TRIBHUVAN UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
CENTRAL CAMPUS OF TECHNOLOGY (CCT), HATTISAR, DHARAN

A Internship Report On

CLOUD AND DEVOPS ENGINEER

Submitted To:
DEPARTMENT OF IT
CENTRAL CAMPUS OF TECHNOLOGY (CCT), HATTISAR, DHARAN, SUNSARI, KOSHI, NEPAL

In partial fulfilment of the requirements for the degree of Bachelor's of Science in Computer Science and Information Technology (B.Sc. CSIT)

Submitted By:
ROHAN KHANAL [29604/078]

TU Registration No : 5-2-8-148-2021

Under the Supervision of:
¡Supervisor's Name¿

Bhadra, 2082

INTERNSHIP CERTIFICATION

SUPERVISOR RECOMMENDATION

This is to recommend that **ROHAN KHANAL [29604/078]** has carried out Internship on the position of "**Cloud and DevOps Engineer**" in partial fulfilment of the requirements for the degree of Bachelor of Science (B.Sc.) degree in CSIT under my supervision in the Department of Information Technology, Central Campus of Technology, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal.

To my knowledge, this work has not been submitted for any other degree. The students have fulfilled all the requirements laid down by the Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal for the submission of the project work for the partial fulfillment of Bachelor of Science (B.Sc.) degree in CSIT

Mr. Prakash Neupane
Supervisor
Department of IT
Central Campus of Technology
Hattisar, Dharan, Sunsari, Koshi, Nepal

CERTIFICATE OF APPROVAL

This is to certify that the internship report entitled "**CLOUD AND DEVOPS ENGINEER**", submitted by **Mr. ROHAN KHANAL [29604/078]**, a student enrolled in the Bachelor of Science (B.Sc.) degree in CSIT at the Central Campus of Technology, Dharan, has been examined and approved by the undersigned.

The report has been evaluated based on its relevance to the internship experience, presentation of facts, adherence to prescribed guidelines, and overall quality. It has been found to meet the standards set by Tribhuvan University for the partial fulfillment of the requirements for the bachelor's degree.

Mr. Prakash Neupane
Supervisor
Department of IT
Central Campus of Technology
IOST, Tribhuvan University

Mr. Balabhadra Bhandari
Program Director
Department of IT
Central Campus of Technology
IOST, Tribhuvan University

Mr. Pradip Khatiwada
External Examiner
Mechi Multiple Campus
IOST, Tribhuvan University

DECLARATION

This internship report, entitled **"CLOUD AND DEVOPS ENGINEER"**, is being submitted to the Department of IT, Central Campus of Technology, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal, for the partial fulfillment of the requirements for the Bachelor of Science (B.Sc.) degree in CSIT. This project has been carried out by us under the supervision of Prakash Neupane, T.U., Department of IT, Central Campus of Technology, Institute of Science and Technology (IoST), Tribhuvan University (T.U.), Nepal.

We hereby declare that this work is original and has not been submitted earlier, in part or in full, to this or any other university or institution, here or elsewhere, for the award of any degree.

ROHAN KHANAL [29604/078]
Department of IT
Central Campus of Technology
Hattisar, Dharan, Sunsari, Koshi, Nepal

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude to **GENESE Solutions Pvt. Ltd.** for providing me with the opportunity to undertake my internship as a **Cloud and DevOps Engineer**. This internship provided valuable industry exposure and practical experience, allowing me to strengthen my technical knowledge and professional skills in a real-world working environment.

I am deeply grateful to my mentor, **Mr. Desh Deepak Dhobi**, for his continuous guidance, technical support, and encouragement throughout the internship period. His mentorship played a vital role in enhancing my understanding of cloud and DevOps practices.

I would also like to extend my sincere thanks to my supervisor, **Mr. Prakash Neupane**, for his support, constructive feedback, and supervision during the internship. His guidance greatly contributed to the successful completion of my assigned tasks and projects.

Furthermore, I would like to thank the faculty members of the **Department of Computer Science and Information Technology**, **Institute of Science and Technology (IoST)**, **Tribhuvan University**, for their academic guidance and support throughout my studies.

I am thankful to my colleagues, friends, and fellow students for their encouragement and support during this internship. I am also sincerely grateful to my family for their unwavering support and motivation throughout this journey.

Finally, I would like to thank everyone who directly or indirectly contributed to making this internship a valuable and enriching learning experience.

Thank you.

ROHAN KHANAL [29604/078]

ABSTRACT

This report presents the work carried out during the internship as a **Cloud and DevOps Engineer** at **GENESE Solutions Pvt. Ltd.**. The internship was conducted with the objective of gaining practical exposure to cloud computing, system administration, and DevOps practices in an industry environment.

During the internship period, various cloud technologies and tools were studied and implemented, primarily using Amazon Web Services (AWS). Major activities included system monitoring using CloudWatch, secure access management using IAM and RDS IAM authentication, and infrastructure maintenance through AWS Systems Manager Patch Manager. The internship also involved automation of deployment workflows by designing and implementing a CI/CD pipeline for a Lambda monorepo using AWS SAM.

In addition, tasks related to email systems, website testing, and operational support were performed. These included conducting a proof of concept for an SMTP server, resolving email rendering issues, and implementing features related to DKIM management. Research on web application security using AWS WAF was also carried out.

Overall, the internship provided practical knowledge of cloud infrastructure, DevOps automation, and operational best practices, helping to bridge the gap between academic learning and real-world application.

Keywords: Cloud Computing, DevOps, AWS, CI/CD, Monitoring, Security.

CONTENTS

LIST OF ABBREVIATIONS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1. Introduction

Brain tumors are among the most lethal forms of cancer, contributing to high mortality and morbidity rates worldwide. Early and accurate detection plays a vital role in improving patient outcomes, as timely treatment significantly enhances survival rates. However, traditional methods of diagnosis, such as manual Magnetic Resonance Imaging (MRI) analysis, prove to be time-consuming, laborious, and error-prone. In recent years, significant progression has been witnessed in the field of Artificial Intelligence, and deep learning, in particular, shows great promise in overcoming these limitations. [1]

In the field of intelligent automation and high accuracy for discriminating brain tumor presence from medical images, Convolutional Neural Networks, or Convolutional Neural Network (CNN)s, demonstrate remarkable potential. This research discusses the brain tumor detection and classification problem. It provides an automated approach to identify the position, size, and shape of tumors through deep learning techniques applied to medical images. Numerous studies have been undertaken using machine learning and deep learning techniques for this purpose, with CNN-based models consistently demonstrating some of the most effective results in medical image analysis. [2]

1.2. Problem Statement

Brain tumors requiring timely diagnosis for effective treatment and improved survival rates. Conventional diagnostic methods, primarily reliant on manual analysis of MRI scans by radiologists, are time-consuming, prone to human error, and often lead to inconsistent results, especially in early detection stages. The rising volume of medical imaging data further exacerbates this issue, putting additional strain on healthcare systems. There is a compelling need for an automated, accurate, and efficient system that can assist in detecting and classifying brain tumors to support medical professionals in making informed decisions.[3]

1.3. Objectives

The main objectives of this project are as follows:

- To design and implement a convolutional neural network (CNN) model.
- To incorporate transfer learning with VGG16 to enhance accuracy for precise brain tumor

detection and classification.

1.4. Scope and Limitation

The scope of this research encompasses comprehensive preprocessing of medical images, systematic training of the CNN model on various available datasets, and thorough evaluation of its performance using well-established metrics including accuracy, precision, recall, and F1-score. This research emphasizes the transformative role of artificial intelligence in reducing human error and accelerating diagnostic processes, thereby significantly improving patient outcomes and healthcare efficiency.

However, this research acknowledges several important limitations. The accuracy and reliability of the system are inherently influenced by the quality, diversity, and representativeness of the dataset used for training the neural network. Given that the datasets employed in this research are limited to available MRI scans, the variability may not fully capture the complete spectrum of differences in tumor types, stages, or imaging conditions encountered in diverse clinical scenarios.

Furthermore, this model is developed primarily for academic research purposes and experimental validation. It should not be considered as a replacement for professional medical judgment or clinical expertise. Additional considerations include hardware computational constraints, model generalization capabilities across different populations, and the complex ethical implications surrounding the use of Artificial Intelligence (AI) in critical medical decision-making processes. These considerations extend beyond the immediate scope of this study.

1.5. Report Organization

This research report is systematically organized into several comprehensive chapters that present the research methodology, implementation details, and findings in a logical progression. The structure is as follows:

- **Chapter 1: Introduction** – Provides an overview of the research, outlining the background, objectives, significance, and scope of the study.
- **Chapter 2: Literature Review** – Provides an extensive review of existing approaches in brain tumor detection utilizing machine learning and deep learning techniques.
- **Chapter 3: System Analysis** – Presents detailed system analysis, including comprehensive requirement analysis and feasibility studies.
- **Chapter 4: System Design** – Covers the system design aspects, encompassing architectural framework decisions and model design considerations.

- **Chapter 5: Implementation and Testing** – Discusses implementation details and comprehensive testing procedures.
- **Chapter 6: Conclusion and Recommendations** – Concludes the research with key findings, practical recommendations, and directions for future research endeavors.

CHAPTER 2
ORGANIZATION DETAILS AND LITERATURE REVIEW

2.1. Introduction to Organization

2.2. Organizational Hierarchy

2.3. Working Domains of Organization

2.4. Description of Intern Department

2.5. Literature Review

CHAPTER 3
INTERNSHIP ACTIVITIES

3.1. Roles and Responsibilities

i. Functional Requirements

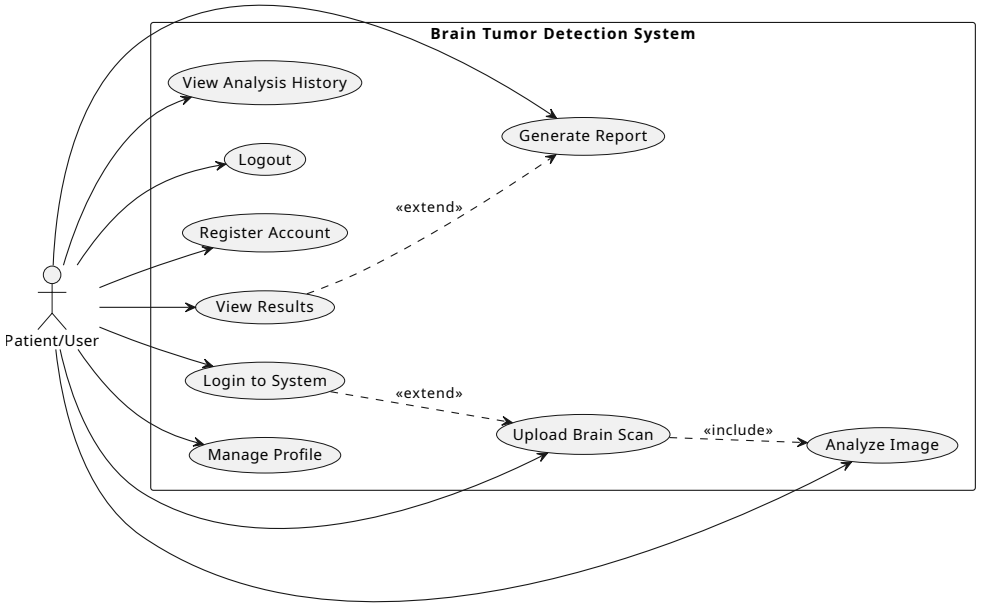

Figure 3.1: Use Case Diagram

The use case diagram presents the fundamental interactions between system actors and core functionalities of the application. The primary actor, Patient/User, represents individuals seeking brain tumor analysis services and can perform essential activities including account registration, system login, brain scan upload, image analysis, result viewing, report generation, profile management, analysis history review, and system logout. The diagram establishes key relationships through include and extend dependencies, where uploading brain scans necessarily includes the analysis process, viewing results can optionally extend to report generation, and successful login enables access to upload functionality.

