 650ms/step - loss: 0.4083 - accuracy: 0.8
996 - precision_m: 0.8274 - val_loss: 0.3854 - val_accuracy: 0.8296 - val_f1_score: 0.82
on_m: 0.8394
Epoch 5/15
373/373 [==============================] - 241s 645ms/step - loss: 0.3684 - accuracy: 0.8
225 - precision_m: 0.8463 - val_loss: 0.3567 - val_accuracy: 0.8431 - val_f1_score: 0.84
on_m: 0.8493
Epoch 6/15
373/373 [==============================] - 247s 663ms/step - loss: 0.3112 - accuracy: 0.8
549 - precision_m: 0.8725 - val_loss: 0.3514 - val_accuracy: 0.8521 - val_f1_score: 0.85
on_m: 0.8564
Epoch 7/15
373/373 [==============================] - 231s 618ms/step - loss: 0.2800 - accuracy: 0.8
697 - precision_m: 0.8839 - val_loss: 0.2955 - val_accuracy: 0.8757 - val_f1_score: 0.87
on_m: 0.8859
Epoch 8/15
373/373 [==============================] - 233s 623ms/step - loss: 0.2598 - accuracy: 0.8
799 - precision_m: 0.8922 - val_loss: 0.2772 - val_accuracy: 0.8778 - val_f1_score: 0.87
on_m: 0.8828
Epoch 9/15
373/373 [==============================] - 219s 586ms/step - loss: 0.2511 - accuracy: 0.8
867 - precision_m: 0.8981 - val_loss: 0.2969 - val_accuracy: 0.8747 - val_f1_score: 0.87
on_m: 0.8797
Epoch 10/15
373/373 [==============================] - 214s 575ms/step - loss: 0.2264 - accuracy: 0.9
003 - precision_m: 0.9090 - val_loss: 0.2704 - val_accuracy: 0.8864 - val_f1_score: 0.88
```

In [42]:
```
model.save('model/cnn.h5')
```

In [55]:
```
train_acc = history.history['accuracy']
train_recall = history.history['recall_m']
train_precision = history.history['precision_m']
train_f1 = history.history['f1_score']
val_acc = history.history['val_accuracy']
val_recall = history.history['val_recall_m']
val_precision = history.history['val_precision_m']
val_f1 = history.history['val_f1_score']

# Create a figure and subplot for each metric
fig, axs = plt.subplots(2, 2, figsize=(12, 8))

# Plot accuracy
axs[0, 0].plot(train_acc, label='Train')
axs[0, 0].plot(val_acc, label='Validation')
axs[0, 0].set_title('Accuracy')
axs[0, 0].legend()

# Plot precision
axs[0, 1].plot(train_precision, label='Train')
axs[0, 1].plot(val_precision, label='Validation')
axs[0, 1].set_title('Precision')
axs[0, 1].legend()

# Plot recall
axs[1, 0].plot(train_recall, label='Train')
axs[1, 0].plot(val_recall, label='Validation')
axs[1, 0].set_title('Recall')
axs[1, 0].legend()

# Plot F1 score
axs[1, 1].plot(train_f1, label='Train')
axs[1, 1].plot(val_f1, label='Validation')
axs[1, 1].set_title('F1 Score')
axs[1, 1].legend()

# Adjust spacing between subplots
plt.tight_layout()

# Display the plot
plt.show()
```


Accuracy
Train
Validation
Precision
Train
Validation
Recall
Train
Validation
F1 Score
Train
Validation

In [56]:
```
a1 = history.history['accuracy'][-1]
f1 = history.history['f1_score'][-1]
p1 = history.history['precision_m'][-1]
r1 = history.history['recall_m'][-1]
```

In [57]:
```
print('Accuracy = ' + str(a1))
print('Precision = ' + str(p1))
print('F1 Score = ' + str(f1))
print('Recall = ' + str(r1))
```

```
Accuracy = 0.9186705350875854
Precision = 0.9223446249961853
F1 Score = 0.9185765385627747
Recall = 0.9148763418197632
```

Xception

In [58]:
```
BATCH_SIZE = 64
IMAGE_SHAPE = (224, 224)

TRAIN_PATH = "output/train"
VAL_PATH = "output/val"

datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255)

train_gen = datagen.flow_from_directory(directory = TRAIN_PATH,
                                        class_mode="categorical",
                                        target_size = IMAGE_SHAPE,
                                        batch_size = BATCH_SIZE,
                                        color_mode='rgb',
                                        seed = 1234,
                                        shuffle = True)

val_gen = datagen.flow_from_directory(directory = VAL_PATH,
```

```
val_gen = datagen.flow_from_directory(directory = VAL_PATH,
                                      class_mode="categorical",
                                      target_size = IMAGE_SHAPE,
                                      batch_size = BATCH_SIZE,
                                      color_mode='rgb',
                                      seed = 1234,
                                      shuffle = True)

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_score(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)

    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

```
Found 47658 images belonging to 15 classes.
Found 11919 images belonging to 15 classes.
```

In [59]:
```
inc = tf.keras.applications.xception.Xception(include_top=False, weights='imagenet'


x31 = Flatten()(inc.output)
predictionss = Dense(15, activation='softmax')(x31)
```

```
model = Model(inputs = inc.inputs, outputs = predictionss)
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weig
tf_kernels_notop.h5
83683744/83683744 [==============================] - 7s 0us/step
Model: "model_3"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_5 (InputLayer)        [(None, 224, 224, 3)]     0         []

 block1_conv1 (Conv2D)       (None, 111, 111, 32)      864       ['input_5[0][0]']

 block1_conv1_bn (BatchNorm  (None, 111, 111, 32)      128       ['block1_conv1[0][0]']
 alization)

 block1_conv1_act (Activati  (None, 111, 111, 32)      0         ['block1_conv1_bn[0][0]']
 on)

 block1_conv2 (Conv2D)       (None, 109, 109, 64)      18432     ['block1_conv1_act[0][0]']
```

In [18]:
```
model.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy',f1_score,recall_m,pre
history = model.fit(train_gen, validation_data=val_gen, epochs=15, steps_per_epoch=len(train_gen), valida
```

```
Epoch 1/15
745/745 [==============================] - 219s 290ms/step - loss: 0.2201 - accuracy: 0.9168 - f1_score:
102 - precision_m: 0.9214 - val_loss: 0.4645 - val_accuracy: 0.8890 - val_f1_score: 0.8887 - val_recall_m
on_m: 0.8936
Epoch 2/15
745/745 [==============================] - 212s 284ms/step - loss: 0.0858 - accuracy: 0.9703 - f1_score:
```

