VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi - 590018

Project Report on

"Machine Learning-Based Early Detection of Autism In Children"

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE & ENGINEERING

By

MILIYANA REENA DSOUZA	4MT20CS098
SAMRUDDHI H.R	4MT21CS131
SANYUKTHA SHETTY	4MT21CS145
SHRADDHA SINGH	4MT21CS149

Under the Guidance of
Dr. Rejeesh Rayaroth
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

(A Unit of Rajalaxmi Education Trust ®, Mangalore)

Autonomous Institute Affiliated to V.T.U, Belagavi , Approved by AICTE, New Delhi

Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

2024-25

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

(A Unit of Rajalaxmi Education Trust ®, Mangalore)

Autonomous Institute Affiliated to V.T.U, Belagavi , Approved by AICTE, New Delhi

Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MITE

Invent Solutions

CERTIFICATE

Certified that the project work entitled **“Machine Learning-Based Early Detection Of Autism In Children”** carried out by **Ms. MILIYANA REENA DSOUZA, USN:4MT20CS098,** a Bonafide student of **Mangalore Institute of Technology & Engineering** in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi**, during the year **2024-25**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project Work prescribed for the said Degree.

Signature of the Guide	Signature of the HOD	Signature of the Principal
Dr. Rejeesh Rayaroth	**Dr. Ravinarayana B**	**Dr. Prashanth C M**
Name of the Examiners	**External Viva**	**Signature with Date**

1.

2.

MANGALORE INSTITUTE OF TECHNOLOGY & ENGINEERING

(A Unit of Rajalaxmi Education Trust ®, Mangalore)

Autonomous Institute Affiliated to V.T.U, Belagavi , Approved by AICTE, New Delhi

Accredited by NAAC with A+ Grade & ISO 9001:2015 Certified Institution

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MITE

Invent Solutions

DECLARATION

I, **MILIYANA REENA DSOUZA(4MT20CS098)** student of 7th semester BE in Computer Science & Engineering, **Mangalore Institute of Technology and Engineering, Moodabidri,** hereby declare that the project work entitled **"Machine Learning-Based Early Detection Of Autism In Children"**, submitted to the Visvesvaraya Technological University, Belagavi during the academic year **2024-25,** is a record of an original work done by me under the guidance of **Dr. Rejeesh Rayaroth,** Assistant Professor,Department of Computer Science & Engineering , Mangalore Institute of Technology and Engineering, Moodabidri. This project work is submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree**.**

Date:
Place: Moodabidri

MILIYANA REENA DSOUZA

ABSTRACT

This project aims to analyze and predict autism spectrum disorder (ASD) diagnosis using machine learning techniques. ASD is a developmental disorder that affects communication, behavior, and social interactions. Early diagnosis is crucial to provide the necessary interventions for individuals with autism. The dataset used in this study contains various features, such as age, gender, and behavioral traits, which are analyzed to identify patterns associated with ASD. Convolutional Neural Networks (CNN) is utilized for image-based data analysis, particularly for facial feature extraction, which enhances the predictive accuracy of the models. The performance of these models is evaluated based on accuracy, precision, recall, and other metrics to determine their effectiveness in real-world scenarios. Additionally, data preprocessing techniques like handling missing values and feature selection are applied to improve model performance. The project also explores the ethical considerations of using machine learning in healthcare, ensuring that the models are interpretable and reliable for medical applications. Overall, the project demonstrates the potential of AI in aiding early autism diagnosis and contributing to better healthcare outcomes.

ACKNOWLEDGEMENT

The satisfaction and the successful completion of this project would be incomplete without the mention of the people who made it possible, whose constant guidance encouragement crowned my efforts with success.

This project is made under the guidance of **Dr. Rejeesh Rayaroth**, **Assistant Professor,**in the Department of Computer Science and Engineering. I would like to express my sincere gratitude to my guide for all the helping hand and guidance in this project.

I would like to thank my project coordinator **Mr. Vijayananda V Madlur, Assistant Professor** in the Department of Computer Science and Engineering, for his cordial support, valuable information and guidance, which helped me in completing this project through the various stages.

I would like to express appreciation to **Dr. Ravinarayana B, Professor and Head**, Department of Computer Science and Engineering, for his support and guidance.

I would like to thank our Principal **Dr. Prashanth C M**, for encouraging me and giving me an opportunity to accomplish the project.

I also thank our management who helped me directly or indirectly in the completion of this project.

My special thanks to faculty members and others for their constant help and support.

Above all, I extend my sincere gratitude to my parents and friends for their constant encouragement with moral support.

MILIYANA REENA DSOUZA(4MT20CS098)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter – 1
INTRODUCTION

CHAPTER 1

INTRODUCTION

Autism Spectrum Disorder (ASD) is a neurodevelopmental condition that affects an individual's ability to communicate, interact socially, and engage in repetitive behaviors. It presents in early childhood, with symptoms varying in severity, making early diagnosis essential for effective intervention and support. Traditional methods for diagnosing autism are often time-consuming, subjective, and resource-intensive, which can delay treatment for children who need early interventions.

In recent years, advancements in machine learning have provided promising opportunities to enhance the early detection of ASD. By analyzing behavioral, demographic, and other clinical data, machine learning algorithms can identify patterns that may not be immediately evident to healthcare professionals. This project leverages data from Kaggle, featuring a dataset containing various attributes linked to ASD traits, to build predictive models for identifying children at risk of autism. Through this research, we aim to create a reliable, efficient, and scalable diagnostic tool and using Convolutional Neural Networks (CNN). These models are trained to assess the likelihood of ASD based on key indicators from the dataset. The goal is to assist healthcare providers in making informed decisions about potential ASD diagnoses, thus improving early intervention outcomes for children affected by the disorder. This project not only demonstrates the practical application of AI in healthcare but also emphasizes the importance of accuracy, interpretability, and ethical considerations when using machine learning in sensitive medical contexts.

1.1 Problem Statement

Autism Spectrum Disorder (ASD) is a developmental condition that significantly affects a child's ability to communicate, interact socially, and adapt to changing environments. Early diagnosis and intervention are critical for improving developmental outcomes, yet traditional diagnostic methods present several challenges. These methods often involve lengthy procedures, subjective assessments, and reliance on specialized expertise that may not be available in all regions, particularly in underserved or rural areas.

This creates delays in identifying ASD, resulting in missed opportunities for timely intervention during the critical early years of brain development when therapeutic efforts are most effective. The lack of early diagnosis can negatively impact a child's cognitive, emotional, and social growth, leading to long-term challenges for the individual and their families.

In light of these challenges, leveraging advancements in machine learning and artificial intelligence can address the inefficiencies of traditional diagnostic processes. By developing an efficient, data-driven diagnostic tool, this project seeks to identify children at risk for ASD at an earlier stage. Such a system can analyze behavioral patterns, responses, or image data with higher accuracy and speed than traditional methods, reducing dependence on subjective judgment.

The proposed solution aims to make ASD diagnosis more accessible and reliable, particularly for communities lacking specialized healthcare providers. Early detection through this approach can significantly enhance the chances of effective treatment, support, and personalized care, ultimately improving the quality of life for children with ASD and their families.

1.2 Objectives

The primary goal of this project is to develop an advanced machine learning model to detect Autism Spectrum Disorder (ASD) efficiently and accurately. Leveraging Convolutional Neural Networks (CNN) for image analysis, the model will analyze various data features to identify subtle indicators of ASD. By integrating advanced preprocessing techniques and data augmentation, the model’s robustness and ability to generalize across diverse datasets will be significantly enhanced, ensuring reliable performance in varied conditions.

To ensure usability, a user-friendly interface will be created, enabling seamless interaction for both healthcare professionals and parents. This interface will make the model practical and accessible, supporting its adoption in real-world scenarios. Additionally, the solution will feature real-time analysis and reporting capabilities, providing timely diagnoses and notifications, which are crucial for early intervention

The project aims to integrate the developed system into existing healthcare frameworks, ensuring scalability and accessibility, particularly in underserved areas. By addressing the limitations of traditional diagnostic methods, this initiative seeks to revolutionize ASD detection, facilitating earlier intervention and improving developmental outcomes for children.

1.3 Scope of the Project

The scope of this project lies in developing an efficient and scalable machine learning framework for the early detection of Autism Spectrum Disorder (ASD) in children.

Utilizing Convolutional Neural Networks (CNN) for image analysis, the project aims to address the shortcomings of traditional diagnostic methods, which are often time-consuming, subjective, and require specialized expertise. By automating the diagnostic process, this project seeks to make ASD detection more reliable, accessible, and faster, particularly benefiting underserved populations.

In the future, this system has the potential to be integrated seamlessly into existing healthcare frameworks, providing a scalable solution for routine ASD screenings. This integration could revolutionize how autism is diagnosed by making advanced diagnostic tools available to a wider demographic, including rural and low-resource areas.

The project establishes a foundation for future research to enhance diagnostic accuracy and robustness. Integrating multimodal data, such as behavioral and demographic information, can refine the model's predictions. By offering insights into ASD indicators, it supports clinical advancements and research, improving outcomes for at-risk children and their families.

1.4 Organization of the Report

Chapter 1 introduces the topic, providing insight and an in-depth overview of the project. It covers the problem statement, highlighting the challenges in early detection of Autism Spectrum Disorder (ASD). The chapter also discusses the objectives, scope, and concludes with the organization of the report.

Chapter 2 includes a literature review, discusses existing systems and their limitations, and introduces the proposed system for early ASD detection using machine learning techniques.

Chapter 3 focuses on the System Requirements Specification, detailing the overall description, functional requirements, and non-functional requirements of the project.

Chapter 4 contains the Gantt chart, presenting the schedule and timeline of the project's tasks and milestones.

Chapter 5 includes various diagrams such as the Architectural Diagram, Use Case Diagram, Sequence Diagram, and Dataflow Diagram, which illustrate the system's design and flow of operations.

Chapter 6 emphasizes the implementation phase, describing the development of the CNN model, the Flask-based interface, and the integration of the AI-driven chat module.

Chapter 7 explains the testing methodologies used, including unit testing, integration testing, and system testing, ensuring the robustness and accuracy of the model and interface.

Chapter 8 showcases the results, with snapshots of the project's interface, model predictions, and graphical representations of performance metrics.

Chapter 9 concludes the report with a summary of the project's outcomes and discusses potential future work for enhancing the system further

Chapter – 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

A literature survey shows the various analysis and research made in the field of interest and results already published, taking into account the various parameters of the project and the extent of the project. It includes researches made by various analysts-their methodology and the conclusion they have arrived at. It is the most important part of the report as it givesthe direction in the area of research.

