AutoVision: An Automatic Machine Learning Tool for Computer Vision Projects

Session 2020-2024

By

Danish Khan Afridi
Qasim Khan

Bachelor of Science in Computer Science

Department of Computer Science
City University of Science & Information Technology
Peshawar, Pakistan
September, 2024

AutoVision: An Automatic Machine Learning Tool for Computer Vision Projects

Session 2020-2024

By

Danish Khan Afridi
Qasim Khan

Supervised by

Dr. Iftikhar Alam

Department of Computer Science
City University of Science & Information Technology
Peshawar, Pakistan
September, 2024

AutoVision: An Automatic Machine Learning Tool for Computer Vision Projects

By

Danish Khan Afridi (11971)
Qasim Khan (10615)

CERTIFICATE

A THESIS SUBMITTED IN THE PARTIAL FULFILMENT OF THE REQUIRMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE

We accept this dissertation as conforming to the required standards

(Supervisor) Dr. Iftikhar Alam	(Internal Examiner)
(External Examiner)	(Head of the Department)
(Coordinator FYP)	(Approved Date)

Department of Computer Science
City University of Science & Information Technology
Peshawar, Pakistan
September, 2024

Dedication

This thesis is sincerely dedicated to our parents, whose love, sacrifice, and inspiration have served as the cornerstone of our academic career. Their unwavering support and faith in our abilities have inspired us to reach higher goals.

Declaration

We confirm that this dissertation is our own work Except where indicated, the research we described in this dissertation is our own work.

Danish Khan Afridi
Qasim Khan
September, 2024

Abstract

Computer vision is a branch of artificial intelligence where computers can acquire and process external images. It is used in applications such as autonomous vehicles, medical imaging, and the field of robotics. Computer vision can be a way of uniting by manual or electronic means the endless stream of background information required (on paper) to tap into all that hidden demand, which is probably just as frightening for noobs as it must prove time-consuming and confusing even too seasoned industry professionals trying to stay current with cutting-edge traffic. In this thesis, AutoVision introduced, a web application for the general public to carry out computer vision tasks easily and with rich features. The main goal is to provide more advanced computer vision strategies for people, from noobs to professionals. AutoVision's user-friendly platform enables users to choose their ways of face detection, object detection, image segmentation, image classification, human pose detection, and other applications for any proper job execution. This project aims to remove the confusion that is often associated with computer vision and to create a more inclusive and collaborative environment where ideas, experience, and creativity can flourish. By doing so, AutoVision hopes to bridge the gap between different ways of computer vision practices and real-world deployment by making the process much easier and scalable, as well as unifying developers and students pursuing a career in computer vision and machine learning.

Acknowledgment

We would like to take this opportunity to first and foremost thank ALMIGHTY ALLAH for giving us strength and knowledge in the writing of this thesis. This project would not have been possible without the support of many people. Many Thanks to our adviser, Dr. Iftikhar Alam, who read our numerous revisions and helped make some sense of the confusion. His guidance and experience proved very helpful in the progress of this project. We would also like to thank all our fellow students and teachers for their fruitful suggestions.

Last but not least, we would like to thank our parents and my (Danish) uncle for always being supportive of our education.

Also, we want to acknowledge the second supervisor, other academic staff in our department, and technical or support staff in our department.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
CNN	Convolutional Neural Networks
CV	Computer Vision
CVPR	Computer Vision and Pattern Recognition
DFD	Data Flow Diagram
DL	Deep Learning
IEEE	Institute of Electrical and Electronics Engineers
ML	Machine Learning
OCR	Optical Character Recognition
SOTA	State of the art
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Overview

Computer vision (CV) is a field of artificial intelligence that trains computers to interpret and understand the visual world for a variety of exciting downstream tasks such as self-driving cars, checkout-less shopping, smart cities, cancer detection, and more. The field of CV has been revolutionized by deep learning over the last decade. Artificial Intelligence (AI) is another, however, strong technological wave that is flattening the world by providing the ability for a machine to perform cognitive functions, such as per-ceiving, reasoning, learning and interacting [1]. Machine learning is an evolving branch of computational algorithms that are designed to emulate human intelligence by learning from the surrounding environment. They are considered the working horse in the new era of the so-called big data. Techniques based on machine learning have been applied successfully in diverse fields ranging from pattern recognition, computer vision, spacecraft engineering, finance, entertainment, and computational biology to biomedical and medical applications [2]. Computer vision (CV) makes computers mimic as human and thus utilizes the concept of human vision. The human vision capabilities are brought in computers and huge number of operations can be performed by the computers. Image processing is an integral part of any CV system that involves acquisition of images and their processing for specific applications [3].

1.2 Background

In recent times, the field of computer vision (CV) has witnessed remarkable progress, bringing about significant changes in various industries and contributing to the development of innovative applications. However, as computer vision techniques and applications become increasingly complex and diverse, it has become challenging for beginners to enter the field. At the same time, experts find themselves in a constantly evolving landscape that requires continuous adaptation and familiarity with a wide range of tools and approaches.

The existing condition of the computer vision community is distinguished by a discernible divide between novices and specialists. A high learning curve and intimidation are common obstacles that novices must overcome to advance and become involved in the industry. However, because techniques and applications are evolving so quickly, it can be challenging for specialists to stay up to date on the latest developments and work well with their peers.

The field of computer vision is broad and includes many different applications, including segmentation, object detection, and image recognition. It calls for a broad range of abilities and in-depth knowledge of numerous algorithms. Thus, a solution that makes resource selection easier, makes knowledge exchange easier, and encourages teamwork is desperately needed in order to build a more diverse and effective computer vision community.