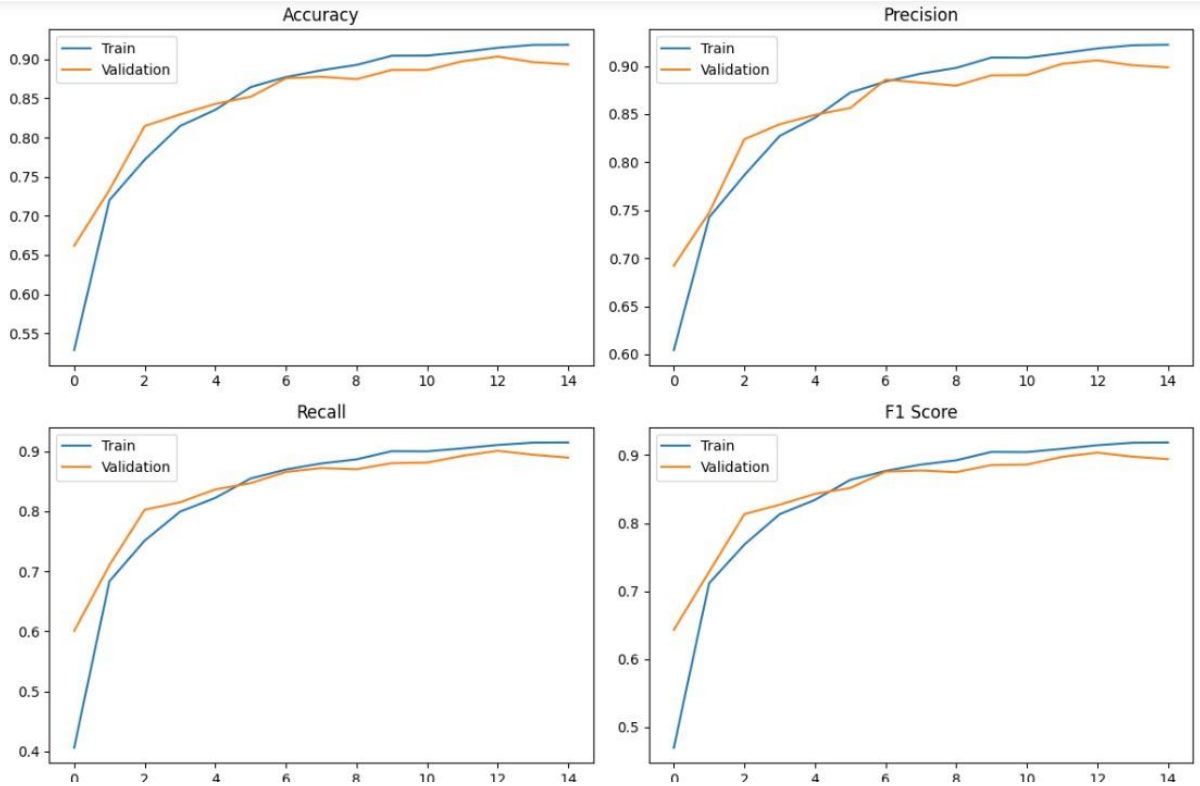
```
Epoch 3/15
745/745 [==============================] - 215s 289ms/step - loss: 0.0657 - accuracy: 0.97
775 - precision_m: 0.9790 - val_loss: 0.1361 - val_accuracy: 0.9652 - val_f1_score: 0.9654
on_m: 0.9660
Epoch 4/15
745/745 [==============================] - 212s 285ms/step - loss: 0.0514 - accuracy: 0.98
826 - precision_m: 0.9837 - val_loss: 0.0789 - val_accuracy: 0.9744 - val_f1_score: 0.9744
on_m: 0.9755
Epoch 5/15
745/745 [==============================] - 219s 294ms/step - loss: 0.0452 - accuracy: 0.98
853 - precision_m: 0.9862 - val_loss: 0.0906 - val_accuracy: 0.9742 - val_f1_score: 0.9742
on_m: 0.9746
Epoch 6/15
745/745 [==============================] - 213s 286ms/step - loss: 0.0403 - accuracy: 0.98
867 - precision_m: 0.9876 - val_loss: 0.0567 - val_accuracy: 0.9830 - val_f1_score: 0.9828
on_m: 0.9836
Epoch 7/15
745/745 [==============================] - 214s 287ms/step - loss: 0.0344 - accuracy: 0.98
893 - precision_m: 0.9900 - val_loss: 0.0417 - val_accuracy: 0.9857 - val_f1_score: 0.9854
on_m: 0.9857
Epoch 8/15
745/745 [==============================] - 211s 283ms/step - loss: 0.0300 - accuracy: 0.99
904 - precision_m: 0.9911 - val_loss: 0.2360 - val_accuracy: 0.9265 - val_f1_score: 0.9278
on_m: 0.9316
Epoch 9/15
745/745 [==============================] - 214s 286ms/step - loss: 0.0290 - accuracy: 0.99
907 - precision_m: 0.9911 - val_loss: 0.0541 - val_accuracy: 0.9848 - val_f1_score: 0.9846
on_m: 0.9848
Epoch 10/15
745/745 [==============================] - 212s 284ms/step - loss: 0.0246 - accuracy: 0.99
918 - precision_m: 0.9921 - val_loss: 0.0529 - val_accuracy: 0.9847 - val_f1_score: 0.9846
on_m: 0.9849
Epoch 11/15
745/745 [==============================] - 211s 284ms/step - loss: 0.0197 - accuracy: 0.99
937 - precision_m: 0.9939 - val_loss: 0.0481 - val_accuracy: 0.9860 - val_f1_score: 0.9861
on_m: 0.9865
Epoch 12/15
745/745 [==============================] - 213s 286ms/step - loss: 0.0209 - accuracy: 0.99
```

```
745/745 [==============================] - 213s 286ms/step - loss: 0.0209 - accuracy: 0.9932
930 - precision_m: 0.9933 - val_loss: 0.1292 - val_accuracy: 0.9607 - val_f1_score: 0.9609 -
on_m: 0.9632
Epoch 13/15
745/745 [==============================] - 213s 286ms/step - loss: 0.0197 - accuracy: 0.9936
935 - precision_m: 0.9938 - val_loss: 0.0894 - val_accuracy: 0.9729 - val_f1_score: 0.9730 -
on_m: 0.9748
Epoch 14/15
745/745 [==============================] - 213s 286ms/step - loss: 0.0166 - accuracy: 0.9946
945 - precision_m: 0.9948 - val_loss: 0.2247 - val_accuracy: 0.9459 - val_f1_score: 0.9459 -
on_m: 0.9471
Epoch 15/15
745/745 [==============================] - 213s 286ms/step - loss: 0.0189 - accuracy: 0.9937
937 - precision_m: 0.9939 - val_loss: 0.0318 - val_accuracy: 0.9912 - val_f1_score: 0.9913 -
on_m: 0.9915
```