2.1 Existing Systems

[1] "A Review On: Autism Spectrum Disorder Detection By Machine Learning Using Small Video" By Garg et.al (2024)

The literature review by Garg et al. (2024) explores the use of machine learning (ML) techniques for the early detection of Autism Spectrum Disorder (ASD). ASD, a neurodevelopmental condition, is marked by communication difficulties, restricted interests, and repetitive behaviors. Early diagnosis is emphasized as critical, as timely interventions can significantly improve developmental outcomes and quality of life. The review discusses how ML approaches such as feature extraction, facial recognition, and behavioral analysis can identify patterns unique to ASD. Various ML models, including Support Vector Machines (SVM), Random Forest (RF), and Convolutional Neural Networks (CNN), are reviewed for their high accuracy in distinguishing ASD from typical development (TD). Despite the promise of these methods, the study highlights challenges such as data scarcity and overfitting, particularly when working with small video datasets. Garg et al. suggest semi-supervised learning as a strategy to enhance model performance and advocate for integrating ML frameworks into clinical settings to improve screening speed and accuracy. The paper concludes by emphasizing the transformative potential of ML in revolutionizing ASD diagnosis and calls for further research to enhance model generalizability and real-world applicability.

[2] "Autism Spectrum Disorder Prediction In Children Using Machine Learning" By Abdelwahab et.al (2024)

The literature review by Abdelwahab et al. (2024) explores the use of machine learning (ML) techniques for predicting Autism Spectrum Disorder (ASD) in individuals across different age groups. ASD is a neurodevelopmental condition characterized by challenges in social interaction, communication, and repetitive behaviors, making early detection crucial for effective intervention. The study evaluates multiple ML models, including Support Vector Machines (SVM), Random Forest Classifier (RFC), Logistic Regression (LR), Naïve Bayes(NB), and K-Nearest Neighbors (KNN), using datasets sourced from Kaggle and the UCI Machine Learning Repository. Emphasizing the importance of data preprocessing, the authors discuss normalization and handling missing values to enhancemodel accuracy. Among the models tested, RFC achieved the highest accuracy of 99.75%, followed by strong performances from SVM and LR. This research highlights the potentialof ML as a valuable diagnostic aid in detecting ASD while stressing the need for largerdatasets to improve model generalizability.

[3] "Detection Of Autism Spectrum Disorder (ASD) In Children And Adults Using Machine Learning" by Farooq et al. (2023)

The literature review by Farooq et al. (2023) explores the application of Federated Learning (FL) for Autism Spectrum Disorder (ASD) detection in children and adults. ASD is a neurodevelopmental condition characterized by difficulties in communication, social interaction, and repetitive behaviours, making early and accurate diagnosis essential for effective intervention. Traditional diagnostic methods face challenges due to data-sharing restrictions, privacy concerns, and the need for extensive datasets. FL addresses these issues by enabling decentralized data processing, where Support Vector Machine (SVM) and Logistic Regression (LR) classifiers are trained locally on datasets without transferring sensitive data to a central server. The study achieved remarkable results, with FL-based models attaining 98% accuracy in children and 81% in adults. The locally trained models were integrated at a central server using a meta-classifier, combining the strengths of individual models to enhance prediction accuracy. This approach not only preserves data privacy but also mitigates issues such as network latency and data theft. Farooq et al. compare FL with traditional ML approaches, highlighting its advantages in maintaining data security and enabling collaborative model training across distributed datasets. The authors emphasize the significance of FL in improving the efficiency and scalability of ASD detection while addressing the limitations of centralized systems. They recommend further exploration to enhance FL's generalizability and its application in clinical settings, demonstrating its potential to transform ASD diagnostics.

[4] "Detecting High-Functioning Autism In Adults Using Eye Tracking And MachineLearning" By Yaneva et al. (2020)

The literature review by Yaneva et al. (2020) introduces a non-invasive approach to detecthigh-functioning autism in adults using eye-tracking technology. The study focuses on differences in visual processing during web page browsing tasks, aiming to identify unique patterns in gaze behaviour between individuals with autism and neurotypical participants. The research analysed eye movements recorded from 71 participants, including 31 with high-functioning autism and 40 neurotypical controls, across two separate studies. Byemploying machine learning classifiers on the eye-tracking data, the study achieved a 74% accuracy rate in detecting autism, demonstrating the potential of this method for effective identification. The review highlights the advantages of using eye-tracking technology as a diagnostic tool, emphasizing its cost-effectiveness and scalability for broader application in clinical and nonclinical settings. The paper also discusses the novelty of this approach infocusing on adults with high-functioning autism, a group that often goes undiagnosed dueto subtle symptom presentation and the lack of objective biomarkers in traditional assessments. By leveraging web-based tasks, the study provides a naturalistic and reliable method for capturing cognitive differences, making it a promising direction for future ASD detection research.

[5] "A Narrative Review of Autism Spectrum Disorder in the Indian Context" by Manushi Srivastav et al. (2023)

The literature review by Manushi Srivastav et al. (2023) sheds light on the rising prevalence of Autism Spectrum Disorder (ASD) in India and the unique challenges faced by affected individuals and their families. The review highlights how societal stigma, limited awareness, and inadequate resources significantly hinder timely diagnosis and effective support for individuals with ASD. These barriers often exacerbate the difficulties faced by families, delaying critical interventions that could improve developmental outcomes. The study also identifies several risk factors contributing to ASD in the Indian context, including advanced parental age, neonatal complications, and consanguinity. These factors underline the importance of targeted research and preventive measures to address the growing incidence of ASD. The review emphasizes the urgent need for inclusive education systems, greater employment opportunities, and robust support networks for individuals with ASD and their families. By fostering awareness and implementing structured interventions, the review argues, India can create a more inclusive environment that significantly enhances the quality of life for individuals with ASD and alleviates the burdens on their families.

2.1 Limitations of the Existing System

Existing systems for Autism Spectrum Disorder (ASD) detection rely heavily on subjective assessments, leading to inconsistent and delayed diagnoses. They often require significant time, resources, and access to specialized professionals, which are scarce in underserved areas. Machine learning models face challenges like data privacy concerns and limited applicability to diverse populations. Additionally, societal stigma, lack of awareness, and insufficient infrastructure in regions like India further delay support and intervention. These limitations highlight the need for accessible, objective, and culturally sensitive diagnostic tools.

2.2 Proposed System

The proposed system is a web application designed for early autism detection in children. It features a simple, user-friendly interface that allows parents and guardians to easily sign up, log in, upload images, and view results. The backend, powered by Flask, handles requests, user authentication, and communicates with the TensorFlow model to predict autism signs. User information is securely stored in a SQLite database with encrypted passwords, and SQLAlchemy manages the database operations. The application uses a pre- trained deep learning model to analyze images, ensuring quick processing and timely results, though its accuracy can improve with more data. Privacy is a key priority, with secure password storage and minimal data collection. The chatbot offers interactive, personalized assistance, guiding users through the process of uploading images and checking results, enhancing user engagement and overall experience.

Chapter – 3

SYSTEM REQUIREMENTS SPECIFICATION

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

The System Requirements Specification (SRS) defines the technical and functional needs for the proposed system. It outlines the application's purpose, scope, and performance expectations, ensuring that the development meets user needs effectively. This document serves as a blueprint for developers and stakeholders, detailing the requirements for hardware, software, system features, and user interactions. The SRS ensures clarity and alignment among all involved parties by specifying how the system should function and the constraints within which it operates. It highlights the system's objectives, including usability, efficiency, security, and accuracy, while addressing potential dependencies and assumptions. By adhering to these specifications, the system can deliver a robust, scalable, and user-centric solution tailored to its intended purpose.

3.1 Overall description

The proposed system is a web application designed to detect Autism Spectrum Disorder (ASD) in its early stages in children. It offers a platform for parents and guardians to upload images and assess potential signs of autism through a simple and accessible interface. The system utilizes advanced machine learning techniques, specifically Convolutional Neural Networks (CNNs), to analyze images and provide accurate predictions. This user-friendly application raises awareness, ensures data security, and fosters early interventions, potentially improving developmental outcomes for children.

3.1.1 Product Perspective

The application integrates advanced image recognition models with a responsive and accessible web interface, combining a lightweight backend with user-centric features. It functions as a standalone diagnostic tool, simplifying ASD detection for non-expert users.

3.1.2 Product Function

The system allows users to create accounts, upload images for analysis, and view results. It processes the images using a CNN-based model to classify them as "autistic" or "non-autistic" and provides feedback with clear and actionable information.

3.1.3 User Classes and Characteristics

This system is designed to benefit following classes of users:

- **Primary Users:** The system is primarily intended for parents, guardians, and caregivers who are concerned about early autism detection in children.

- **Technical Expertise:** Recognizing that many users may have limited technical knowledge, the system is designed to be intuitive and easy to navigate.

- **User Experience:** The platform provides step-by-step guidance, clear instructions, and a user-friendly interface, enabling seamless interactions, from signing up to uploading images and viewing results.

- **Empowerment:** By simplifying the process, the system empowers users to access valuable insights without needing advanced technical skills, making autism detection accessible to all.

3.1.4 Design and Implementation Constraints

- Built on Flask for robust backend operations and SQLite for efficient data management.

- Ensures secure storage and processing of uploaded images to protect user privacy and comply with data protection standards.

- Relies on a well-trained TensorFlow model for accurate predictions, requiring high-quality datasets and computational resources.

- Designed to work seamlessly across various devices and platforms for universal accessibility.

- Efficiently manages storage and processing power, especially under high traffic or large-scale deployment conditions.

- Handles errors gracefully, providing clear feedback to users to maintain a smooth user experience.

- Depends on pre-trained models and external libraries, which must be managed for long-term stability and maintainability.

3.1.5 Assumptions and Dependencies

- The system assumes access to labeled datasets of sufficient quality and quantity for training the machine learning model effectively.

- A stable internet connection is required for seamless operation, including user interactions, image uploads, and result retrieval.

- The system depends on the availability and compatibility of key technologies, such as TensorFlow, Flask, and supporting libraries, for backend and model functionality.

3.2 Functional Requirements

The system is designed to meet specific functional requirements to ensure its efficiency and usability. It enables users to register, log in, and securely upload images for analysis. The backend processes these uploads using a TensorFlow-based Convolutional Neural Network (CNN) to detect signs of autism. The system ensures that predictions are quickly generated and displayed to users through a clean and interactive interface. A lightweight SQLite database securely manages user credentials and image-related metadata, with Bcrypt used for password hashing to enhance security. The frontend, built with HTML, CSS, and JavaScript, provides a responsive and user-friendly platform, enabling smooth navigation across devices. Additionally, the system includes features like image preprocessing, such as resizing and normalization, to optimize model performance, ensuring accurate and reliable results.

3.2.1 Hardware Requirements

Hardware requirements refer to the physical parts of a computer and related devices. Internal hardware devices include motherboards, hard drives, and RAM External hardware devices include monitors, keyboards, mice, printers, and scanners.