ii. Non-Functional Requirements

- **Performance:** The system should process MRI images and provide classification results within 10–15 seconds per image, ensuring minimal waiting time for medical practitioners during diagnosis.
- **Accuracy:** The brain tumor classification model must maintain a minimum accuracy of 95% on test datasets, with precision and recall rates above 90% for each tumor type (glioma, meningioma, pituitary).
- **Usability:** The user interface must be intuitive and easy to navigate for medical practitioners with varying levels of technical expertise, requiring minimal training for effective system usage.
- **Reliability:** The system should maintain 99.5% uptime with robust error handling and recovery mechanisms to ensure continuous availability for critical medical applications.
- **Security:** Patient data and medical images must be encrypted during transmission and storage, with secure user authentication to comply with healthcare data protection regulations.
- **Compatibility:** The web-based interface must be compatible with major browsers (Chrome, Firefox, Safari, Edge) and responsive across desktop, tablet, and mobile devices.
- **Maintainability:** The codebase should follow clean coding practices and modular design principles to facilitate future updates, model improvements, and feature additions.
- **Data Integrity:** The system must ensure data consistency and prevent corruption during image upload, processing, and storage operations with appropriate backup and recovery mechanisms.
- **Extensibility:** The system architecture should support future enhancements such as additional imaging modalities (CT scans, X-rays), new tumor types, and integration with existing hospital information systems.

3.2. Weekly log

i. **Technical**

The system demonstrates strong technical viability through proven technology integration. The architecture combines React/TypeScript frontend with Node.js/Express backend, providing robust scalability for medical applications. TensorFlow/Keras implementation ensures reliable deep learning capabilities with 90% accuracy in brain tumor classification. Docker containerization enables consistent deployment across environments, while MongoDB offers flexible medical data storage. JWT authentication and bcrypt encryption meet healthcare security standards. The system supports cross-platform compatibility and future DICOM integration, confirming technical robustness and extensibility for medical imaging workflows.

ii. **Operational**

The system aligns effectively with existing healthcare workflows and user requirements. The

intuitive web-based interface requires minimal training for medical professionals, supporting multiple user roles including patients, doctors, and administrators. Automated analysis reduces interpretation time while maintaining human oversight through confidence scoring. The system integrates seamlessly with existing radiology workflows and generates comprehensive PDF reports for documentation. Cloud deployment supports telemedicine scenarios, while flexible architecture accommodates both individual practitioners and institutional deployments across various healthcare settings.

iii. **Economic**
The project demonstrates favorable cost-benefit ratios through open-source technology utilization, eliminating expensive licensing fees. Development costs remain manageable with estimated ROI within 18–24 months based on subscription models and pay-per-analysis pricing. The system addresses significant market demand for affordable medical imaging tools, particularly in resource-limited settings. Operational costs scale proportionally with adoption, while revenue potential exists through institutional subscriptions and premium features. Long-term sustainability benefits from cloud scaling and automated operations reducing maintenance overhead.

iv. **Schedule** The four-month development timeline from Jestha through Bhadra provides realistic project completion within allocated resources. Modular architecture enables parallel development streams, while agile methodology ensures continuous progress monitoring. The schedule accommodates comprehensive testing phases and risk mitigation strategies with adequate buffer time for unforeseen challenges. Critical milestones include system design completion, core functionality implementation, thorough testing, and final deployment preparation, ensuring successful delivery within the specified timeframe.

Task	**Jestha**				**Ashar**				**Shawarn**				**Bhadra**			
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4
Requirement Analysis	P	P														
Planning		P	P													
Design			P	P	P	P										
Iteration 1 — Dev					D	D	D									
Iteration 1 — Test								T								
Iteration 2 — Dev									D	D	D					
Iteration 2 — Test											T					
Iteration 3 — Dev											D	D	D			
Iteration 3 — Test													T			
Iteration 4 — Dev													D	D	D	
Iteration 4 — Test															T	
Deployment															Y	Y
Documentation (ongoing)	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C

Table 3.1: Gantt-style weekly schedule following iterative development. Shaded cells indicate active work.

3.3. Description of the Project Involved During Internship

The system is analyzed using the Object-Oriented Approach, which includes three aspects. Object Modelling with Class and Object Diagrams shows the static structure of the system. Dynamic Modelling with State and Sequence Diagrams explains object interactions and state changes. Process Modelling with Activity Diagrams illustrates the workflow and control flow of operations.

i. **Class Diagram**

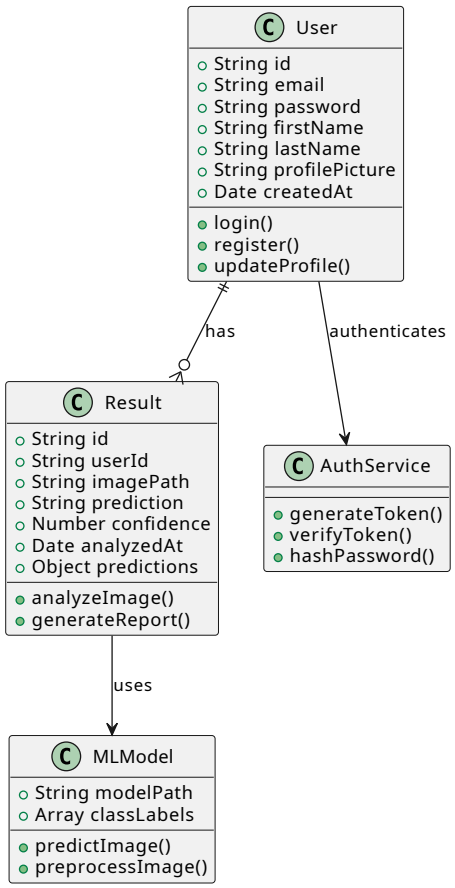

Figure 3.2: Class Diagram

The class diagram presents the fundamental building blocks of the brain tumor detection system. The User class serves as the central entity representing application users, containing basic authentication and profile information along with methods for login, registration, and profile updates. The Result class captures the outcomes of brain tumor analyses, storing prediction results, confidence scores, and timestamps while providing methods for image analysis and report generation. The MLModel class abstracts the machine learning functionality, encapsulating the model path and class labels with methods for image prediction and

preprocessing. The AuthService class handles security operations including token generation, verification, and password hashing. The relationships show that users can have multiple analysis results, results utilize the ML model for predictions, and users interact with the authentication service for security purposes.

ii. **Object Diagram**

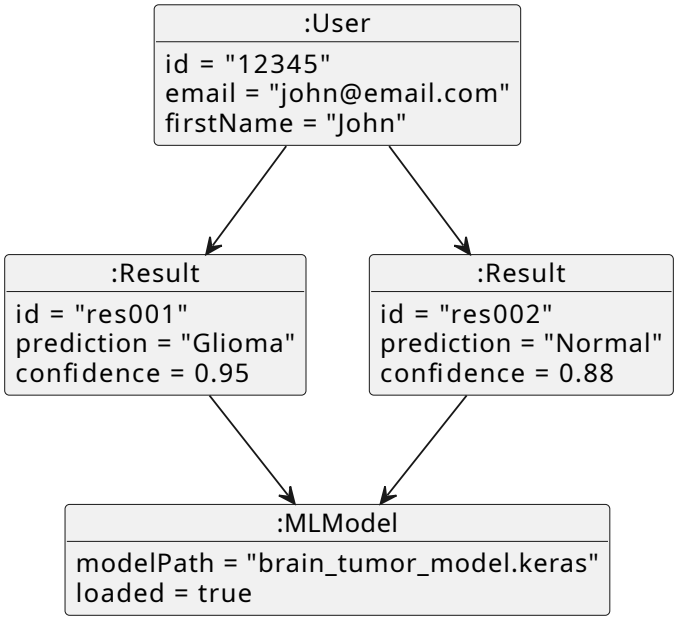

Figure 3.3: Object Diagram

This object diagram illustrates a specific runtime scenario where a user named John has performed brain tumor analyses. The diagram shows concrete instances of the classes, with User object "12345" representing John's account information, connected to two Result objects showing actual analysis outcomes – one detecting a Glioma tumor with 95% confidence and another showing a Normal scan with 88% confidence. Both results are linked to the same MLModel instance that has successfully loaded the Keras model file. This snapshot demonstrates how the system maintains relationships between users and their analysis history while sharing the ML model instance across multiple predictions.

iii. **State Diagram**

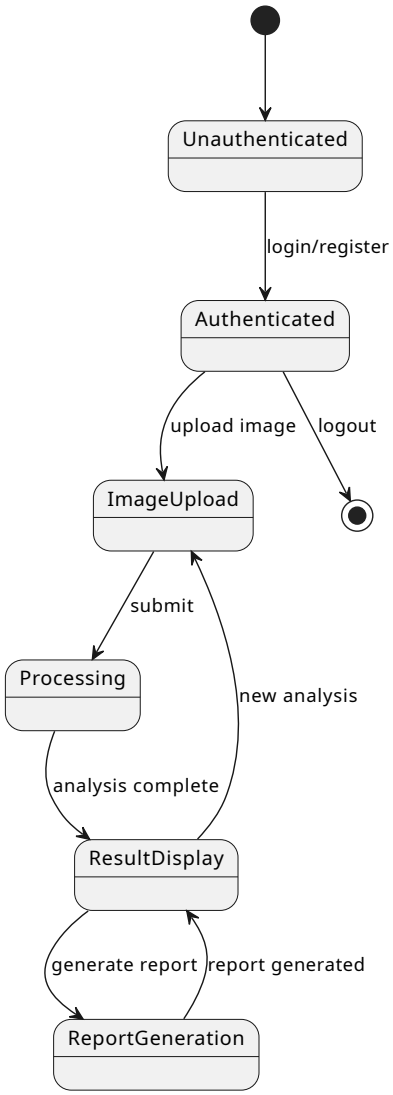

Figure 3.4: State Diagram

The state diagram traces the user journey through the application's main workflow states. Users begin in an unauthenticated state and transition to authenticated status through successful login or registration. Once authenticated, users can navigate to image upload functionality, where they submit brain scan images for analysis. The system then moves to a processing state where the ML model analyzes the uploaded image. Upon completion, users reach the result display state where they can view their analysis outcomes. From this state, users can either initiate new analyses by returning to image upload or generate detailed PDF reports. The diagram also shows the logout transition that returns users to the unauthenticated state, completing the application lifecycle. This workflow represents the essential data flow from image submission to result presentation.