In [19]:
```
model.save('model/Xception.h5')
```

In [60]:
```
train_acc = history.history['accuracy']
train_recall = history.history['recall_m']
train_precision = history.history['precision_m']
train_f1 = history.history['f1_score']
val_acc = history.history['val_accuracy']
val_recall = history.history['val_recall_m']
val_precision = history.history['val_precision_m']
val_f1 = history.history['val_f1_score']

# Create a figure and subplot for each metric
fig, axs = plt.subplots(2, 2, figsize=(12, 8))

# Plot accuracy
axs[0, 0].plot(train_acc, label='Train')
axs[0, 0].plot(val_acc, label='Validation')
axs[0, 0].set_title('Accuracy')
axs[0, 0].legend()
```

```
# Plot precision
axs[0, 1].plot(train_precision, label='Train')
axs[0, 1].plot(val_precision, label='Validation')
axs[0, 1].set_title('Precision')
axs[0, 1].legend()

# Plot recall
axs[1, 0].plot(train_recall, label='Train')
axs[1, 0].plot(val_recall, label='Validation')
axs[1, 0].set_title('Recall')
axs[1, 0].legend()

# Plot F1 score
axs[1, 1].plot(train_f1, label='Train')
axs[1, 1].plot(val_f1, label='Validation')
axs[1, 1].set_title('F1 Score')
axs[1, 1].legend()

# Adjust spacing between subplots
plt.tight_layout()

# Display the plot
plt.show()
```



In [61]:
```
a2 = history.history['accuracy'][-1]
f2 = history.history['f1_score'][-1]
p2 = history.history['precision_m'][-1]
r2 = history.history['recall_m'][-1]


print('Accuracy = ' + str(a2))
print('Precision = ' + str(p2))
print('F1 Score = ' + str(f2))
print('Recall = ' + str(r2))
```

```
Accuracy = 0.9186705350875854
Precision = 0.9223446249961853
F1 Score = 0.9185765385627747
Recall = 0.9148763418197632
```

ResNET (Residual Neural Network)

In [24]:
```
BATCH_SIZE = 64
IMAGE_SHAPE = (224, 224)

TRAIN_PATH = "output/train"
VAL_PATH = "output/val"

datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255)

train_gen = datagen.flow_from_directory(directory = TRAIN_PATH,
                                        class_mode="categorical",
                                        target_size = IMAGE_SHAPE,
                                        batch_size = BATCH_SIZE,
                                        color_mode='rgb',
                                        seed = 1234,
                                        shuffle = True)

val_gen = datagen.flow_from_directory(directory = VAL_PATH,
```

```
                                            class_mode="categorical",
                                            target_size = IMAGE_SHAPE,
                                            batch_size = BATCH_SIZE,
                                            color_mode='rgb',
                                            seed = 1234,
                                            shuffle = True)

def recall_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))
    recall = true_positives / (possible_positives + K.epsilon())
    return recall

def precision_m(y_true, y_pred):
    true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))
    predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))
    precision = true_positives / (predicted_positives + K.epsilon())
    return precision

def f1_score(y_true, y_pred):
    precision = precision_m(y_true, y_pred)
    recall = recall_m(y_true, y_pred)

    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

```
Found 47658 images belonging to 15 classes.
Found 11919 images belonging to 15 classes.
```

In [31]:
```
resnet = tf.keras.applications.ResNet50(weights='imagenet', include_top=False, input_
for layer in resnet.layers:
    layer.trainable = False

# Add custom layers on top of ResNet
x = resnet.output
x = Flatten()(x)
x = Dense(512, activation='relu')(x)
predictions = Dense(15, activation='softmax')(x)
```

```
# Create a new model
resnet_model = Model(inputs=resnet.input, outputs=predictions)
```

In [34]:
```
resnet_model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['a
```

In [35]:
```
history_resnet = resnet_model.fit(train_gen, validation_data=val_gen, epochs=15, st
```

```
Epoch 1/15
745/745 [==============================] - 290s 384ms/step - loss: 1.0631 - accurac
436 - precision_m: 0.7273 - val_loss: 0.7391 - val_accuracy: 0.6723 - val_f1_score:
on_m: 0.7311
Epoch 2/15
745/745 [==============================] - 310s 416ms/step - loss: 0.7037 - accurac
260 - precision_m: 0.7548 - val_loss: 0.7110 - val_accuracy: 0.7221 - val_f1_score:
on_m: 0.7597
Epoch 3/15
745/745 [==============================] - 346s 465ms/step - loss: 0.6270 - accurac
720 - precision_m: 0.7715 - val_loss: 0.5913 - val_accuracy: 0.7406 - val_f1_score:
on_m: 0.7821
Epoch 4/15
745/745 [==============================] - 310s 416ms/step - loss: 0.5842 - accurac
997 - precision_m: 0.7843 - val_loss: 0.5576 - val_accuracy: 0.7597 - val_f1_score:
on_m: 0.7992
Epoch 5/15
745/745 [==============================] - 305s 410ms/step - loss: 0.5510 - accurac
209 - precision_m: 0.7912 - val_loss: 0.5201 - val_accuracy: 0.7805 - val_f1_score:
on_m: 0.8113
Epoch 6/15
745/745 [==============================] - 344s 460ms/step - loss: 0.5374 - accurac
316 - precision_m: 0.7970 - val_loss: 0.5365 - val_accuracy: 0.7681 - val_f1_score:
on_m: 0.7934
Epoch 7/15
745/745 [==============================] - 350s 470ms/step - loss: 0.5123 - accurac
419 - precision_m: 0.8034 - val_loss: 0.5544 - val_accuracy: 0.7645 - val_f1_score:
on_m: 0.7938
```