- **Processor:** Dual-core CPU or higher
- **RAM:** 4GB or more
- **Storage:** 500MB for application and database

3.2.2 Software Requirements

Software requirements deal with the necessary components and frameworks to enable system functionality. The software requirements of our project are given below:

1. Python
2. Machine Learning(TensorFlow for implementing the CNN model)
3. Frontend Development (HTML, CSS, JavaScript, and Bootstrap)
4. Data Extraction(OpenCV and PIL)
5. Flask Framework(backend operations)
6. SQLite(Database)
7. Bcrypt(hashing passwords to ensure user security)

1. Python:

Python is the cornerstone of this project, serving as the main programming language for both backend development and machine learning model integration. Its versatility and extensive library ecosystem enable seamless implementation of key features. For backend development, Python, coupled with the Flask framework, handles routing, request processing, and user authentication efficiently. On the machine learning side, Python integrates the TensorFlow library to develop and deploy the CNN model, enabling the classification of images into "autistic" or "non-autistic" categories. Its simplicity ensures smooth integration between the frontend and backend, providing users with a cohesive and responsive application.

2. Machine Learning(TensorFlow for implementing the CNN model):

The machine learning component of this project is built using TensorFlow, a robust framework for developing deep learning models. A Convolutional Neural Network (CNN) is employed to classify images by extracting key features through convolution and pooling layers. TensorFlow supports training the CNN on augmented and normalized image datasets to enhance the model's ability to generalize. The framework optimizes training through the Adam optimizer and uses binary cross-entropy for loss calculation. Once trained, the model processes uploaded images, predicts their class, and returns results in real-time, achieving high accuracy and efficiency in identifying autistic traits.

3. Frontend Development(HTML, CSS, JavaScript, and Bootstrap):

The frontend leverages a combination of HTML, CSS, JavaScript, and Bootstrap to create a visually appealing, responsive, and user-friendly interface. HTML structures the content, while CSS enhances its design with clean layouts and interactive elements. Bootstrap adds responsiveness, ensuring the application adapts seamlessly to various devices and screen sizes. JavaScript enables dynamic behavior, powering functionalities like opening and closing the chatbot, handling user interactions, and providing smooth transitions. Together, these technologies create an intuitive dashboard where users can upload images, view results, and access the chatbot for assistance, ensuring a modern and engaging experience.

4. Data Extraction (OpenCV and PIL):

For data preprocessing, libraries like OpenCV and Pillow (PIL) are utilized. OpenCV handles image manipulation tasks such as resizing, grayscale conversion, and filtering, while PIL complements this by supporting image loading and format conversion. These tools ensure that all images are uniformly processed before feeding them into the CNN model. Image augmentation techniques like rotation, flipping, and shifting are also implemented using these libraries to diversify the dataset, improve model training, and reduce overfitting, ultimately enhancing the performance of the machine learning model.

5. Flask Framework(backend operations):

Flask serves as the backend framework for managing server-side operations. It handles user requests, routing, and session management with ease. Flask integrates the TensorFlow model to process user-uploaded images and return predictions efficiently. Additionally, Flask's simplicity allows for secure user authentication using the Flask-Login extension. It supports smooth communication between the frontend and backend, ensuring data flow is uninterrupted. Its lightweight nature makes it an excellent choice for rapid development while maintaining the scalability required for this application.

6. SQLite(Database):

SQLite is the database used for securely storing user data, including login credentials and uploaded image details. It is lightweight yet powerful, making it ideal for small-scale projects like this one. SQLite stores image paths, associated results, and user authentication data, ensuring quick retrieval and reliable organization. Its integration with Flask and SQLAlchemy simplifies database operations, allowing for seamless data insertion, querying, and management. This ensures that the application remains efficient and secure while providing users with fast access to their results.

7. Bcrypt(hashing passwords to ensure user security) :

Bcrypt is employed to enhance security by hashing user passwords. It ensures that sensitive data, such as passwords, is never stored in plaintext within the database. By salting and hashing passwords, bcrypt makes it extremely difficult for attackers to reverse-engineer credentials, even if the database is compromised. This robust encryption mechanism helps protect user information and builds trust, ensuring that authentication processes are both secure and efficient.

3.1 Non-Functional Requirements

Non-functional requirements are the requirements that are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behavior. They ensure the system meets quality standards and performs efficiently.

- **Reliability:** Ensures consistent performance of the TensorFlow model with accurate predictions for uploaded images.

- **Robustness:** Capable of handling various image formats like JPEG and PNG without errors during upload or processing.

- **Maintainability:** Modular architecture supports easy updates, including model improvements or feature additions.

- **Availability:** Accessible 24/7 through modern web browsers, ensuring uninterrupted functionality for users.

- **Security:** Implements Bcrypt for password hashing and secure protocols to protect user data and prevent breaches.

- **Usability:** Provides an intuitive and user-friendly interface with clear guidance, designed for non-technical users

Chapter – 4
GANTT CHART

CHAPTER 4

GANTT CHART

A Gantt chart is a type of bar chart, developed by Henry Gantt that illustrates a project schedule. Gantt charts illustrate the start and finish of the terminal elements and summary elements of the project. Terminal elements and summary elements comprise the work breakdown structure of the project.

The following is the Gantt chart of the project "Machine Learning-Based Early Detection Of Autism In Children".

Table 4.1 Gantt chart of planning and scheduling of project

Number	Task	Start	End	Duration (In days)
1	Synopsis	05-Oct-2024	10-Oct-2024	5
2	Presentation on idea	23-Sep-2024	23-Sep-2024	1
3	Software Requirement Specification	12-Oct-2024	17-Oct-2024	5
4	System Design	20-Oct-2024	25-Oct-2024	5
5	Implementation	27-Oct-2024	24-Nov-2024	28
6	Presentation on work progress	09-Nov-2024	09-Nov-2024	1
7	Testing	25-Nov-2024	30-Nov-2024	5
8	Result and Report	01-Dec-2024	07-Dec-2024	7

ACTIVITY/ MONTH	**SEP**	**OCT**	**NOV**	**DEC**
SYNOPSIS		▨		
PRESENTATION ON IDEA	▨			
SOFTWARE REQUIREMEN SPECIFICATION		▨		
SYSTEM DESIGN		▨		
IMPLEMENTATION		▨		
PRESENTATION PO WORK PROGRESS			▨	
TESTING			▨	
RESULT AND REPORT				▨

Figure 4.1: Gantt chart

Chapter – 5
SYSTEM DESIGN

CHAPTER 5

SYSTEM DESIGN

5.1 Architectural Diagram

An architectural description outlines a system's structure, detailing its components, their roles, and interactions to support development and integration. Architectural design specifies the system's overall structure by identifying components and their control and data flow. Rectangles represent functional units, and arrows indicate connections. Figure 5.1 depicts the project's Architectural Design Diagram.

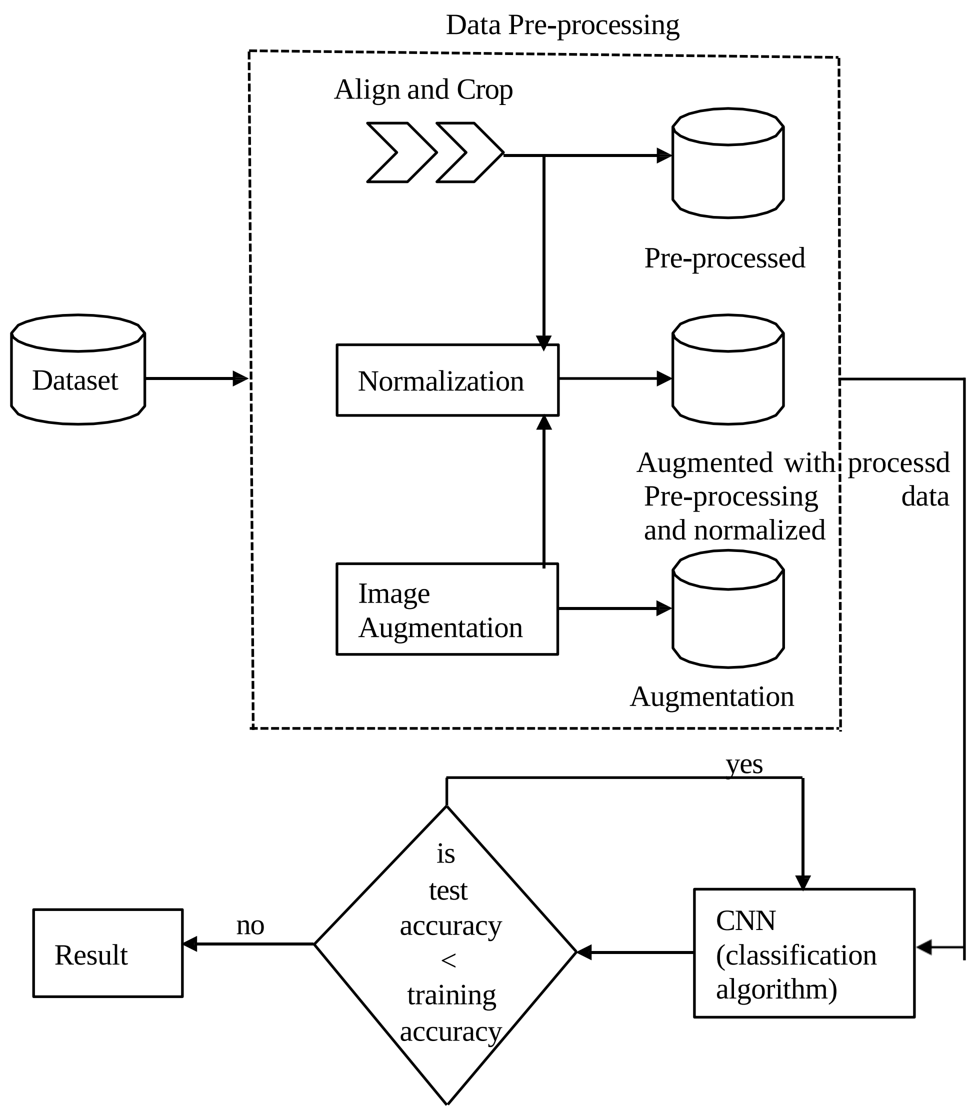


Figure 5.1: Architectural Diagram of Proposed System

The architectural design of this project adopts a structured and iterative approach to optimize image classification using a Convolutional Neural Network (CNN). It consists of three key stages: data preprocessing, CNN-based classification, and performance evaluation. The data preprocessing stage includes image pre-processing, normalization, and augmentation, ensuring the dataset is cleaned, scaled, and enriched for better generalization.These steps create multiple versions of the processed data, including pre-processed, augmented, and normalized datasets, which enhance the diversity and quality of the trainingdata.The CNN is employed as the core classification algorithm, leveraging the processed data for training and testing. The system evaluates performance by comparing test and training accuracy to ensure reliability on unseen data. If test accuracy is low, it iterates back to data preprocessing for refinement. This adaptable architecture ensures seamless transitions and robust image classification.