iv. **Sequence Diagram**

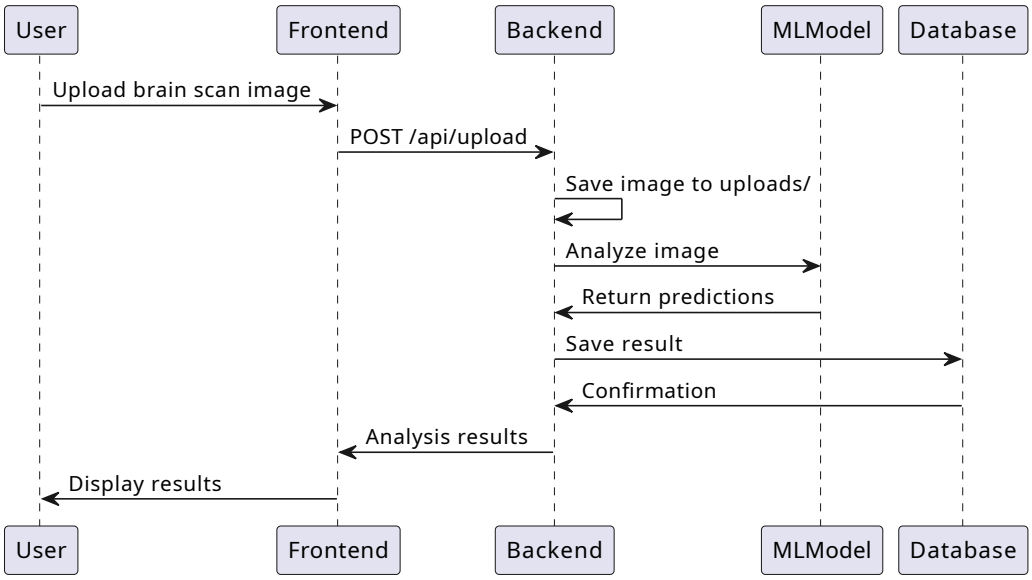

Figure 3.5: Sequence Diagram

This sequence diagram depicts the core interaction flow for brain tumor analysis. The process begins when a user uploads a brain scan image through the frontend interface. The frontend forwards this request to the backend server, which first saves the uploaded image to the designated uploads directory. The backend then invokes the ML model service to analyze the saved image, receiving prediction results including tumor classification and confidence scores. These results are subsequently stored in the database for future reference and user history. The backend returns the analysis results to the frontend, which presents them to the user in a comprehensible format. This workflow represents the essential data flow from image submission to result presentation.

v. **Activity Diagram**

The activity diagram outlines the decision-based workflow of the application. The process starts with authentication verification, directing unauthenticated users through the login process before proceeding. Once authenticated, users can upload images, which undergo validation to ensure they meet the system requirements for format and size. Valid images proceed to ML model processing, where the deep learning algorithm analyzes the brain scan for tumor detection. Successful analysis results are saved to the database and displayed to users. The workflow includes an optional branch for PDF report generation, allowing users to create downloadable documentation of their analysis results. Error handling is integrated throughout, redirecting users to appropriate error states when validation or processing fails.

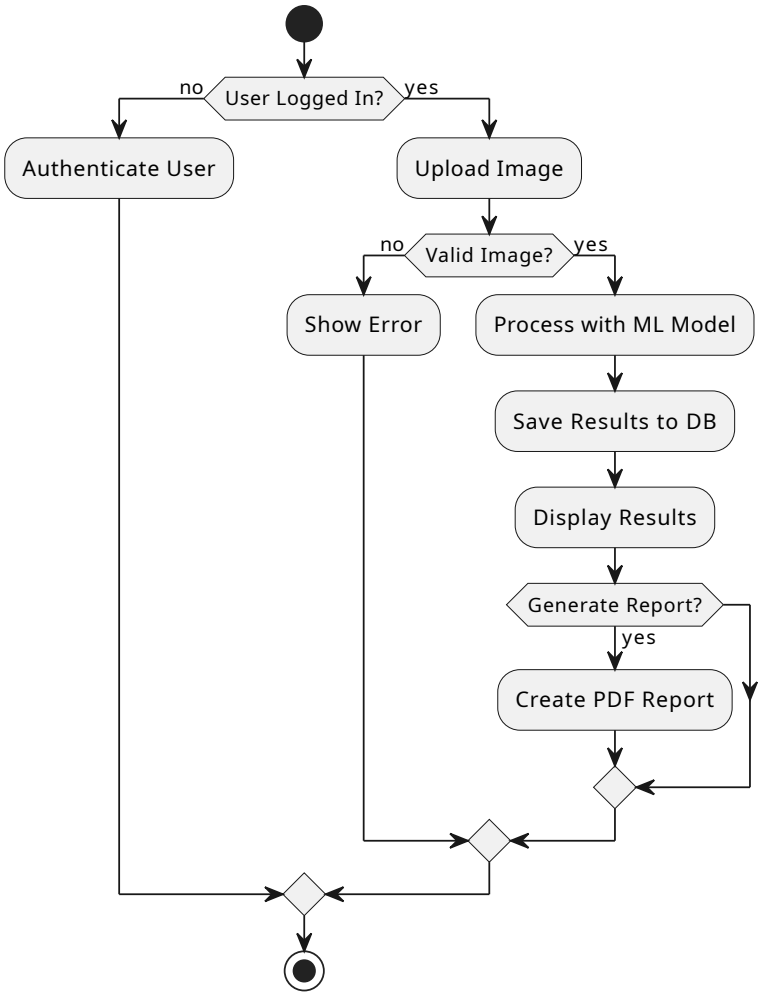

Figure 3.6: Activity Diagram

3.4. Activities Performed

CHAPTER 4
CONCLUSION AND LEARNING OUTCOMES

4.1. Conclusion

System design takes the findings from analysis and converts them into a practical plan for development. It describes the system's architecture, components, data flow, and deployment environment. The design ensures that each part of the system frontend, backend, database, and machine learning model works together smoothly. Diagrams like class, sequence, and deployment diagrams help visualize the structure and workflow, making implementation easier and more reliable.

i. **Class Diagram**

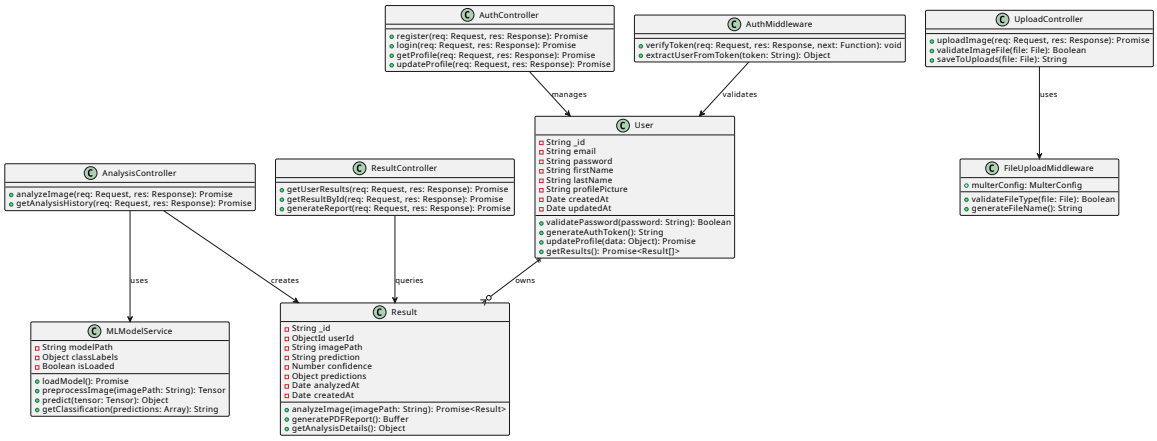

Figure 4.1: Refinement of Class Diagram

The refined class diagram significantly expands upon the high-level version by introducing detailed implementation specifics and architectural components. While the original diagram focused on core domain entities, this version incorporates the complete MVC architecture with dedicated controller classes for different functional areas. The AuthController, UploadController, AnalysisController, and ResultController represent the application's API endpoints and business logic handlers. New middleware components like AuthMiddleware and FileUploadMiddleware demonstrate the request processing pipeline and security layers. The MLModelService class provides more granular methods for model loading, image preprocessing, tensor operations, and classification logic. Database models now include private fields with proper encapsulation and detailed method signatures showing parameter types and return values. This refined diagram reveals the actual software architecture with separation of

concerns, dependency injection patterns, and proper abstraction layers that weren't visible in the high-level overview.

ii. **Object Diagram**

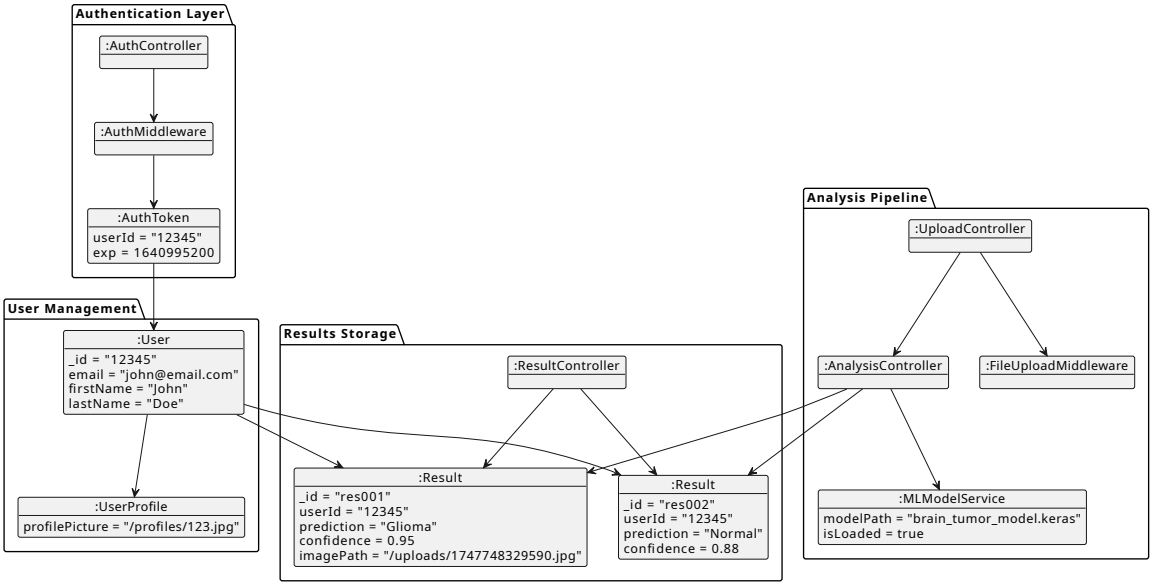

Figure 4.2: Refinement of Object Diagram

The refined object diagram presents a more comprehensive runtime view by organizing components into logical packages representing different architectural layers. Unlike the simple high-level version, this diagram shows the Authentication Layer with controller instances, middleware objects, and JWT token representations. The User Management package demonstrates the relationship between user entities and their profile information. The Analysis Pipeline package reveals the complete request processing flow from upload controllers through file middleware to ML services. The Results Storage package shows how multiple result instances are managed by dedicated controllers. This detailed view exposes the actual object collaboration patterns, package dependencies, and layer interactions that occur during system execution, providing insights into the implementation architecture that the high-level diagram abstracted away.

iii. **State Diagram**

The refined state diagram transforms the simple linear workflow into a comprehensive state machine with nested states and complex transitions. The original diagram showed basic authentication and analysis states, while this version introduces hierarchical state organization with distinct flows for authentication, image analysis, and profile management. The AuthenticationFlow composite state includes detailed substates for token checking, form validation, and error handling.