```
Epoch 8/15
745/745 [==============================] - 532s 714ms/step - loss: 0.4826 - accuracy: 0.79
612 - precision_m: 0.8150 - val_loss: 0.5652 - val_accuracy: 0.7522 - val_f1_score: 0.7522
on_m: 0.7705
Epoch 9/15
745/745 [==============================] - 351s 469ms/step - loss: 0.4889 - accuracy: 0.78
584 - precision_m: 0.8091 - val_loss: 0.5135 - val_accuracy: 0.7823 - val_f1_score: 0.7794
on_m: 0.8062
Epoch 10/15
745/745 [==============================] - 314s 421ms/step - loss: 0.4664 - accuracy: 0.79
727 - precision_m: 0.8187 - val_loss: 0.4629 - val_accuracy: 0.8027 - val_f1_score: 0.7978
on_m: 0.8234
Epoch 11/15
745/745 [==============================] - 336s 450ms/step - loss: 0.4437 - accuracy: 0.80
832 - precision_m: 0.8258 - val_loss: 0.4937 - val_accuracy: 0.7871 - val_f1_score: 0.7831
on_m: 0.8038
Epoch 12/15
745/745 [==============================] - 311s 417ms/step - loss: 0.4611 - accuracy: 0.79
770 - precision_m: 0.8202 - val_loss: 0.5120 - val_accuracy: 0.7872 - val_f1_score: 0.7846
on_m: 0.8034
Epoch 13/15
745/745 [==============================] - 345s 463ms/step - loss: 0.4324 - accuracy: 0.81
917 - precision_m: 0.8297 - val_loss: 0.4182 - val_accuracy: 0.8189 - val_f1_score: 0.8194
on_m: 0.8390
Epoch 14/15
745/745 [==============================] - 356s 477ms/step - loss: 0.4200 - accuracy: 0.81
977 - precision_m: 0.8335 - val_loss: 0.5087 - val_accuracy: 0.7921 - val_f1_score: 0.7911
on_m: 0.8086
Epoch 15/15
745/745 [==============================] - 312s 419ms/step - loss: 0.4287 - accuracy: 0.81
956 - precision_m: 0.8302 - val_loss: 0.4793 - val_accuracy: 0.7992 - val_f1_score: 0.7957
on_m: 0.8170
```

In [ ]:
```
# resnet = tf.keras.applications.ResNet50(weights='imagenet', include_top=False, input_sha

# for layer in resnet.layers:
#     layer.trainable = False
```

```
# x = resnet.output
# x = layers.GlobalAveragePooling2D()(x)
# x = layers.Dense(512, activation='relu')(x)
# x = layers.Dropout(0.5)(x)
# x = layers.Dense(256, activation='relu')(x)
# x = layers.Dropout(0.5)(x)
# x = layers.Dense(128, activation='relu')(x)
# x = layers.Dropout(0.5)(x)
# x = layers.Dense(64, activation='relu')(x)
# x = layers.Dropout(0.5)(x)

# predictions = layers.Dense(15, activation='softmax')(x)
# model = Model(inputs = resnet.input, outputs = predictions)
```

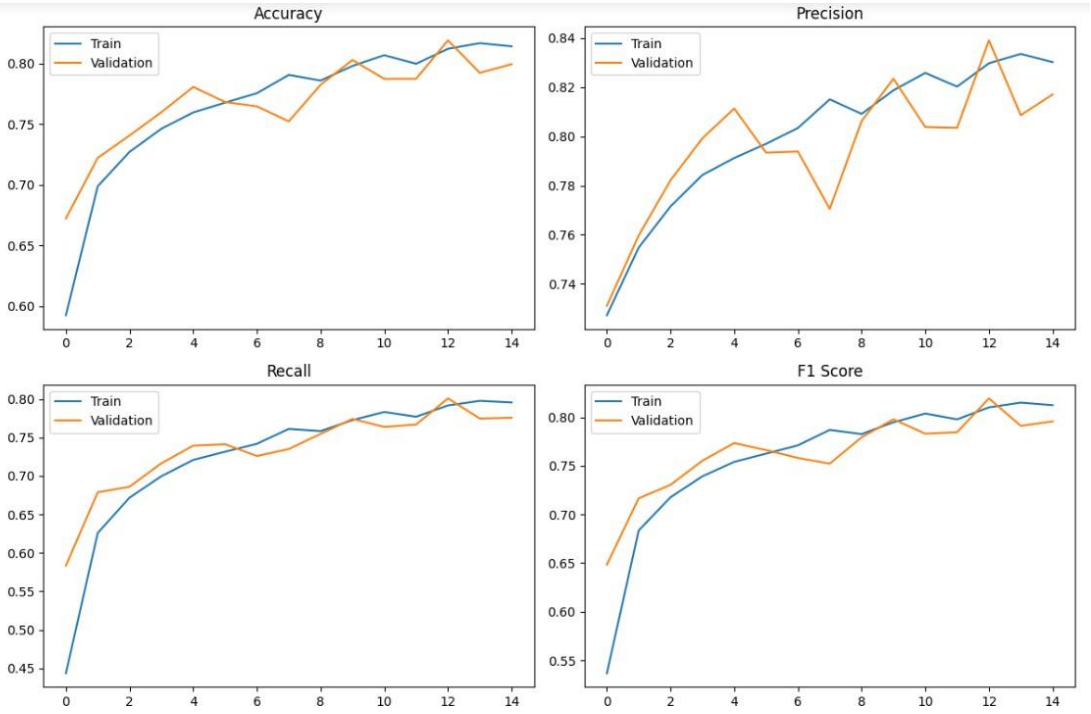
In [27]:
```
# model.compile(optimizer='adam',loss="categorical_crossentropy", metrics=[
```

In [ ]:
```
# history_resnet = model.fit(train_gen, validation_data=val_gen, epochs=15,
```

In [36]:
```
resnet_model.save('model/ResNet.h5')
```

In [37]:
```
train_acc = history_resnet.history['accuracy']
train_recall = history_resnet.history['recall_m']
train_precision = history_resnet.history['precision_m']
train_f1 = history_resnet.history['f1_score']
val_acc = history_resnet.history['val_accuracy']
val_recall = history_resnet.history['val_recall_m']
val_precision = history_resnet.history['val_precision_m']
val_f1 = history_resnet.history['val_f1_score']

# Create a figure and subplot for each metric
fig, axs = plt.subplots(2, 2, figsize=(12, 8))
```

```
# Plot accuracy
axs[0, 0].plot(train_acc, label='Train')
axs[0, 0].plot(val_acc, label='Validation')
axs[0, 0].set_title('Accuracy')
axs[0, 0].legend()

# Plot precision
axs[0, 1].plot(train_precision, label='Train')
axs[0, 1].plot(val_precision, label='Validation')
axs[0, 1].set_title('Precision')
axs[0, 1].legend()