5.2 Activity Diagram

Activity diagram is an important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

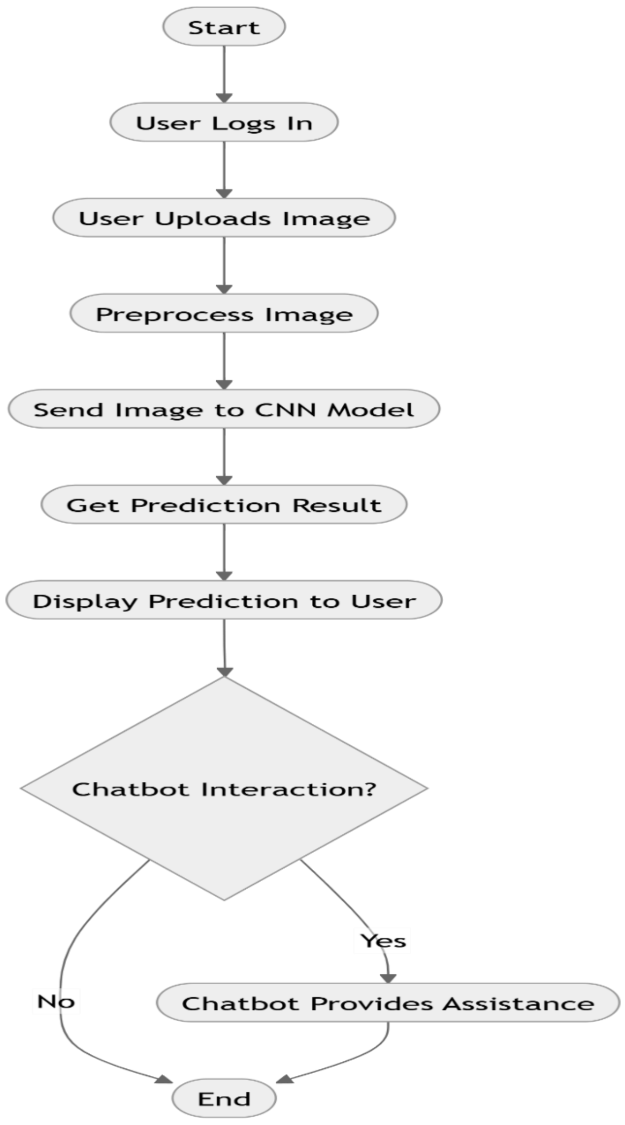


Figure 5.2: Activity Diagram of Proposed System

The activity diagram represents a system process where a user begins by logging in. Once logged in, the user uploads an image to the system. The uploaded image is then preprocessed to prepare it for further analysis. After preprocessing, the image is sent to a Convolutional Neural Network (CNN) model, which generates a prediction result. This result is then displayed to the user. At this point, the system checks if chatbot assistance isrequired. If the user does not need assistance, the process ends. However, if assistance is required, the chatbot provides the necessary help before the process concludes.

5.3 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the flow of data through an information system. It is a structured analysis and design technique that models how information moves between different entities within a system. The main purpose of a DFDis to depict the system's components, processes, and data flows, as well as the inputs and outputs associated with each component. It is a useful tool for understanding, analyzing, and designing information systems. The Figure 5.3.1 shows the Data Flow Diagram of Proposed System.

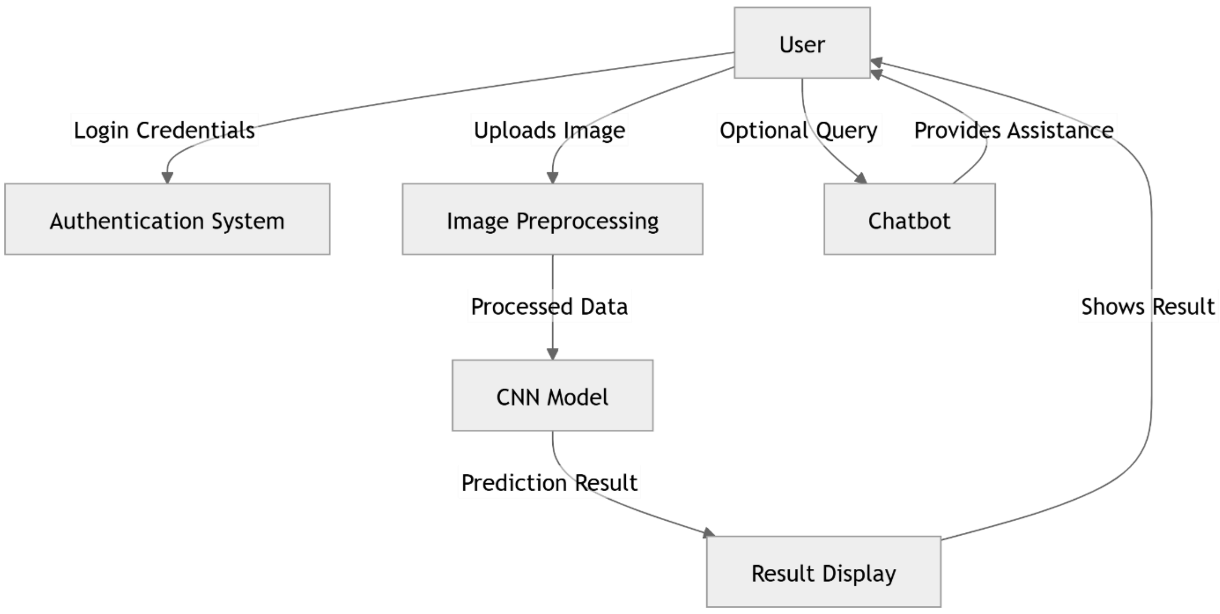


Figure 5.3: Data Flow Diagram of Proposed System

The Data Flow Diagram (DFD) illustrates the system's interaction with the user and the flow of data across various components. The process begins with the user providing login credentials to the authentication system for validation. Upon successful authentication, the user gains access to the system and uploads an image, which is sent to the image preprocessing module. The preprocessed image is then forwarded to the CNN model, where it is analyzed to generate a prediction result. This result is passed to the result display module, where it is presented to the user. Additionally, the user has the option to interact with a chatbot for assistance by raising queries. The chatbot processes these queries and provides the necessary support. Finally, the prediction results, along with any chatbot assistance, are delivered to the user, completing the data flow cycle.

5.4 Sequence Diagram

A sequence diagram is a powerful visual representation in software engineering, illustrating the interactions between objects or components within a system over time. It showcases theflow of messages exchanged between these objects, depicting the sequence of events that occur during the execution of a particular scenario or functionality. Sequence diagrams playa crucial role in understanding the dynamic behavior of a system, offering insights into the order of operations and the collaboration between various components. By mapping out the sequence of interactions between objects, sequence diagrams aid in identifying potential bottlenecks, errors, or inefficiencies within the system's design.

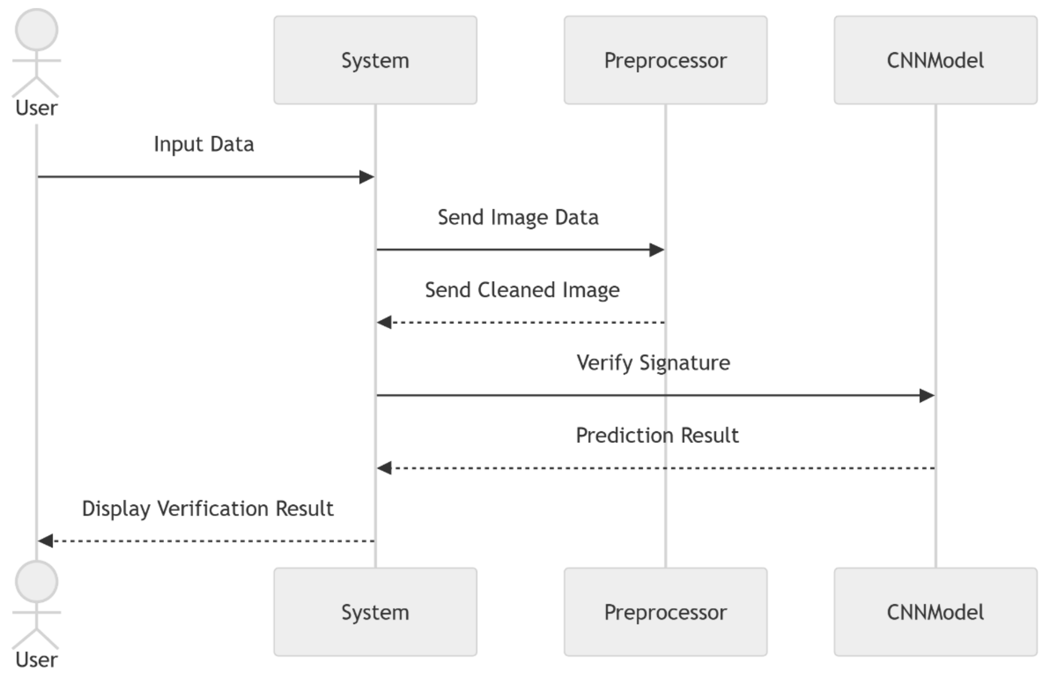


Figure 5.4: Sequence Diagram of Proposed System

The sequence diagram depicts the interaction between the user, the system, the preprocessor, and the CNN model in a step-by-step manner. Initially, the user provides input data to the system. The system forwards the image data to the preprocessor for cleaning and preparation. The preprocessor processes the image and sends the cleaned image back to the system, which then verifies the signature and forwards the data to the

CNN model. The CNN model processes the input and generates a prediction result. This result is returned to the system, which then displays the verification outcome to the user, completing the interaction. The sequence diagram emphasizes the sequential flow of actions and the communication between components for prediction verification.

5.5 Use Case Diagram

A use case diagram is a graphical depiction of the various interactions between actors and a system, illustrating the functionality and behavior of a software application or system. It provides a high-level overview of the different ways users (actors) can interact with the system to achieve specific goals or tasks. Use case diagrams are instrumental in understanding the system's requirements and functionalities from an end-user perspective,helping stakeholders visualize the intended behavior and scope of the system. By representing the system's functionalities in a clear and organized manner, use case diagrams facilitate effective communication between developers, designers, and stakeholders throughout the software development lifecycle.