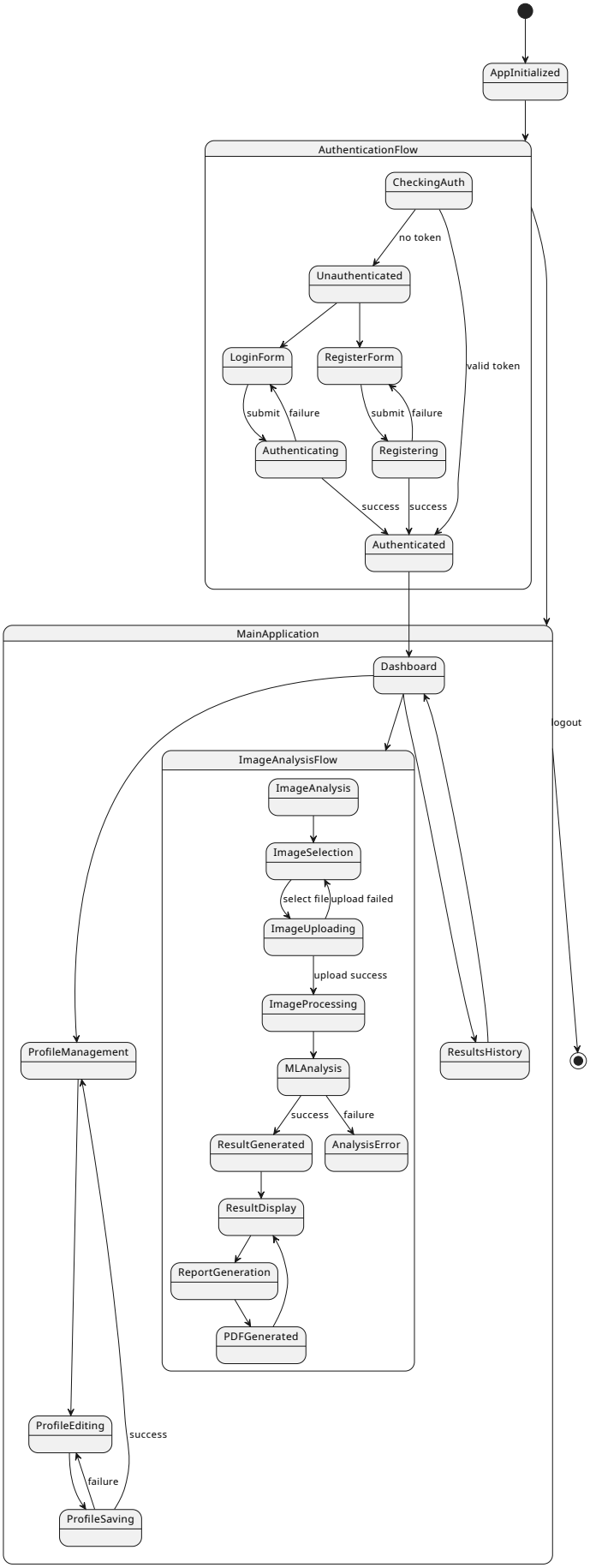

Figure 4.3: Refinement of State Diagram

The ImageAnalysisFlow demonstrates the complete image processing pipeline with states for selection, uploading, processing, ML analysis, result generation, and error recovery. Additional states for profile management and report generation show parallel user workflows. This refined diagram captures real-world application complexity with multiple concurrent processes, error states, and recovery mechanisms that provide a complete picture of system behavior under various conditions.

iv. **Sequence Diagram**

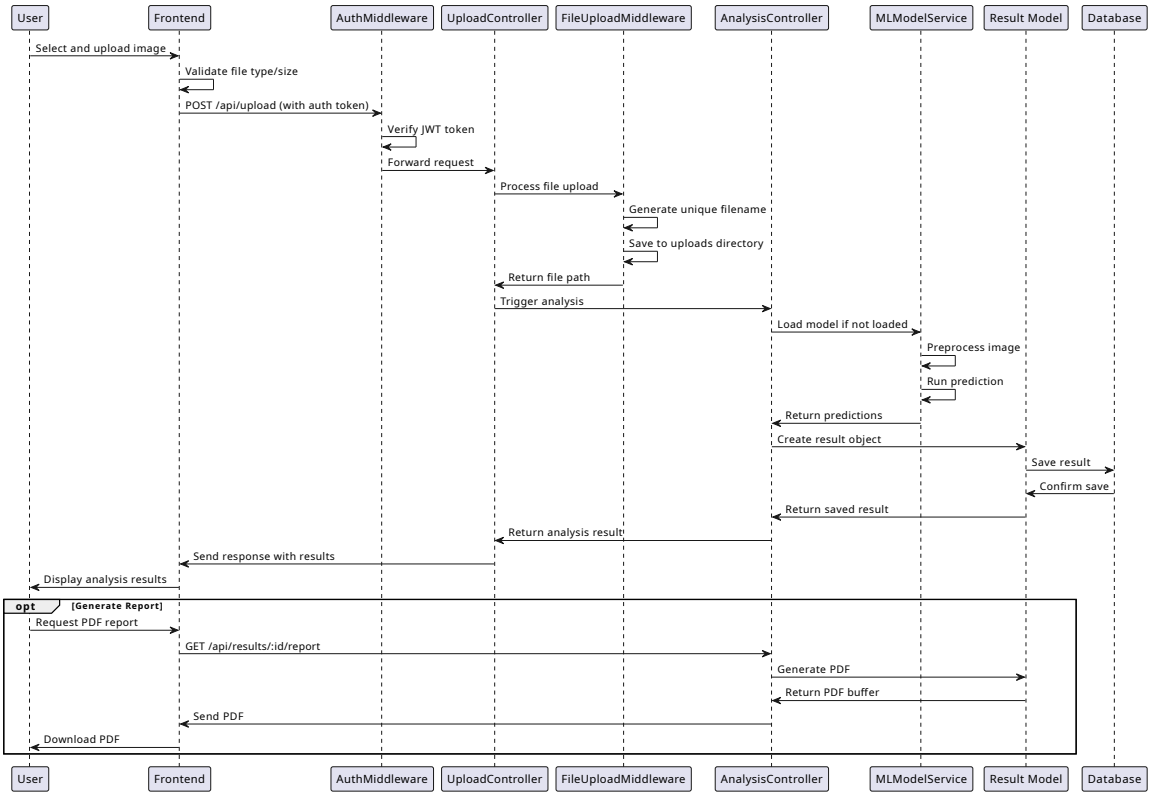

Figure 4.4: Refinement of Sequence Diagram

The refined sequence diagram expands the simple high-level interaction into a detailed multi-participant collaboration showing the complete request processing pipeline. While the original diagram showed basic frontend-backend-model communication, this version reveals the actual middleware stack, authentication verification, file processing layers, and database operations. The diagram now includes AuthMiddleware for token verification, FileUploadMiddleware for file handling, and separate controllers for different responsibilities. Additional detail shows model loading optimization, image preprocessing steps, database confirmation patterns, and optional PDF generation workflows. Error handling scenarios and alternative flows are represented through optional fragments. This comprehensive view demonstrates the

actual software architecture with proper separation of concerns, middleware processing, and the complete data transformation pipeline from user input to final output.

v. **Activity Diagram**

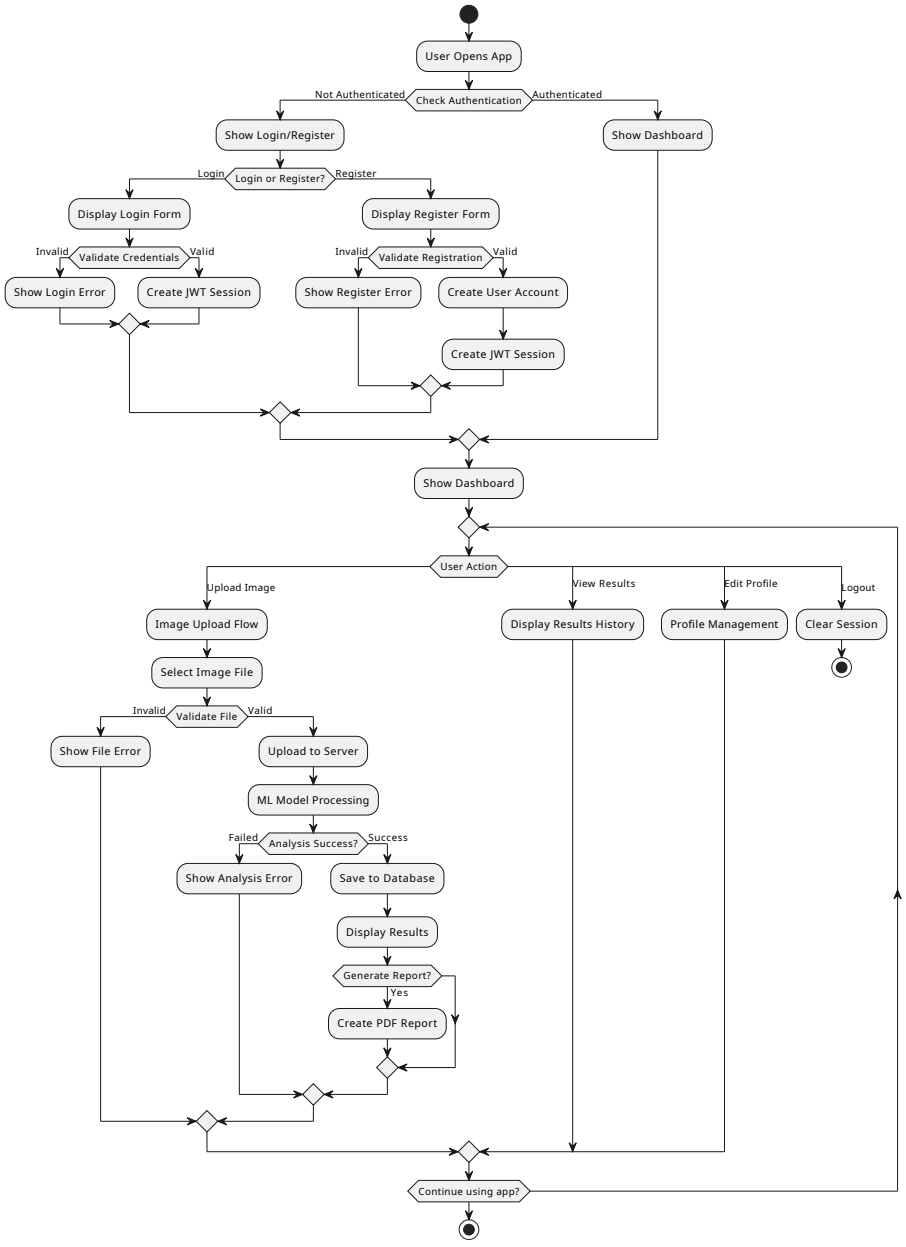

Figure 4.5: Refinement of Activity Diagram

The refined activity diagram transforms the basic decision flow into a comprehensive business process model with detailed error handling and multiple workflow branches. The high-level diagram showed simple authentication and analysis paths, while this version incorporates complete user registration flows, detailed validation processes, and comprehensive error re-