# Plot recall
axs[1, 0].plot(train_recall, label='Train')
axs[1, 0].plot(val_recall, label='Validation')
axs[1, 0].set_title('Recall')
axs[1, 0].legend()

# Plot F1 score
axs[1, 1].plot(train_f1, label='Train')
axs[1, 1].plot(val_f1, label='Validation')
axs[1, 1].set_title('F1 Score')
axs[1, 1].legend()

# Adjust spacing between subplots
plt.tight_layout()

# Display the plot
plt.show()
```



In [39]:
```
a3 = history_resnet.history['accuracy'][-1]
f3 = history_resnet.history['f1_score'][-1]
p3 = history_resnet.history['precision_m'][-1]
r3 = history_resnet.history['recall_m'][-1]


print('Accuracy = ' + str(a3))
print('Precision = ' + str(p3))
print('F1 Score = ' + str(f3))
print('Recall = ' + str(r3))
```

```
Accuracy = 0.8139451742172241
Precision = 0.8301525712013245
F1 Score = 0.8123766779899597
Recall = 0.7956335544586182
```

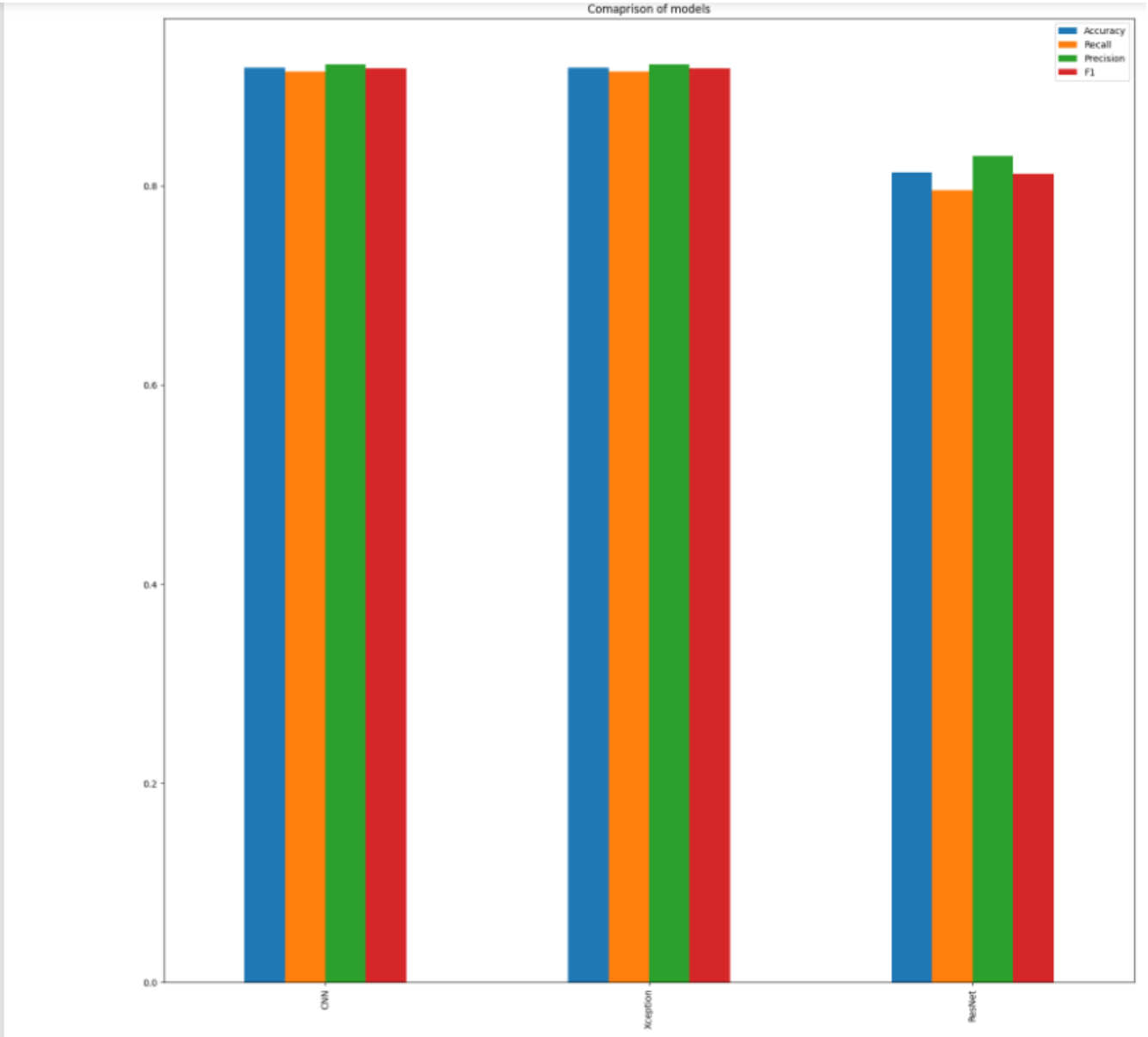
Comparison

In [62]:
```
results ={'Accuracy': [a1,a2,a3],
 'Recall':[r1,r2,r3],
 'Precision': [p1,p2,p3],
    'F1' : [f1,f2,f3]}
index = ['CNN','Xception','ResNet']
```

In [63]:
```
results =pd.DataFrame(results,index=index)
print(results)
```

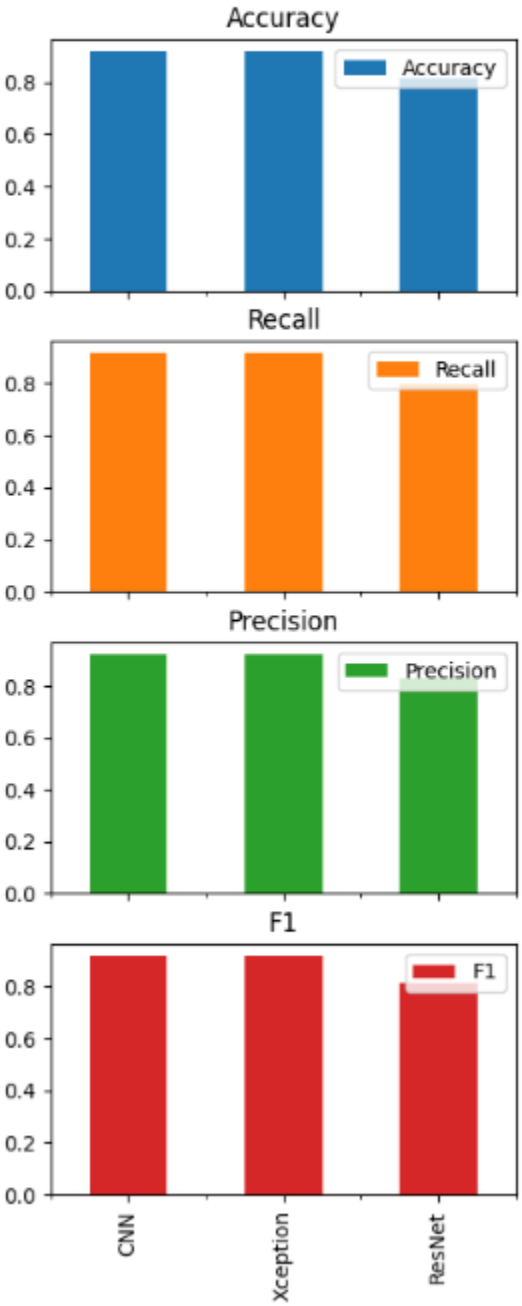
```
          Accuracy    Recall  Precision        F1
CNN       0.918671  0.914876   0.922345  0.918577
Xception  0.918671  0.914876   0.922345  0.918577
ResNet    0.813945  0.795634   0.830153  0.812377
```

In [64]:
```
fig =results.plot(kind='bar',title='Comaprison of models',figsize =(19,19)).get_figu
fig.savefig('Final Result.png')
```

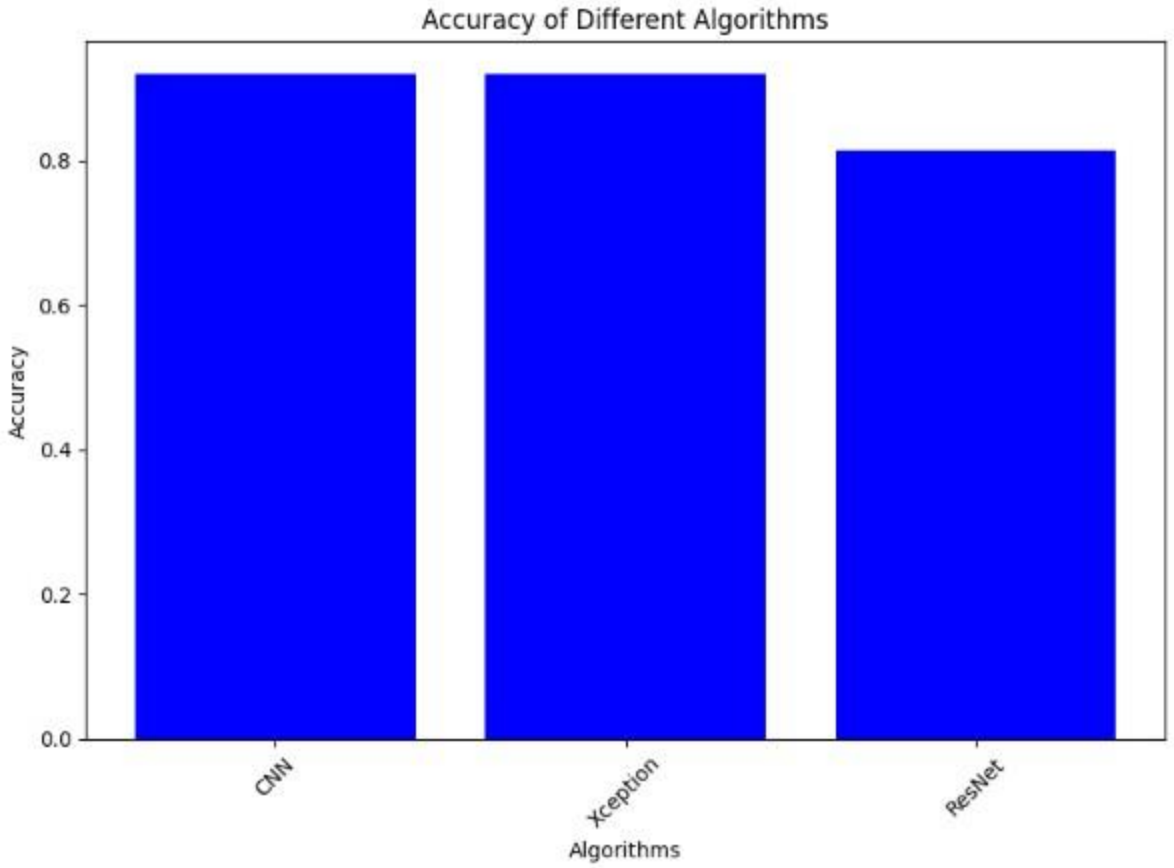

Comaprison of models
Accuracy
Recall
Precision
F1
0.0
0.2
0.4
0.6
0.8
CNN
Xception
ResNet