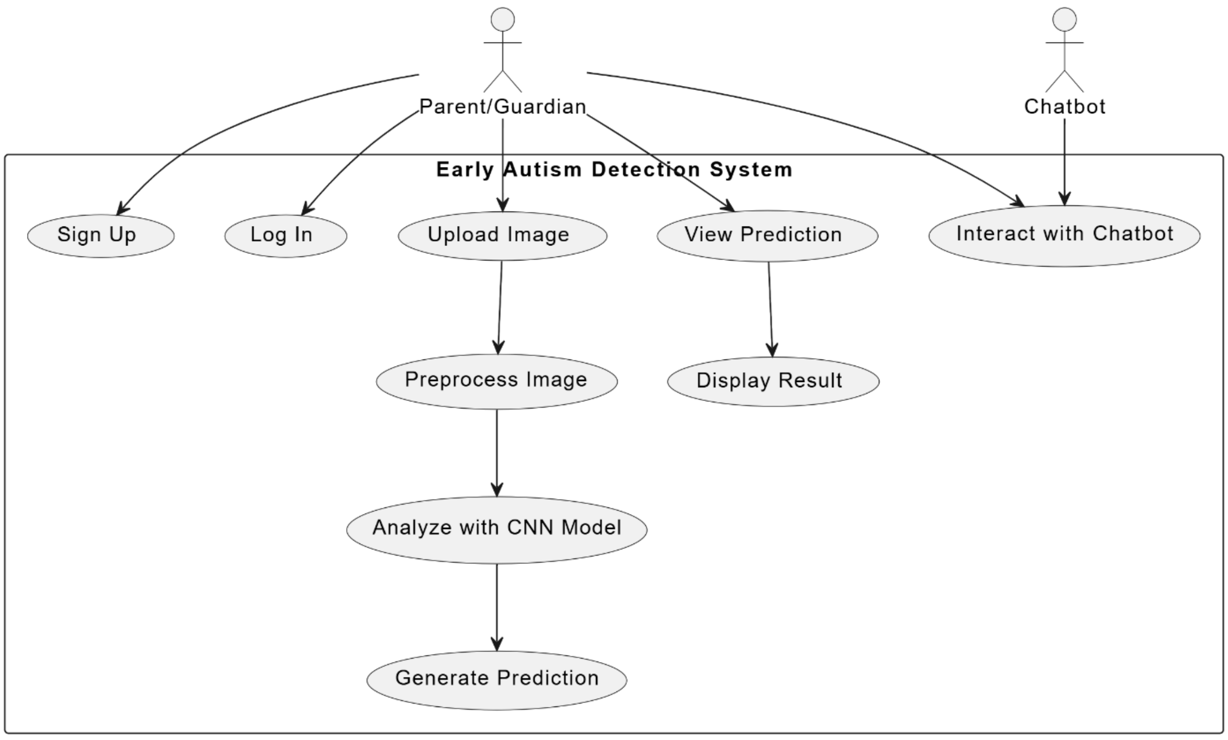


Figure 5.5: Use Case Diagram of Proposed System

The system involves two actors: Parent/Guardian and Chatbot. The Parent/Guardian interacts with several functionalities provided by the system. Initially, the Parent/Guardian can Sign Up or Log In to access the system. Once logged in, they can Upload Image of a child for analysis. This uploaded image undergoes the process of Preprocessing, followed by Analysis using a CNN Model, and finally, a Prediction is

Generated. The Parent/Guardian can then View the Prediction and access the Result Displayed by the system. Additionally, the Parent/Guardian can communicate directly with the Chatbot via the Interact with Chatbot use case to seek guidance or ask questions about the process.

The Chatbot actor is responsible for providing assistance and guidance to the Parent/Guardian throughout the interaction, ensuring a seamless user experience. The system's flow connects key use cases, showing how the inputs from the Parent/Guardian lead to actionable outputs, such as predictions and results, through a structured and intuitiveprocess.

.

Chapter – 6

IMPLEMENTATION

CHAPTER 6

IMPLEMENTATION

Implementation involves the practical realization of the proposed system, transforming the design and plan into a functional application. It includes setting up the system, whether as a completely new design or a significant modification of an existing one, ensuring the system operates as intended. An implementation plan was devised before starting the process to ensure smooth execution.

Testing, a critical phase in the development lifecycle, validates the functionality and reliability of the system. Test cases were executed, and their outcomes evaluated to verify expected performance. Identified errors were corrected, and the process was documented for future reference. Multiple rounds of testing were conducted to ensure the system's readiness for deployment. It's important to note that testing is distinct from Software Quality Assurance (SQA), which focuses on overall quality management across all development phases.

6.1 Set Up Development Environment

Setting up the development environment is a crucial step in ensuring the seamless implementation of the project. It begins with installing Python 3.8 or higher and setting up Anaconda Navigator to manage dependencies and create a virtual environment. Jupyter Notebook is configured for running Python scripts efficiently, and essential libraries like TensorFlow, Keras, NumPy, Pandas, Matplotlib, and OpenCV are installed using pip or conda. Flask is then set up to handle backend development and integrate the machine learning model with the user interface. Additionally, an integrated development environment (IDE) such as Visual Studio Code or PyCharm is configured for efficient code management. Ensuring the system has adequate computational resources, including GPU support for training deep learning models, further enhances the environment's capabilities and supports the project's requirements.

6.2 Required Module Implementation

The implementation of required modules focuses on developing and integrating essential components for the project. The first module involves data preprocessing, where image datasets are cleaned, resized, and augmented to ensure diversity and improve model performance. The next step is the design and training of the Convolutional Neural Network (CNN) model, which forms the core of the Autism Spectrum Disorder detection system. Transfer learning is utilized to enhance the model's accuracy and efficiency.

6.3 Algorithm and Pseudo Code

This section explains the algorithm and provides the pseudo code for the process involved in early autism detection using image analysis and a machine learning model.

6.1.1 Algorithm: Early Autism Detection System

Step 1: Initialize system components

Set up Flask web application for user interface (UI) and backend. Load the pre-trained Convolutional Neural Network (CNN) model for image analysis. Initialize database connection using SQLite for user management.

Step 2: User Sign Up and Log In

Allow users (parents/guardians) to sign up with personal details (name, email, password).Secure passwords using hashing for storage in the database. Allow users to log in using their credentials.

Step 3: Image Upload:

After logging in, provide an option to upload an image. Validate the uploaded image to ensure it is in the correct format (e.g., JPEG, PNG).

Step 4: Preprocess Image

Resize the image to a consistent size compatible with the CNN model. Apply any necessary image enhancements like noise removal, contrast adjustment, etc.

Step 5: Analyze Image Using CNN

Feed the preprocessed image into the pre-trained CNN model. Perform the forward pass to extract features and make predictions.

Step 6: Display Prediction Result

Get the prediction result (autism or non-autism) based on the CNN model's output. Show the result to the user in a readable format.

Step 7: Chatbot Interaction :

Provide an option for the user to interact with a chatbot for additional assistance or clarification. The chatbot can guide the user through the process or answer any questions.

Step 8: End the process

After displaying the result, allow users to log out and safely exit the system.

6.4 Code Snippets

Flask Web Application: Early Autism Detection

6.2.1 Flask App Code Snippet:app.py

```
import os
import numpy as np
from flask import Flask, render_template, request, redirect, url_for, flash
from flask_sqlalchemy import SQLAlchemy
from flask_bcrypt import Bcrypt
from flask_login import UserMixin, login_user, login_required, logout_user, current_user,
LoginManager
from werkzeug.utils import secure_filename
import tensorflow as tf
from PIL import Image
from flask_mail import Mail, Message  # Import Flask-Mail for email functionality
app = Flask(__name__)
app.secret_key = 'your_secret_key'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
bcrypt = Bcrypt(app)
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'
app.config['UPLOAD_FOLDER'] = 'uploads/'
app.config['ALLOWED_EXTENSIONS'] = {'jpg', 'jpeg', 'png'}
if not os.path.exists(app.config['UPLOAD_FOLDER']):
    os.makedirs(app.config['UPLOAD_FOLDER'])
app.config['MAIL_SERVER'] = 'smtp.gmail.com'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'sanyukthasatishshetty@gmail.com'
app.config['MAIL_PASSWORD'] = '12345'
mail = Mail(app)
def allowed_file(filename):
```

```
img = Image.open(image_path).convert('RGB')
    img = img.resize((128, 128))  # Resize to match model input
    img_array = np.array(img) / 255.0  # Normalize pixel values
    return np.expand_dims(img_array, axis=0)
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        if User.query.filter_by(username=username).first():
            flash('Username already exists.', 'danger')
            return redirect(url_for('signup'))
        hashed_password = bcrypt.generate_password_hash(password).decode('utf-8')
        new_user = User(username=username, password=hashed_password)
        db.session.add(new_user)
        db.session.commit()
        flash('Account created successfully!', 'success')
        return redirect(url_for('login'))
    return render_template('signup.html')
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = User.query.filter_by(username=username).first()
        if user and bcrypt.check_password_hash(user.password, password):
            login_user(user)
            flash('Logged in successfully.', 'success')
            return redirect(url_for('dashboard'))
        else:
            flash('Invalid username or password.', 'danger')
    return render_template('login.html')
```

```
@app.route('/dashboard')
@login_required
def dashboard():
    return render_template('dashboard.html', username=current_user.username)
@app.route('/upload', methods=['GET', 'POST'])
@login_required
def upload():
    if request.method == 'POST':
        if 'file' not in request.files:
            flash('No file uploaded.', 'danger')
            return redirect(request.url)

        file = request.files['file']
        if file.filename == '':
            flash('No selected file.', 'danger')
            return redirect(request.url)

        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(file_path)

            if model:
                try:
                    image_data = preprocess_image(file_path)
                    prediction = model.predict(image_data)
                    result = 'Autistic' if prediction[0][0] > 0.5 else 'Non-Autistic'

                    # Send email to parent's email
                    parent_email = request.form['email']
                    child_name = request.form['name']
                    child_age = request.form['age']

                    msg = Message(
                        "Autism Detection Report",
                         sender="your_email@gmail.com",
                        recipients=[parent_email]
                    )
                    msg.body = (f"Dear Parent,\n\n"
                    f"Here is the autism detection report for your child, {child_name} (Age:
                         {child_age}):\n\n"
                               f"Detection Result: {result}\n\n"
                               f"Thank you for using our service.\n\n"
                               f"Best regards,\nAutism Detection Team")
                    mail.send(msg)
                              flash('Report sent to parent\'s email successfully.', 'success')
                    return redirect(url_for('result', result=result))

                except Exception as e:
                    flash(f"Error during prediction or email sending: {str(e)}", 'danger')
            else:
                flash('Model not loaded. Please check the setup.', 'danger')

return render_template('upload.html')
@app.route('/result')
@login_required
def result():
    result = request.args.get('result', 'No result available.')
    return render_template('result.html', result=result)
@app.route('/logout')
@login_required
def logout():
    logout_user()
    flash('Logged out successfully.', 'info')
    return redirect(url_for('login'))
@login_manager.unauthorized_handler
def unauthorized():
    flash("You need to log in to access this page.", "warning")
    return redirect(url_for('login'))
```

```
if _name___== '_main_':
    with app.app_context():
        db.create_all()
    app.run(debug=True)
```