covery mechanisms. New branches handle user account creation, profile management workflows, and multiple analysis result viewing options. The diagram now shows parallel processes for different user actions, detailed file validation steps, and granular ML processing phases. Error states are properly connected to recovery paths, and the workflow includes session management, logout procedures, and application lifecycle management. This detailed process model reflects the actual user experience with all possible paths, decision points, and system responses that users encounter during real application usage.

vi. **Component Diagram**

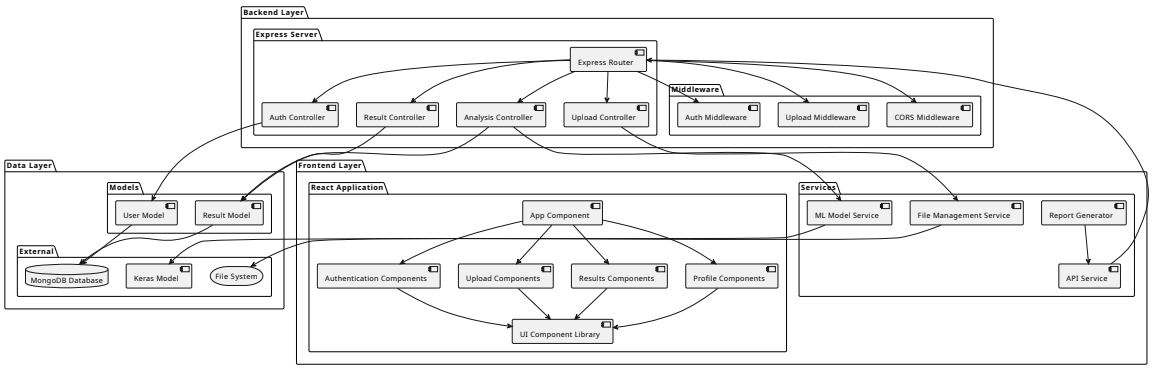

Figure 4.6: Refinement of Component Diagram

The component diagram presents the system's architectural organization across three distinct layers, demonstrating how components collaborate to deliver application functionality. The Frontend Layer contains the React application with its constituent components for authentication, file upload, results display, and profile management, all built upon a shared UI component library. The API Service and Report Generator provide client-side services for server communication and document generation. The Backend Layer implements the Express server architecture with route handlers, controllers for different functional domains, and middleware components for cross-cutting concerns like authentication, file upload processing, and CORS handling. Business logic services handle ML model operations and file management. The Data Layer abstracts database operations through Mongoose models while connecting to external systems including MongoDB for data persistence, the file system for image storage, and the Keras model for machine learning predictions. This organization demonstrates clear separation of concerns with defined interfaces between layers.

vii. **Deployment Diagram**

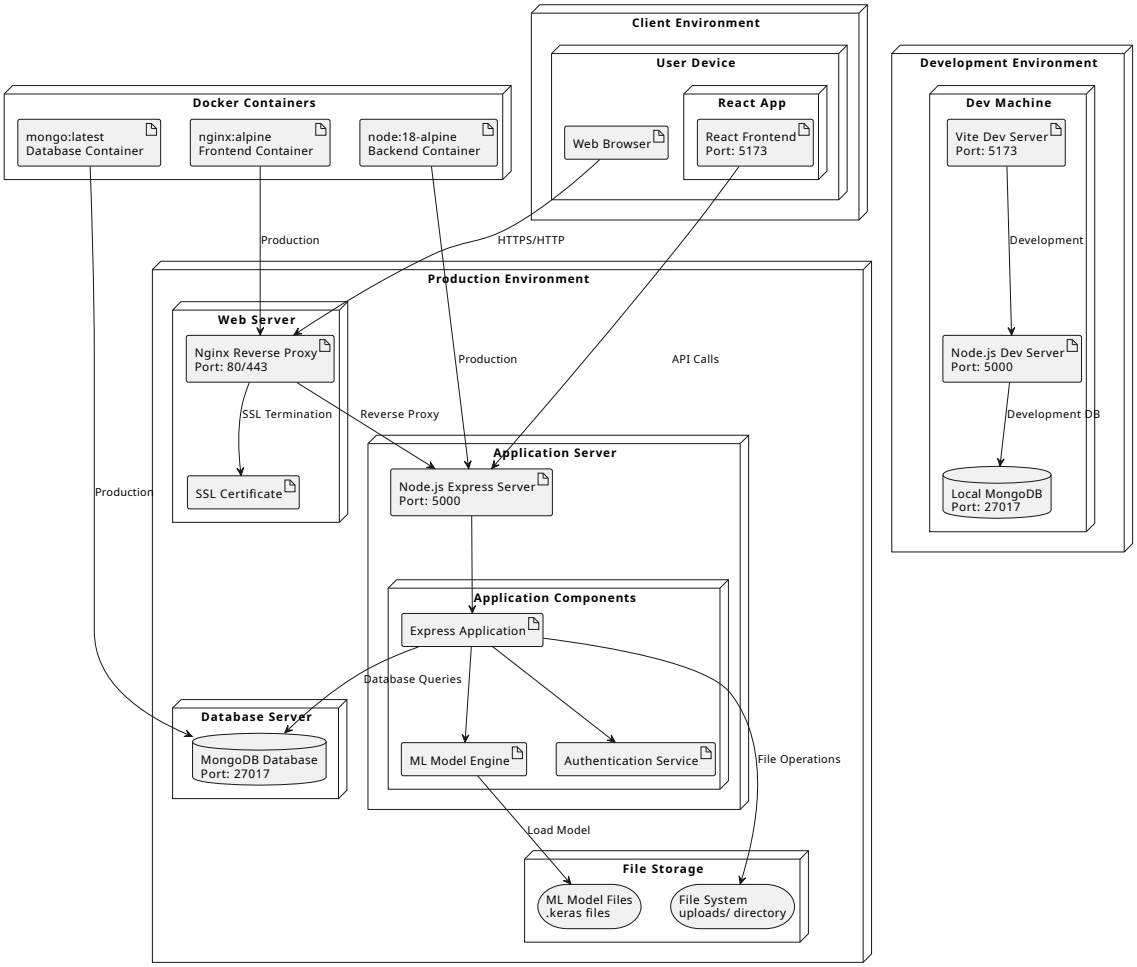

Figure 4.7: Refinement of Deployment Diagram

The deployment diagram illustrates the complete system infrastructure across development, production, and containerized environments. The Client Environment shows end-user devices running web browsers that host the React frontend application. The Production Environment demonstrates a multi-tier architecture with Nginx serving as a reverse proxy and SSL termination point, forwarding requests to the Node.js application server. The application server hosts the Express framework with integrated ML model engine and authentication services, connecting to a dedicated MongoDB database server and file storage systems for uploads and model files. The Development Environment mirrors production with local development servers and database instances for testing. Docker containerization provides deployment flexibility with separate containers for frontend, backend, and database components. This infrastructure supports scalability, security, and maintainability while providing development-production parity through containerized deployments that ensure consistent environments across different deployment targets.

4.2. Learning Outcome

Convolutional Neural Network (CNN)

A CNN is a type of artificial neural network specifically designed for processing and classifying visual data. In this project, we developed a custom CNN architecture tailored for medical imaging.

The primary algorithm powering the brain tumor detection system is a Convolutional Neural Network implemented using TensorFlow/Keras framework. This deep learning algorithm employs multiple convolutional layers with ReLU activation functions, pooling layers for dimensionality reduction, and fully connected dense layers for final classification [4]. The CNN automatically learns hierarchical features from brain MRI images, starting with simple edge detection in early layers and progressing to complex anatomical structures in deeper layers. The network architecture includes batch normalization for training stability [5], dropout layers for regularization to prevent overfitting [6], and a softmax output layer that produces probability distributions across four classes: Glioma, Meningioma, Pituitary tumors, and Normal brain scans. The algorithm processes 224× 224 pixel RGB images and outputs confidence scores for each tumor category, enabling medical professionals to assess the reliability of predictions alongside the primary classification result.

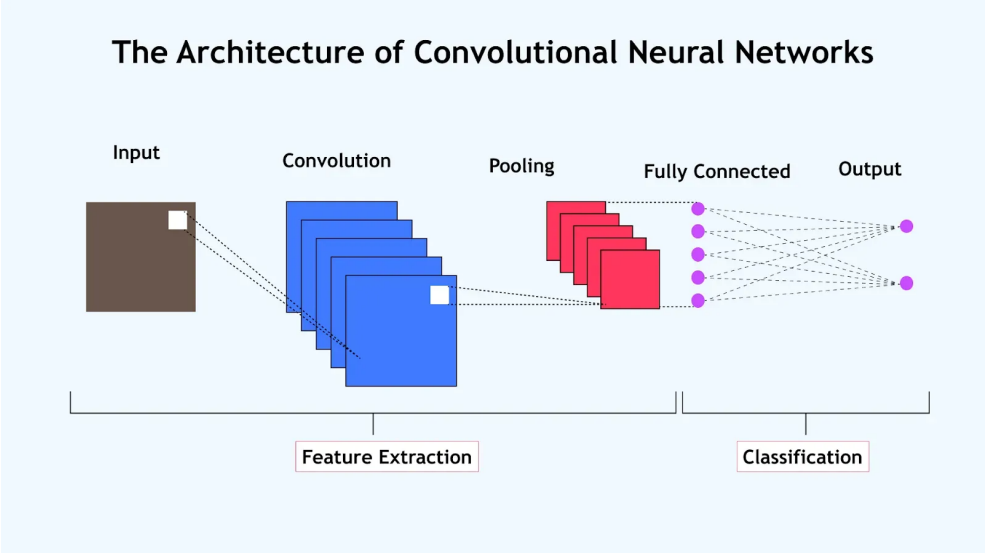

Figure 4.8: CNN Architecture

Transfer Learning with Visual Geometry Group 16-layer model (VGG16)

To complement the custom CNN and further enhance accuracy, we integrate a transfer learning approach using the VGG16 architecture pretrained on the ImageNet dataset. VGG16 is a well-established deep network known for its simplicity and performance in image classification tasks.