```
In [65]: results.plot(subplots=True,kind ='bar',figsize=(4,10))
```

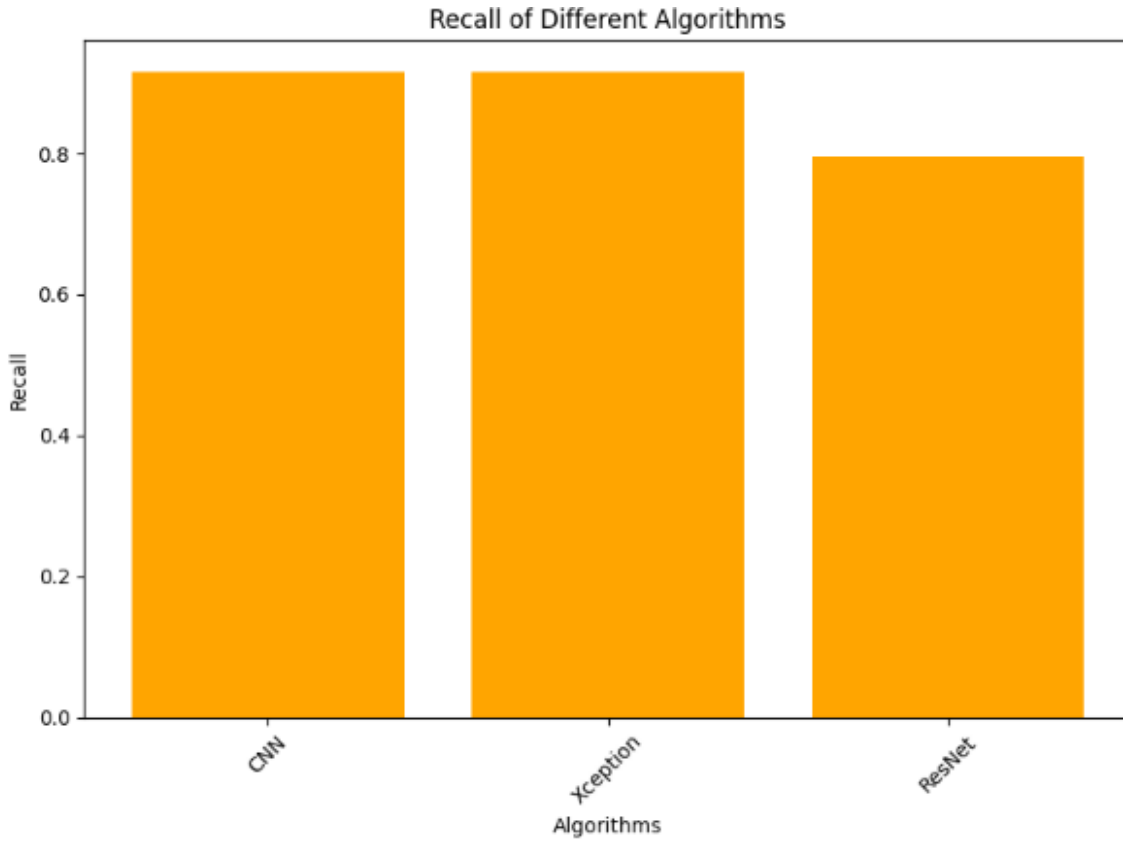
```
Out[65]: array([<Axes: title={'center': 'Accuracy'}>,
                <Axes: title={'center': 'Recall'}>,
                <Axes: title={'center': 'Precision'}>,
                <Axes: title={'center': 'F1'}>], dtype=object)
```



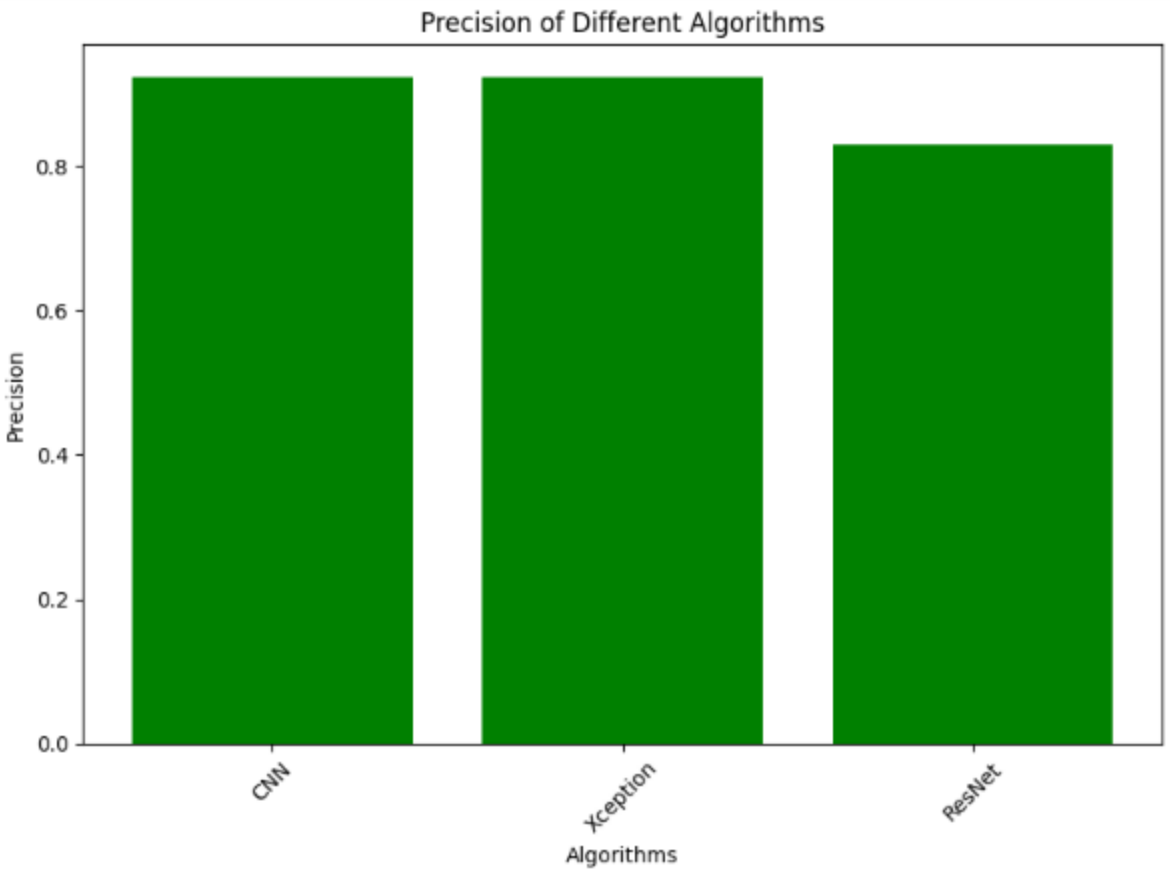
```
accuracy_values = results['Accuracy']
accuracy_df = pd.DataFrame({'Algorithms': index, 'Accuracy': accuracy_values})