6.2.2 Code Snippet: Image Classification using CNN: train.py

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from sklearn.model_selection import train_test_split
import numpy as np
import os
import cv2

def load_data(data_path):
    images, labels = [], []
    for label, class_dir in enumerate(['Non-Autistic', 'Autistic']):
        class_path = os.path.join(data_path, class_dir)
        if not os.path.exists(class_path):
            raise FileNotFoundError(f"Directory not found: {class_path}")
        for img_name in os.listdir(class_path):
            img_path = os.path.join(class_path, img_name)
            img = cv2.imread(img_path, cv2.IMREAD_COLOR)
            if img is None:
                print(f"Warning: Unable to read image: {img_path}")
                continue
            img = cv2.resize(img, (128, 128))  # Resize to match input size
            images.append(img)
            labels.append(label)
    return np.array(images), np.array(labels)
      data_path = r"C:\Users\91845\Desktop\major project\dataset"
```

```
try:

    X, y = load_data(data_path)
    X = X / 255.0 # Normalize pixel values to [0, 1]
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=32)
    model.save('models/trained_model.h5')
    print("Model training complete and saved as 'models/trained_model.h5'.")
except FileNotFoundError as e:
    print(f"Error: {e}")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

6.2.3 Code Snippet: test.py

```
import tensorflow as tf
import numpy as np
import cv2

def predict_image(image_path, model_path='models/trained_model.h5'):
    model = tf.keras.models.load_model(model_path)
    img = cv2.imread(image_path, cv2.IMREAD_COLOR)
    img = cv2.resize(img, (128, 128))
    img = img / 255.0
    img = np.expand_dims(img, axis=0)  # Add batch dimension
    prediction = model.predict(img)
    result = 'Autistic' if prediction[0][0] > 0.5 else 'Non-Autistic'
    return result
image_path = '0001.jpg'
print(f"The predicted result for the image is: {result}")
```

Chapter – 7

TESTING

CHAPTER 7

TESTING

Testing is a crucial phase in the software development lifecycle, where the program is evaluated against predefined test cases to ensure it behaves as expected. Any discrepancies or errors are documented and corrected through established procedures, ensuring the software meets quality standards before implementation. It is important to distinguish software testing from Software Quality Assurance (SQA), which covers a wider range of processes beyond testing, including all stages of software development and deployment. While testing focuses on functionality, SQA ensures overall software quality across the entire development process.

7.1 Testing Levels

Testing is an integral part of the **Verification and Validation** process, playing a critical role in ensuring software quality and reliability. It serves as a cornerstone of quality assurance by identifying and rectifying errors before deployment. The primary objective of testing is to uncover previously undiscovered bugs, ensuring the software performs as expected under various conditions. Effective testing is characterized by well-designed test cases that have a high probability of identifying unnoticed errors. Adopting a pessimistic approach during testing—running the software with the specific intent of finding flaws—further enhances the robustness of the application. Testing is conducted at multiple levels, including unit testing to validate individual components, integration testing to ensure modules work together seamlessly, and system testing to evaluate the software's overall functionality in a real-world scenario.

7.1.1 Unit Testing

Unit testing is a critical phase to verify the functionality of individual components such as the image preprocessing module, Convolutional Neural Network (CNN) model, and chatbot responses. Each component is tested in isolation to ensure they work correctly before integration. For example, the image preprocessing module is tested to ensure that it correctly resizes, normalizes, and augments input images for accurate model predictions. Similarly, the CNN model's prediction function is tested to ensure that it correctly classifies images as either "Autistic" or "Non-Autistic" based on preprocessed input. The chatbot's response function is also unit tested to verify that it provides accurate and contextually appropriate answers to user queries. Automated unit tests are used to quickly identify defects and ensure that each unit produces the correct output. These tests are particularly important to catch errors early, prevent them from affecting other parts of the system, and ensure reliable performance throughout the development process.

7.1.2 Integration Testing

Integration testing focuses on verifying the seamless interaction between different modules of the Autism Spectrum Disorder detection project. It ensured that the user interface, backend Flask application, TensorFlow model, and database functioned cohesively. For instance, user inputs, such as credentials during login or image uploads, were tested to confirm proper transmission from the frontend to the backend without errors. The Flask application was tested to ensure that uploaded images were correctly saved, preprocessed (resized, normalized, and reshaped), and passed to the TensorFlow model for predictions. The SQLite database integration was validated to ensure accurate storage and retrieval of user data during signup, login, and session management. End-to-end tests confirmed that users could register, log in, upload an image, and view the model’s prediction seamlessly. Integration testing guaranteed that all components worked together effectively, providing an error-free and functional user experience.

7.1.3 System Testing

System testing validates the overall functionality and performance of the Autism Spectrum Disorder detection system against predefined requirements. It ensures seamless navigation through features like sign-up, login, image upload, and prediction generation. Uploaded images were tested for correct preprocessing and accurate predictions by the TensorFlow CNN model. The chatbot feature was evaluated for responsiveness, and scenarios like invalid inputs, user session management, and route redirections were tested. Stress tests ensured the system's robustness under multiple user requests, while database integration was checked for proper data storage and retrieval. This testing confirmed the system's reliability, usability, security, and scalability.

7.1.4 Acceptance Testing

Acceptance testing ensures the Autism Spectrum Disorder detection system meets user expectations and aligns with project objectives. This phase evaluated the system's usability, functionality, and performance from an end-user perspective. Key areas included verifying the user interface for intuitiveness and ease of navigation, ensuring users could seamlessly sign up, log in, upload images, and receive accurate predictions such as "Autism Detected" or "No Autism Detected." The chatbot was tested for prompt and helpful responses to user queries. Scenarios involving invalid actions, like unsupported file uploads or blank fields, were also assessed to confirm proper error handling. Feedback from testers was used to enhance functionality, reliability, and responsiveness, validating the system's readiness for real-world deployment.

7.1 Test Cases

A test case is a structured document used in software testing, including details of events, actions, inputs, outputs, expected results, and actual results. Test cases aim to verify software functionality, identify potential errors, and ensure compliance with requirements. Each test case is assessed using individual PASS/FAIL criteria, with a PASS result required for the application to be considered functional.

Table 7.1: Test cases for the project

Test Number	**Test Cases**	**Expected Output**	**Actual Output**	**Status**
1.	Image upload functionality	Image is successfully uploaded	Image uploaded successfully	**PASS**
2.	Prediction generation	Correct classification (Autistic/Non-Autistic)	Accurate prediction displayed	**PASS**
3.	Chatbot assistance	Relevant and helpful responses provided	Responses matched expectations	**PASS**
4.	User login	User is logged in successfully with valid credentials	Login successful	**PASS**
5.	System error handling	Appropriate error message displayed for invalid input	Error handled correctly	**PASS**

Chapter – 8

RESULTS AND DISCUSSION

CHAPTER 8

RESULTS AND DISCUSSION

The outcomes of the Autism Spectrum Disorder detection system, focusing on its performance, functionality, and user satisfaction. The system successfully enabled user registration, login, image upload, and prediction generation using a TensorFlow CNN model, accurately detecting Autism Spectrum Disorder with results like "Autism Detected “or "No Autism Detected." The integrated chatbot provided useful assistance, enhancing the user experience. The system performed reliably under various test scenarios, handling edge cases and large image uploads effectively. It also managed invalid inputs by displaying appropriate error messages. The system met all functional and non-functional requirements, with high accuracy and an intuitive interface. Challenges included fine-tuning the model for diverse datasets and input compatibility. Suggested improvements involve multi-language support and advanced chatbot features. Feedback was positive, confirming the system's effectiveness for early detection and user support. The system is ready for real-world deployment, demonstrating robust performance and accurate predictions.

8.1 Snapshots

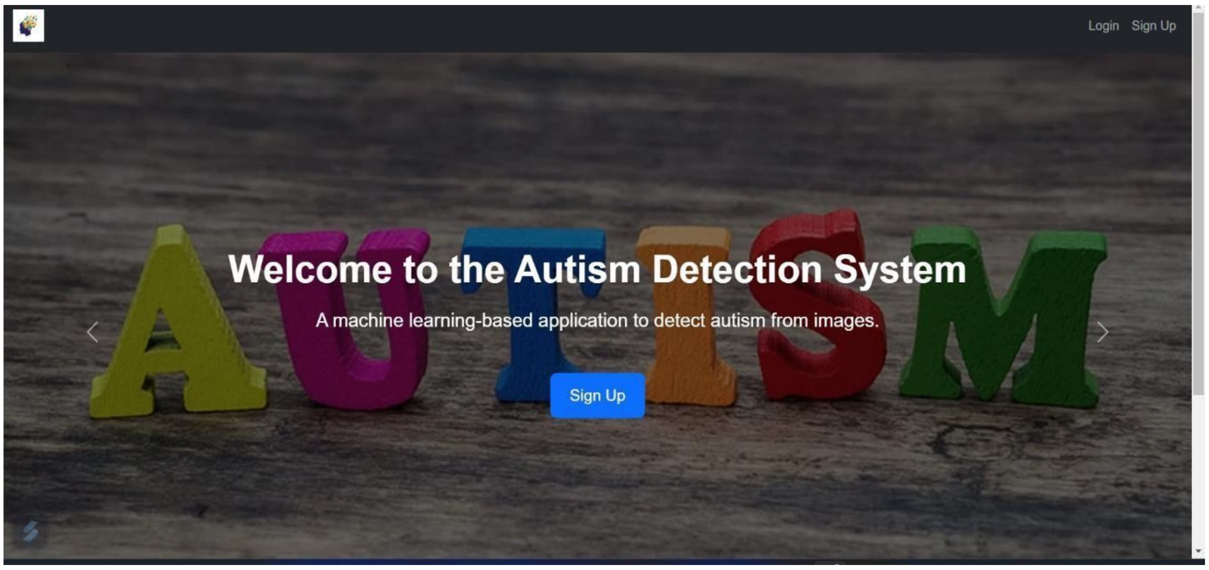

Login
Sign Up
Welcome to the Autism Detection System
A machine learning-based application to detect autism from images.
Sign Up

Figure 8.1.1: Home Page

Above, Figure 8.1.1 shows the homepage of the Autism Spectrum Disorder detection system. It features a clean and intuitive user interface with clear options for users to either sign up or log in.The layout is simple and easy to navigate, allowing users to quickly access the system's main functions. The design ensures a seamless experience for both new and returning users, with responsive elements that maintain consistency across devices.