In our approach, the VGG16 base model is used without its top classification layers. All convolutional layers of VGG16 are frozen to retain the pretrained features, which are known to capture

generic visual patterns that are also relevant to medical images.

On top of the VGG16 base, we add a Global Average Pooling layer followed by fully connected Dense layers with Rectified Linear Unit (ReLU) activations and Dropout. The final classification layer uses softmax activation to output probabilities for each class. This hybrid model combines the general visual understanding of VGG16 with task-specific learning layers adapted to brain tumor detection.

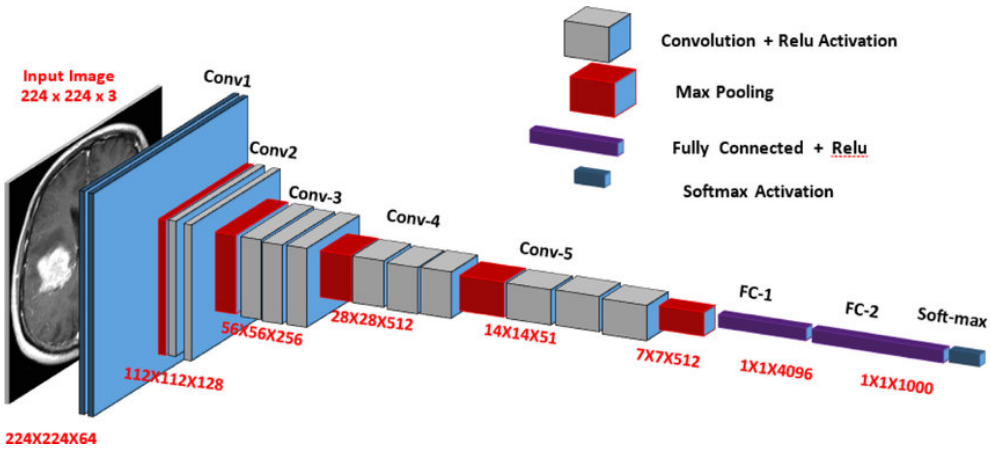

Figure 4.9: VGG16 Architecture

JSON Web Token (JWT) Authentication Algorithm

The stateless authentication mechanism employs JSON Web Tokens to manage user sessions and API access control [7]. The JWT algorithm creates digitally signed tokens containing user identity information and access permissions, typically using Hash-based Message Authentication Code with Secure Hash Algorithm 256-bit (SHA256) or Rivest–Shamir–Adleman (RSA) algorithms for signature generation [7]. Upon successful login, the server generates a JWT containing the user's ID, email, role, and expiration timestamp, signing it with a secret key to ensure its integrity and authenticity. The algorithm enables distributed authentication without requiring server-side session storage, as each token carries sufficient information for access verification. Token validation involves signature verification using the same secret key, expiration time checking, and payload extraction for user identification. The implementation includes automatic token refresh mechanisms and secure transmission protocols to maintain session security while providing seamless user experience across multiple application components.

Bcrypt Password Hashing Algorithm

The authentication security relies on the bcrypt algorithm for secure password storage and verification [8]. This adaptive hashing function incorporates a configurable salt parameter that increases computational complexity and prevents rainbow table attacks. The algorithm generates unique salt values for each password, combines them with the plaintext password, and applies multiple rounds of hashing using the Blowfish cipher [8]. The system implements a minimum of 12 salt rounds,

creating a computationally expensive operation that significantly slows down brute-force attacks while remaining efficient for legitimate authentication requests. The bcrypt algorithm automatically handles salt generation and storage within the hash output, enabling seamless password verification during user login attempts while maintaining resistance against timing attacks and providing forward security as computational power increases over time.

REFERENCES

[1] M. Razzak, S. Naz, and A. Zaib, "Deep learning for medical image processing: Overview, challenges and the future," in *Classification in BioApps*, ser. Lecture Notes in Computational Vision and Biomechanics, N. Dey, A. Ashour, and S. Borra, Eds. Springer, Cham, 2018, vol. 26.

[2] I. Ilic and M. Ilic, "International patterns and trends in the brain cancer incidence and mortality: An observational study based on the global burden of disease," *Heliyon*, vol. 9, no. 7, p. e18222, 2023.

[3] S. Aksoy, "Brain tumor detection and classification via vgg16-based deep learning on mri imaging," in *Proceedings of [Conference Name]*, 2025, under review.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[5] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456, 2015.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[7] JWT.io, "Introduction to json web tokens," 2024, accessed: 2025-09-03. [Online]. Available: https://www.jwt.io/introduction#what-is-json-web-token

[8] GeeksforGeeks, "Npm bcrypt," 2024, accessed: 2025-09-03. [Online]. Available: https://www.geeksforgeeks.org/node-js/npm-bcrypt/

APPENDIX

Table 4.1: Meeting log with supervisor detailing project discussions and feedback.

Date	Time	Purpose of Visit	Discussion / Notes
2025-05-15	11:00 AM	Project proposal discussion	Defined scope of the brain tumor classification project. Supervisor suggested focusing on MRI dataset selection, preprocessing, and evaluation metrics.
2025-06-05	1:30 PM	Data preprocessing review	Reviewed MRI image preprocessing pipeline (normalization, augmentation, resizing). Supervisor approved approach and recommended testing multiple augmentation strategies.
2025-06-27	11:00 AM	Model architecture discussion	Discussed CNN-based model architecture for tumor classification. Supervisor advised experimenting with transfer learning and incorporating dropout for regularization.
2025-07-24	11:00 AM	Training and validation	Presented initial training results with accuracy and loss curves. Supervisor highlighted the need to address overfitting and suggested cross-validation with multiple folds.
2025-09-23	1:30 PM	Final review	Reviewed final model performance including confusion matrix, classification report, and visualizations. Supervisor approved results and recommended preparing documentation and final report.