plt.figure(figsize=(8, 6))
plt.bar(accuracy_df['Algorithms'], accuracy_df['Accuracy'], color='blue')
plt.title('Accuracy of Different Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Accuracy')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



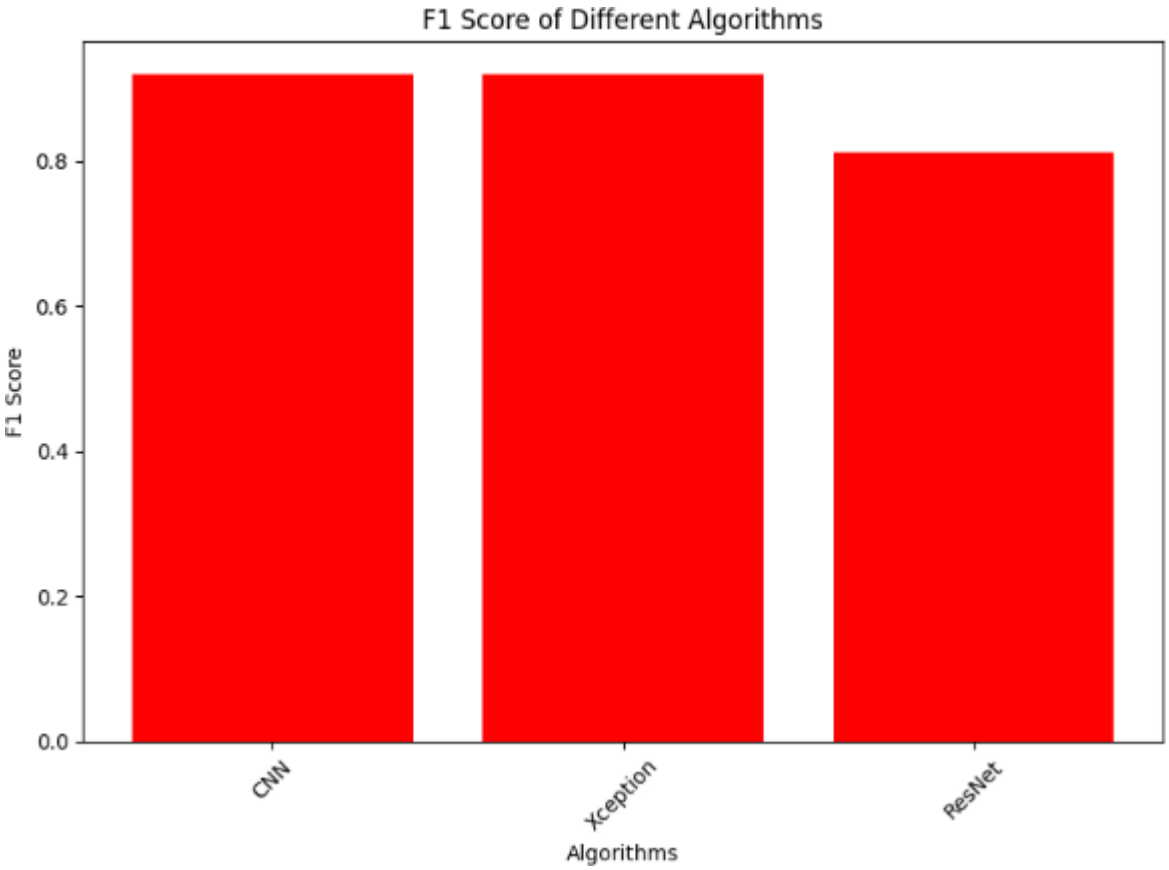
n [67]:
```
recall_values = results['Recall']
recall_df = pd.DataFrame({'Algorithms': index, 'Recall': recall_values})

plt.figure(figsize=(8, 6))
plt.bar(recall_df['Algorithms'], recall_df['Recall'], color='orange')
plt.title('Recall of Different Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Recall')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [68]:
```
precision_values = results['Precision']
precision_df = pd.DataFrame({'Algorithms': index, 'Precision': precision_values})

plt.figure(figsize=(8, 6))
plt.bar(precision_df['Algorithms'], precision_df['Precision'], color='green')
plt.title('Precision of Different Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('Precision')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



In [69]:
```
f1_values = results['F1']
f1_df = pd.DataFrame({'Algorithms': index, 'F1 Score': f1_values})

plt.figure(figsize=(8, 6))
plt.bar(f1_df['Algorithms'], f1_df['F1 Score'], color='red')
plt.title('F1 Score of Different Algorithms')
plt.xlabel('Algorithms')
plt.ylabel('F1 Score')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```


F1 Score of Different Algorithms
F1 Score
0.0
0.2
0.4
0.6
0.8
CNN
Xception
ResNet
Algorithms

REFERENCES

[1] Sanjeev Jain, Dr. Uday Pratap Singh, Sanjeev Jain, “Brain Tumor Detection and Classification Using Intelligence Techniques”.

[2]. Rohith v., “Kidney tumor detection and classification using image processing”.

[3] "Applications of CNN in Healthcare" by Medium: https://medium.com/@pankaj90382/applications-of-cnn-in-healthcare-849b2abf6d6d

[4]"A review of convolutional neural networks for image and video classification in medical imaging" by Jure Zbontar et al.: https://arxiv.org/abs/2001.05566

[5]"Applications of ResNet in Healthcare" by Medium: https://medium.com/@vinaypratapsinghvictor/resnet-the-real-superhero-in-medicalimage-processing-c1f07a983d73

[6]"Deep learning for healthcare: review, opportunities, and challenges" by Mohammad Ghassemi et al.: https://arxiv.org/abs/1804.02767

[7]"Anomaly Detection in Healthcare with AI" by TensorFlow: https://blog.tensorflow.org/2019/03/anomaly-detection-in-healthcare-with.html

[8]"Anomaly Detection in Electronic Health Records with Hybrid CNN-LSTM Based Deep Learning Models" by Jinsung Yoon et al.: https://arxiv.org/abs/1811.05281

[9]A Review on Deep Learning Techniques for Healthcare Applications" by Sharmila Nagraj et al.: https://www.researchgate.net/publication/327126274_A_Review_on_Deep_Learning_Techniques_for_Healthcare_Applications

[10] "Applications of Convolutional Neural Networks (CNNs) in Medical Image Segmentation" by Xiaohua Qian et al.: https://www.frontiersin.org/articles/10.3389/fneur.2019.01396/full

[11]"A Survey of Deep Learning Architectures and their Applications in Healthcare" by M.M. Fraz et al.: https://arxiv.org/abs/1803.03633

[12] "Deep Residual Learning for Small-Footprint Biomedical Image Segmentation" by Konstantinos Kamnitsas et al.: https://arxiv.org/abs/1606.08921

[13]"Anomaly Detection in Health Data: A Review" by Anjan K. Bhoi et al.: https://www.sciencedirect.com/science/article/pii/S2352914818301682

[14]"Deep Learning for Anomaly Detection: A Review" by Chong Zhou and Randy C. Paffenroth: https://arxiv.org/abs/1901.03407

[15]"Deep Learning in Medical Image Analysis" by Dinggang Shen et al.: https://www.sciencedirect.com/science/article/pii/S1361841518301184

[16]"Applications of Convolutional Neural Networks in Health Informatics" by Qingyao Wu et al.: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6313929/

[17]"Deep Learning for Healthcare Applications Based on Physiological Signals: A Review" by Amine Bendini et al.: https://www.mdpi.com/2504-446X/2/1/4/htm

[18]"Deep Residual Learning for MRI Reconstruction" by Kerem Caglar et al.: https://arxiv.org/abs/1808.06710