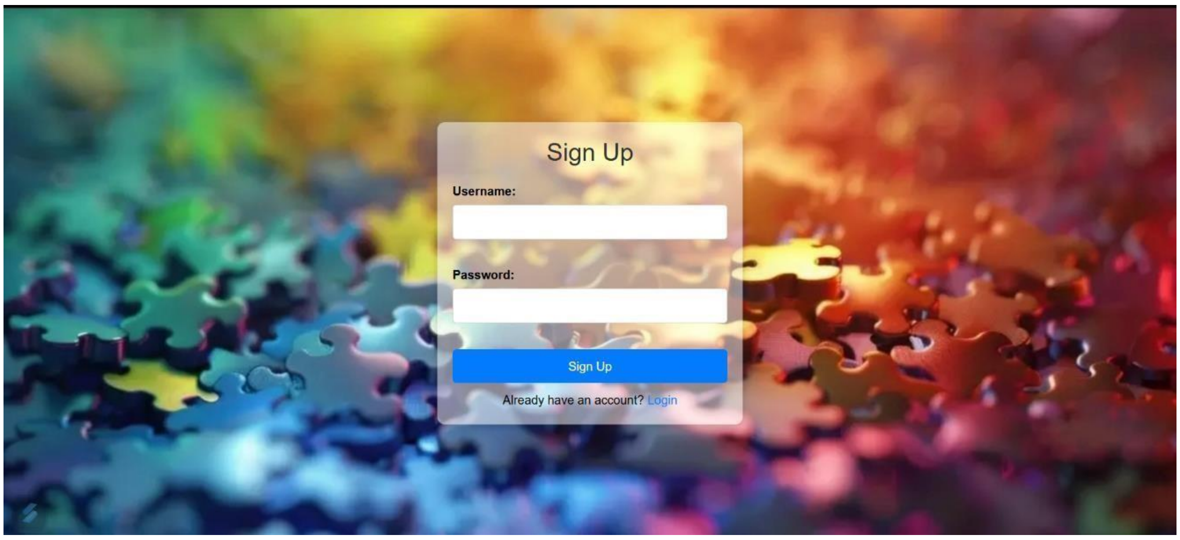


Figure 8.1.2: Signup Page

Above, Figure 8.1.2 shows the Sign-Up page, which displays the registration form with fields for username, email, and password. It highlights form validation to ensure proper user data input, guiding users to fill in the required information accurately. The page also includes error messages to prompt users in case of incorrect or incomplete submissions, ensuring smooth and accurate registration

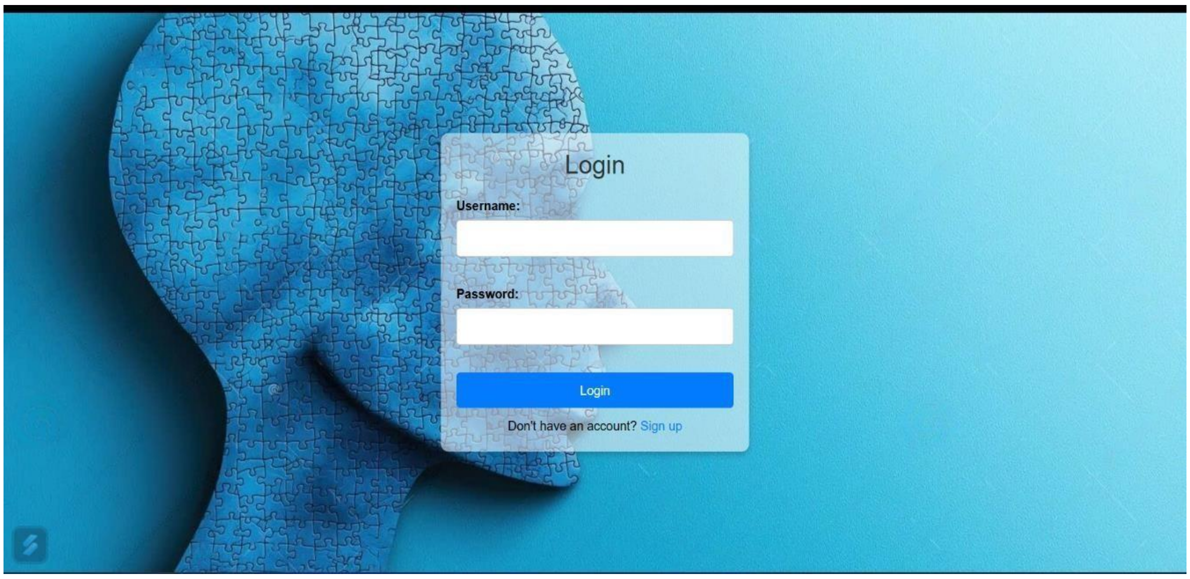
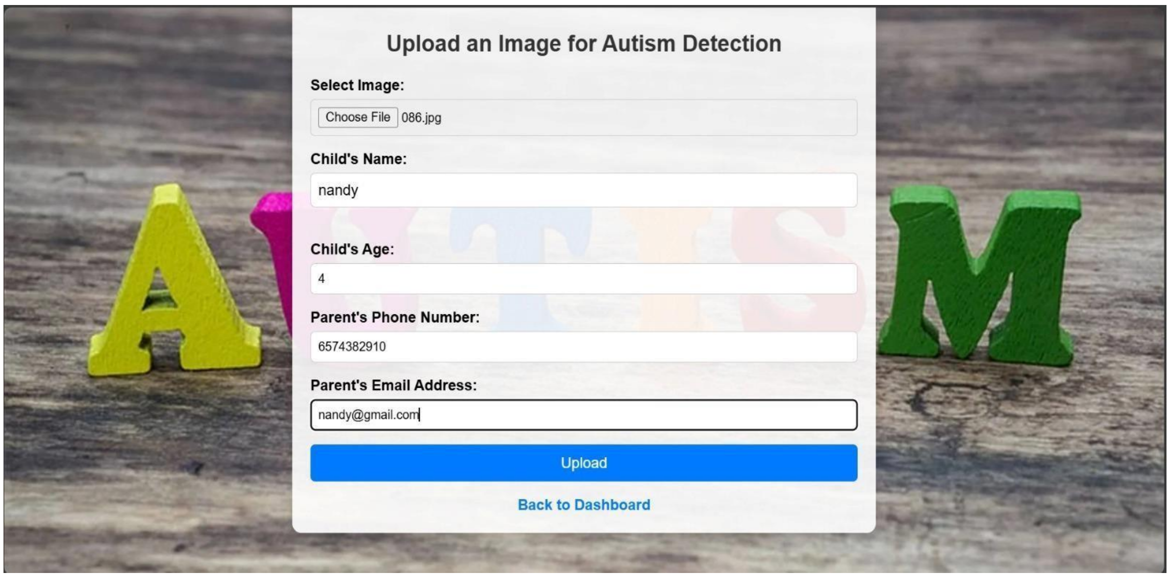


Figure 8.1.3: Login Page

Above, Figure 8.1.3 shows the Login page, which provides the interface for existing users to enter their credentials. It includes error handling for incorrect username or password inputs, displaying appropriate messages to inform users when their login attempt fails. This ensures a smooth user experience by guiding users to correct errors and successfully access their accounts.



Upload an Image for Autism Detection

Select Image: Choose File 086.jpg

Child's Name: nandy

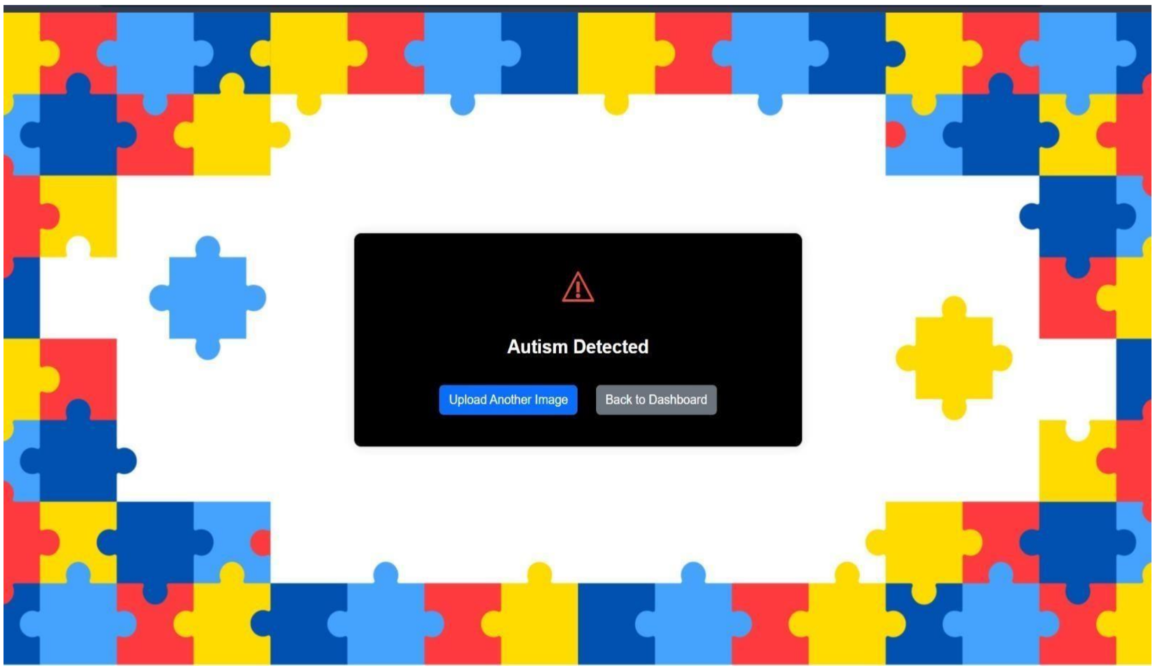
Child's Age: 4

Parent's Phone Number: 6574382910

Parent's Email Address: nandy@gmail.com

Figure 8.1.4: Image Upload Page

Above, Figure 8.1.4 shows the Image Upload page, where users can upload images for analysis. The interface confirms the successful upload of the image, ensuring users are notified once the file has been successfully processed. This feature streamlines the processof submitting images for prediction and guides users through each step.