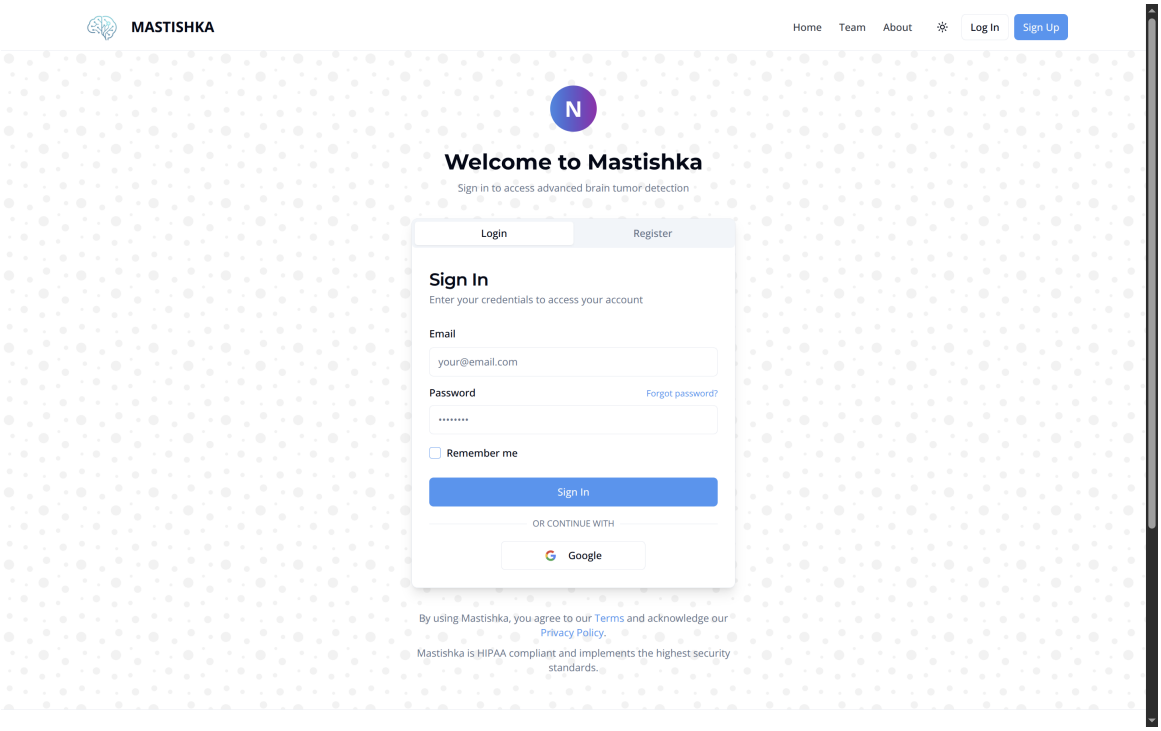

Figure 4.10: Login page of the web application

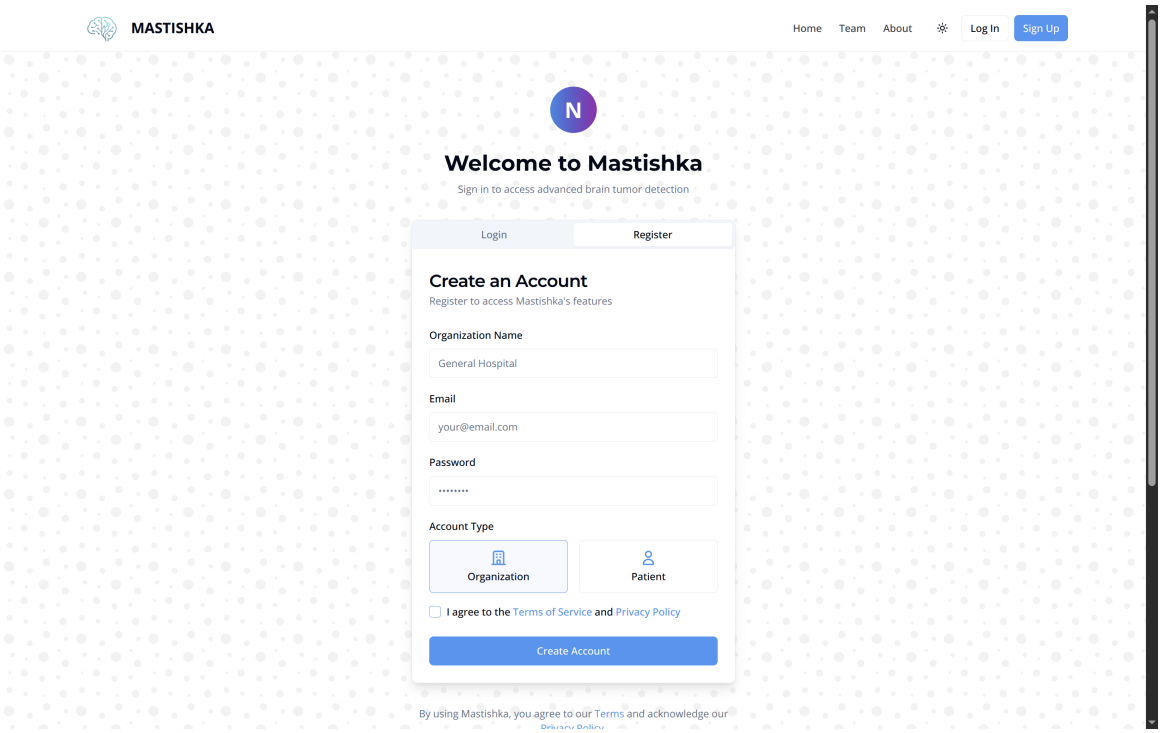

Figure 4.11: Sign in page of the web application

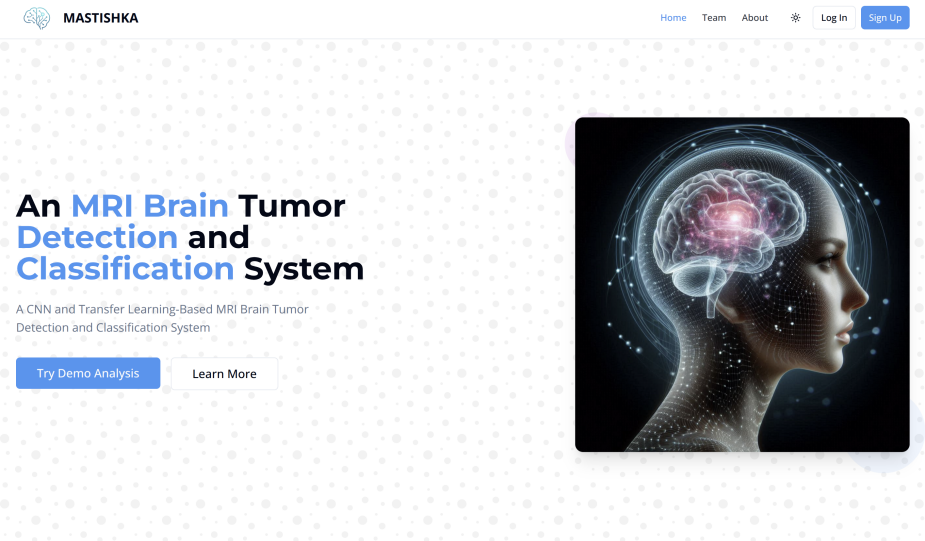

Figure 4.12: Home page of the web application

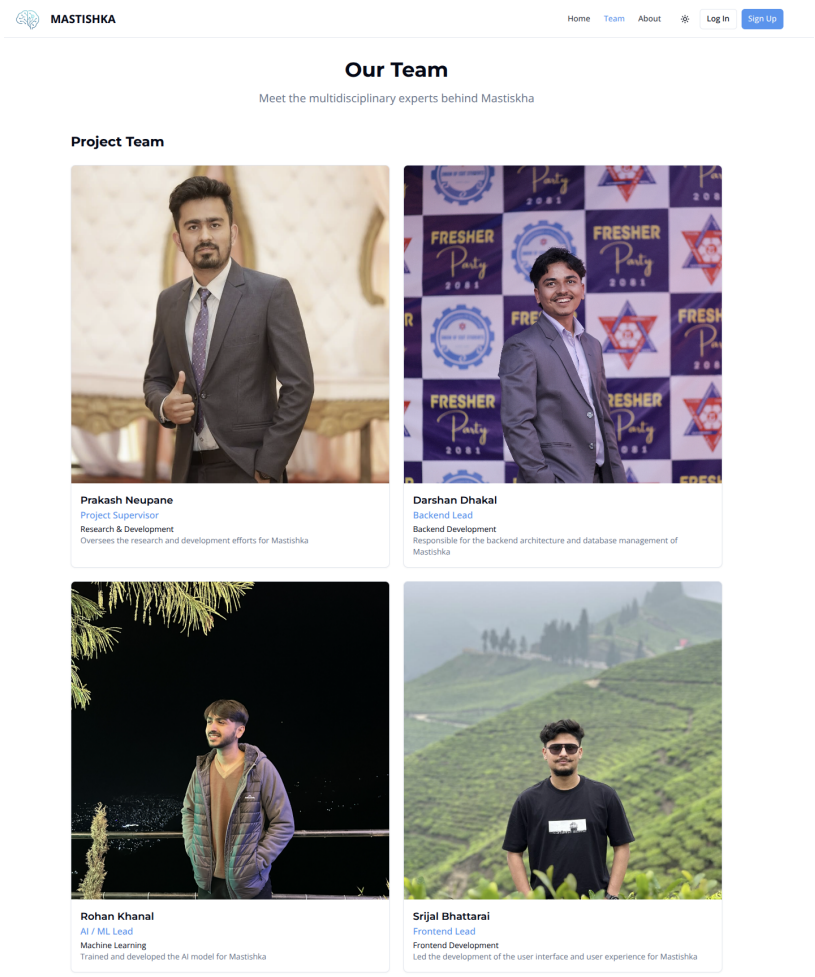

Figure 4.13: Team page of the web application

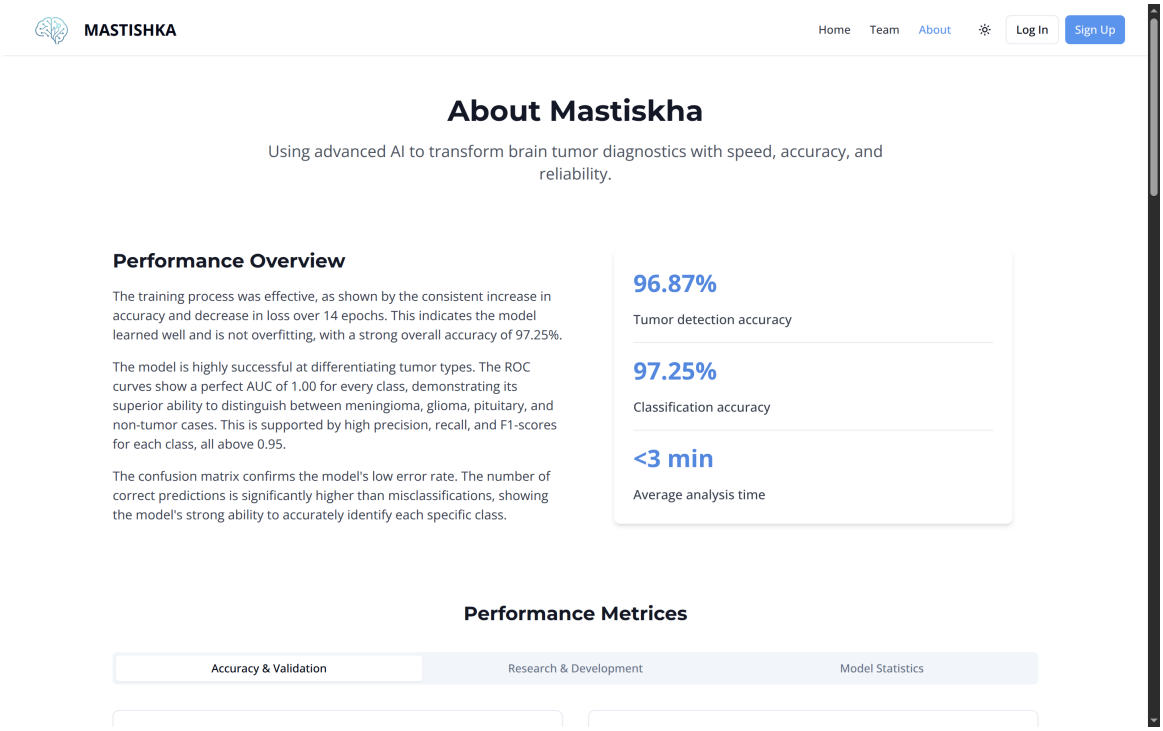

Figure 4.14: Statistics page of the web application

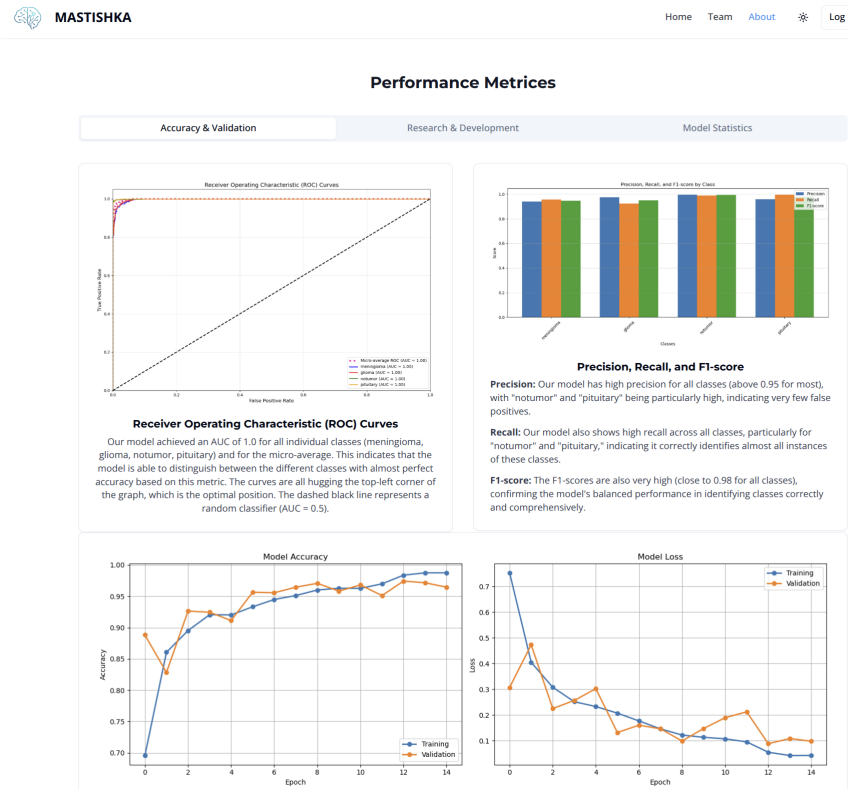