Autism Detected

Figure 8.1.5: Prediction Result Page(positive)

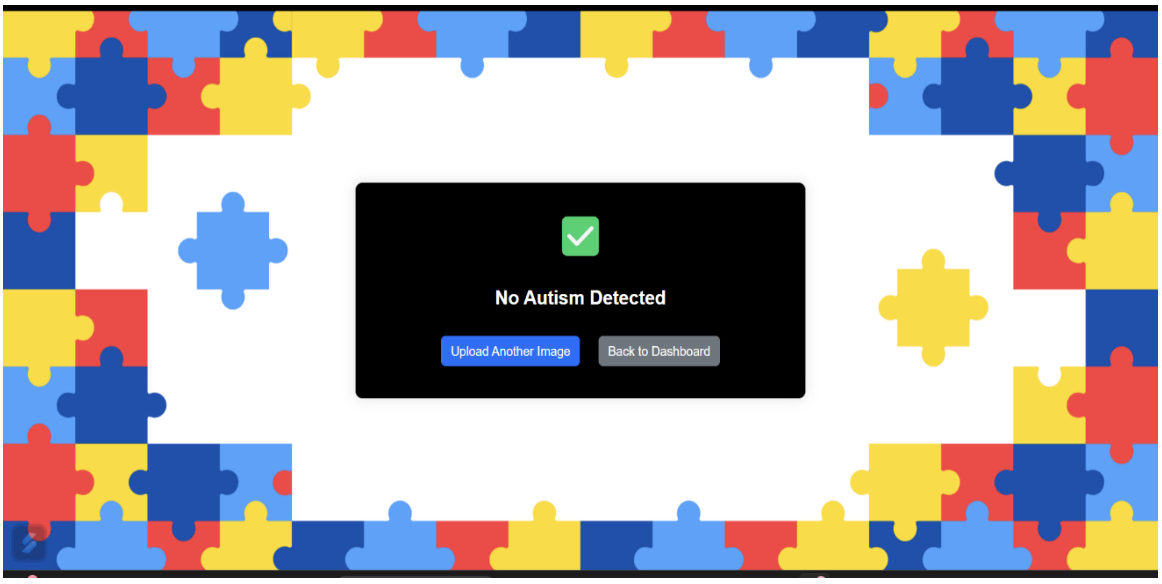
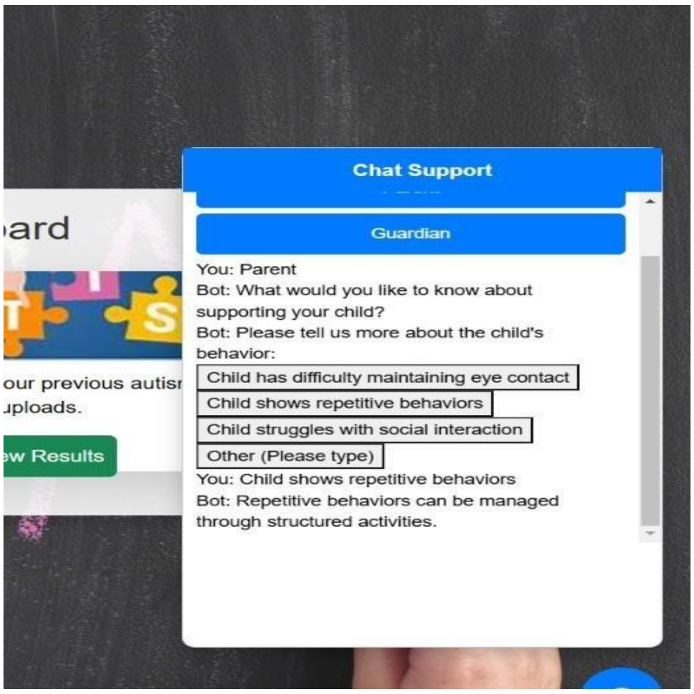


Figure 8.1.6: Prediction Result Page(negative)

Above, Figures 8.1.5 and 8.1.6 show the Prediction Result page, where the system displaysthe analysis output. In Figure 8.1.5, the result "Autism Detected" is clearly shown, indicating that the uploaded image has been classified as positive. In Figure 8.1.6, "No Autism Detected" appears, showing a negative result. Both results are presented concisely,ensuring users can easily interpret the outcome of the analysis.



You: Parent
Bot: What would you like to know about supporting your child?
Bot: Please tell us more about the child's behavior:
Child has difficulty maintaining eye contact
Child shows repetitive behaviors
Child struggles with social interaction
Other (Please type)
You: Child shows repetitive behaviors
Bot: Repetitive behaviors can be managed through structured activities.

Figure 8.1.7: Chatbot Interaction

Above, Figure 8.1.7 shows the Chatbot Interaction page, where users interact with the chatbot for assistance. The snapshot highlights user queries and the chatbot's responses, demonstrating how the chatbot provides relevant information based on the user's needs, such as offering guidance on next steps or answering general queries. This featureenhances user experience by offering real-time support.

Chapter – 9
CONCLUSION

CHAPTER 9

CONCLUSION

9.1 Conclusion

The Autism Spectrum Disorder (ASD) detection system presented in this project successfully addresses the need for early identification of ASD in children. By leveraging machine learning techniques, specifically a Convolutional Neural Network (CNN) model in TensorFlow, the system is able to analyze images and accurately predict whether a childmay have autism. The integration of a user-friendly interface, which includes features like user registration, image upload, and prediction results, ensures that parents and caregivers can easily navigate the system. Furthermore, the inclusion of a chatbot feature enhances theuser experience by providing real-time assistance.

The system's robustness was demonstrated during testing, where it consistently produced accurate results and handled edge cases effectively. The integration of error handling ensured that users were guided appropriately in case of invalid inputs. Overall, the system met its functional and non-functional requirements, such as usability, security, and scalability, making it a reliable tool for early ASD detection.

9.2 Future Work

As described below, the project can improve its scope in several ways to further enhance theimpact of the models in cancer diagnosis and treatment:

- **Parental Support Community**: Integrate online support groups and live consultations with medical professionals to connect parents and provide expert advice, fostering a sense of community and reducing isolation.
- **Integration with Therapy Tools**: Incorporate behavioral therapy tracking and gamified therapy features to monitor a child's progress and engage them in therapeutic exercises in a fun, interactive manner.
- **Personalized Parenting Tips**: Deliver daily, tailored advice to parents on managing autism-related behaviors, including activity suggestions, social skills exercises, and communication techniques.
- **Location-Based Support Services**: Provide a map-based feature to help parents find nearby autism specialists, therapy centers, and support groups, along with notifications about autism-related events and workshops in their area

REFERENCES

REFERENCES

[1] K. Garg, N. N. Das and G. Aggrawal, "A Review On: Autism Spectrum Disorder Detection by Machine Learning Using Small Video," *2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, Jaipur, India, 2023, pp. 1-8.

[2] Mahmoud M. Abdelwahab, Khamis A. Al-Karawi and E. M. Hasanin et al. “Autism Spectrum Disorder Prediction in Children Using Machine Learning”. *JDR*. 2024. Vol. 3(1).

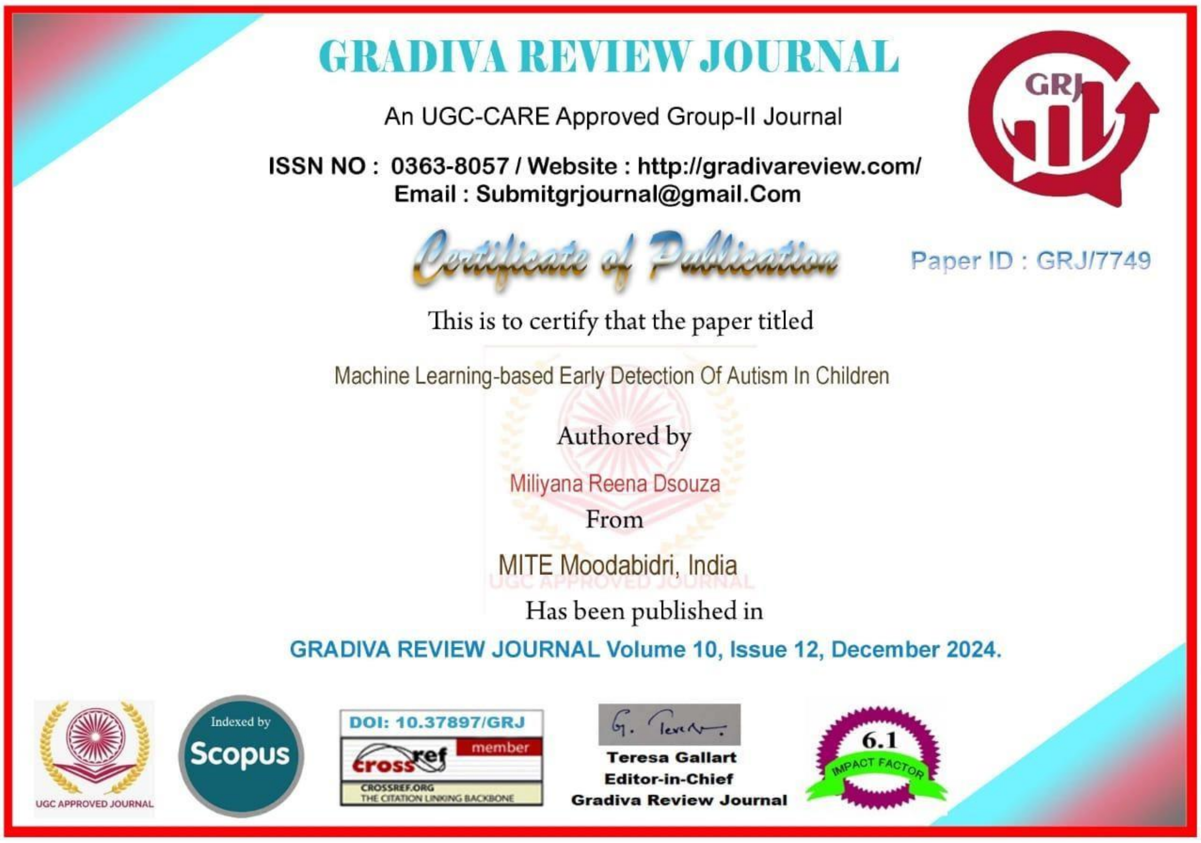
[3] Farooq, M.S., Tehseen, R., Sabir, M. *et al.* “Detection of autism spectrum disorder (ASD) in children and adults using machine learning”. *Sci Rep* **13**, 9605 (2023).

[4] V. Yaneva, L. A. Ha, S. Eraslan, Y. Yesilada and R. Mitkov, "Detecting High-Functioning Autism in Adults Using Eye Tracking and Machine Learning," in *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, no. 6, pp. 1254-1261, June 2020.

[5] Srivastava M, Srivastava P, Dubey AK, Srivastava P. A Narrative Review of Autism Spectrum Disorder in the Indian Context. “*Journal of Indian Association for Child and Adolescent Mental Health”*. 2023;19(4):336-343.

PUBLISHED PAPER AND CERTIFICATE

CERTIFICATE



GRADIVA REVIEW JOURNAL

An UGC-CARE Approved Group-II Journal

ISSN NO : 0363-8057 / Website : http://gradivareview.com/
Email : Submitgrjournal@gmail.Com

Certificate of Publication

Paper ID : GRJ/7749

This is to certify that the paper titled

Machine Learning-based Early Detection Of Autism In Children

Authored by

Miliyana Reena Dsouza

From

MITE Moodabidri, India

Has been published in

GRADIVA REVIEW JOURNAL Volume 10, Issue 12, December 2024.

UGC APPROVED JOURNAL

Indexed by Scopus

DOI: 10.37897/GRJ

crossref member
CROSSREF.ORG
THE CITATION LINKING BACKBONE

Teresa Gallart
Editor-in-Chief
Gradiva Review Journal

6.1 IMPACT FACTOR