Figure 4.15: Statistics page of the web application

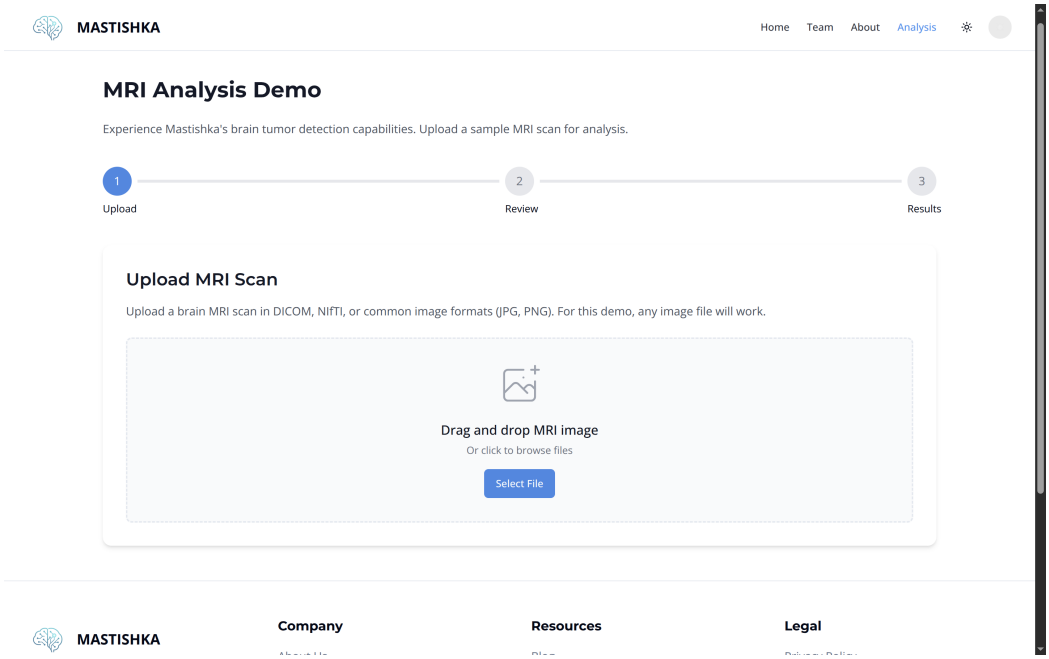

Figure 4.16: Analysis page of the web application

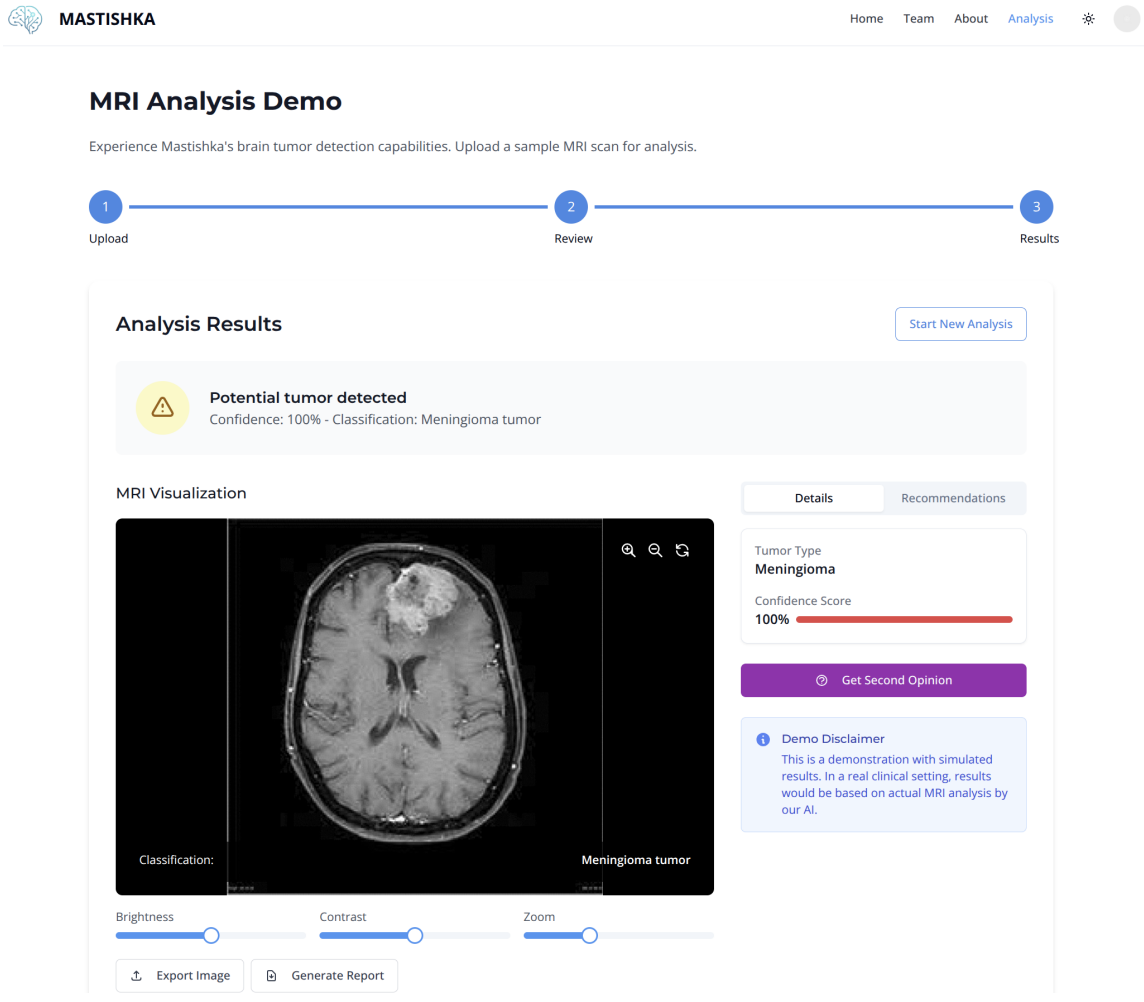

Figure 4.17: Analysis page of the web application

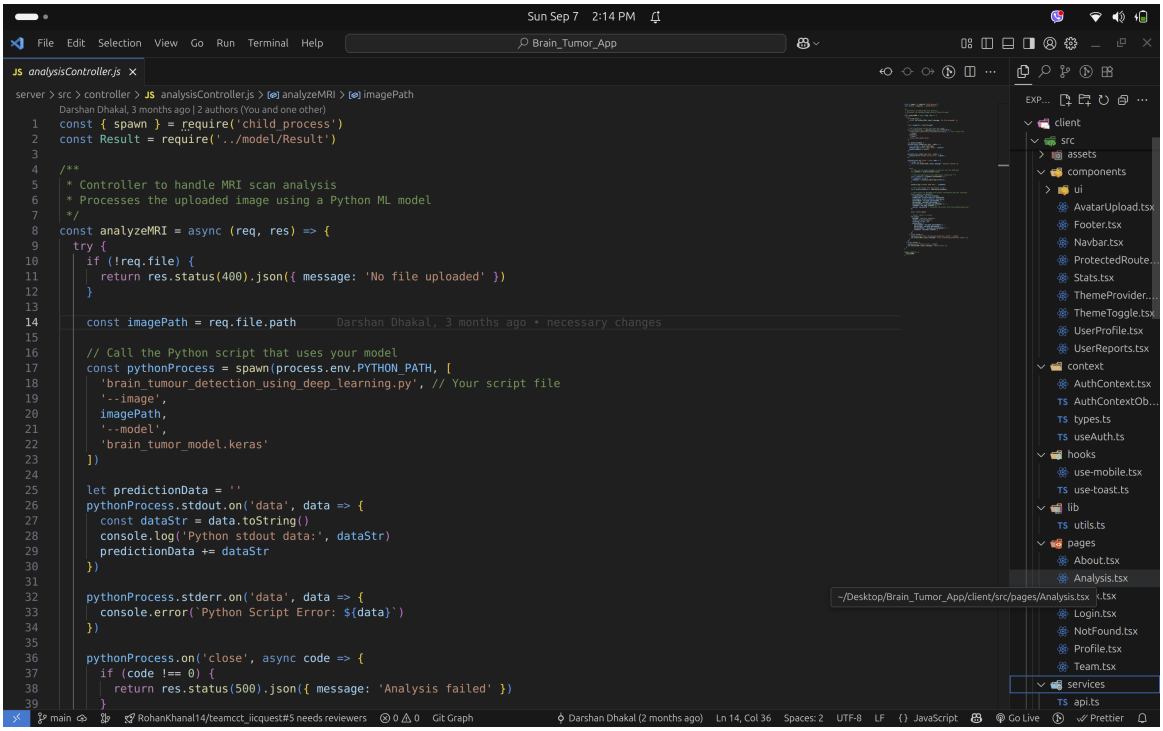

Figure 4.18: Frontend File Structure of the web application

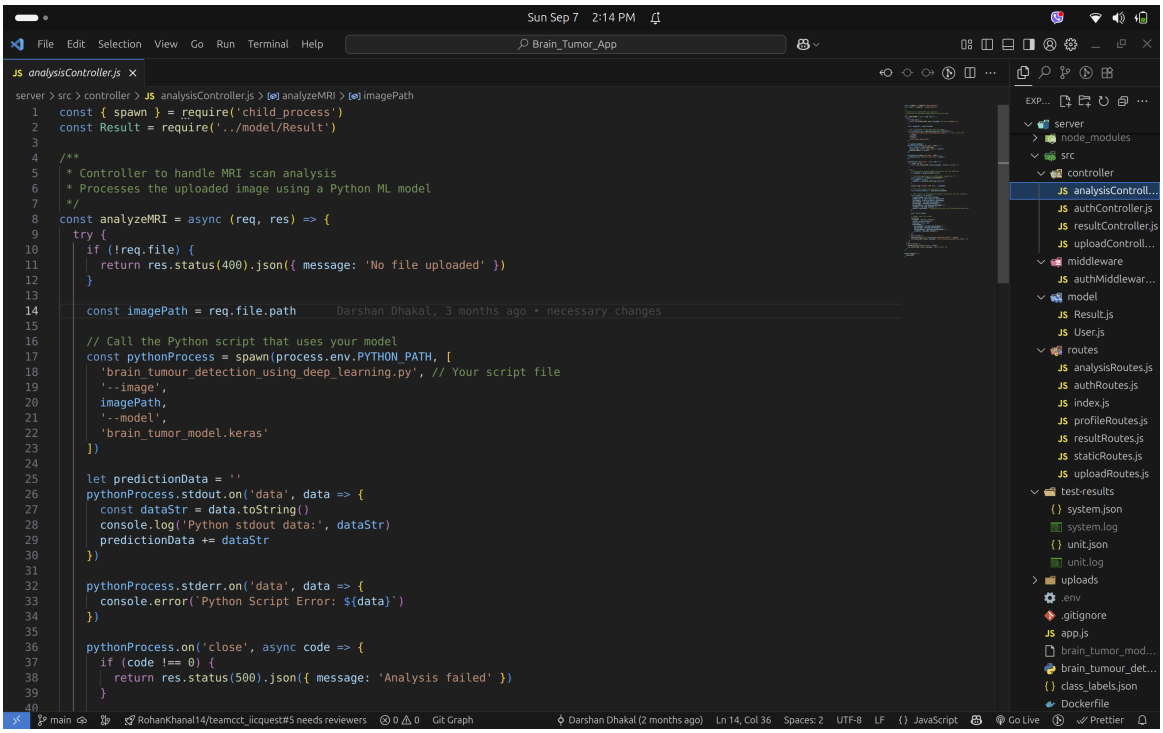

Figure 4.19: Backend File Structure of the web application

page of the web application