1.3 Motivation

The motivation of this project is to standardize CV's by providing a user-friendly website that empowers users to do CV activities with simplicity. The portal also hopes to promote knowledge exchange and enhance the CV industry.

1.4 Problem Statement

This thesis addresses the challenge that the variety of approaches in computer vision (CV) can overwhelm novices and complicate the work of experts. The lack of standardization and the steep learning curve often result in confusion for beginners and inconsistent outcomes for experienced users. By focusing on cohesive frameworks and simplifying complex methodologies, this research aims to make CV more accessible to newcomers while ensuring it remains robust for experts.

1.5 Proposed Solution

In proposed solution we developed a web-based platform with a user-friendly interface. Users have the option to select from a list of recommended algorithms and datasets for specific CV tasks. Additionally, they can choose their own data-sets as well.

1.6 Scope and Objective

Following is the scope and objective

1.6.1 Scope

The scope of this project focuses on the development of the website, creation of user documentation, and the establishment of a platform for knowledge sharing among CV enthusiasts. The project will focus on tasks such as face detection, object detection, image segmentation, human pose detection.

1.6.2 Objectives

The following are the project objectives.

- To develop a user-friendly Web-Application for CV tasks.
- To allow users to choose their preferred algorithms and datasets for optimized tasks.
- To establish a platform for users to share their results and insights with other developers/users.

1.7 Required Tools and Techniques

The tools and techniques to be used are:

1.7.1 Next-JS

The front end of the web application is developed using Next-JS, a very popular and powerful front end frameworks. Next-JS provides the ability of dynamic page construction with content by using React-JS, a widely used front end library [4].

1.7.2 Python

The algorithm implementation is done in Python using its vast libraries like (OpenCV. TensorFlow, Ultralytics, Mediapipe etc). Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. [5].

1.7.3 Streamlit

The algorithm implemented and models train with python are deployed to streamlit. Streamlit is a python library design to provide an interface for models trained in python [6].

1.7.4 Vercel

The web application is deployed using Vercel. Vercel is free, fast and simplest way to get an application up and running on internet [7]. It is perfect for developers who want to deploy code and not worry about managing the underlying infrastructure.

1.8 Thesis Outline

The remaining chapters in this thesis are arranged as follows: Software project planning is defined in Chapter 2, which also includes information on products and activities, resource allocation, risk management, and activity estimation. In Chapter 3, the system requirements are interpreted. Includes details regarding both functional and non-functional requirements. The content in Chapter 4, "Illustrated System Design," pertains to the database's logical and physical architecture. Testing is done in Chapter 5, which also includes information on white- and black-box testing and a conclusion.

Chapter 2

Software Project Planing

The project's whole activity plan is covered in this chapter, along with information on how each activity will be managed and observed during the project. A thorough explanation of the several tasks, including project management, milestones for different tasks, and software project planning, is also covered.

2.1 Project Feasibility Report

The feasibility report for "AutoVision: An Automatic Machine Learning Tool for Computer Vision Projects," a web-based machine learning tool for CV projects, shows promising outcomes in terms of technical, schedule, and specification feasibility. Given the project's nature as a tool that doesn't require database usage and its expected development time, it is clear that this project has the potential to make CV tasks easier.

- Technical Feasibility
- Resource Feasibility
- Market Feasibility
- Specification Feasibility

2.1.1 Technical Feasibility

Scalability is well-founded, of the web-based machine learning tool for CV projects. The project's technical feasibility is ensured by the availability of well-established tools and technologies like Next-JS, Python, and streamlit. Robust support for computer vision tasks is provided by algorithm implementation using libraries like OpenCV and TensorFlow.

2.1.2 Schedule Feasibility

With four sprints and an 8-month duration, the project schedule guarantees a rigorous and iterative development strategy. Two-month sprint cycles support planning, research, front-end, algorithm integration, testing, debugging, and documentation. A realistic and flexible timetable can be achieved by allowing revisions depending on feedback and changing requirements. With enough time for careful development, testing, and improvements, the team can complete "AutoVision: An Automatic Machine Learning Tool for Computer Vision Projects" and deliver a fully functional product in the allotted eight months thanks to the well-defined timeline, the project is completed.

2.1.3 Market Feasibility

AutoVision is a good fit for the market because it meets the needs of users with different levels of competence while addressing the increasing complexity and diversity of computer vision activities. Easy-to-use solutions that facilitate knowledge exchange and task implementation are clearly needed as computer vision continues to spread across several industries, including automotive, retail, and healthcare. AutoVision wants to take a big chunk of the industry by providing an easy-to-use web application and thorough documentation, enabling both beginners and specialists to efficiently employ computer vision.

2.1.4 Specification Feasibility

The suggested web tool's specs are in line with user-centered objectives and are still doable in the given time period. The Next-Js-based user-friendly interface encourages an intuitive user experience for CV tasks [4]. Flexibility and interoperability with a variety of datasets are ensured by the web-based tool's ability to use pre-trained models or build bespoke ML models. The specs help ensure that the project can be completed within the allotted eight months. The project requirements are realistic and in line with user objectives.

2.2 Project Characteristics

There are a few key ways that the project is distinct. It is intended to make things simple for users of web apps, particularly in the context of training custom machine learning models. Unlike other popular libraries like OpenCV, TensorFlow, etc., which could be difficult for beginners, "AutoVision: An Automatic Machine Learning Tool for Computer Vision Projects" places an emphasis on simplicity. Acknowledging that some users may lack experience, it offers an intuitive user interface. All that needs to be done to train a bespoke model for the user is to provide it a dataset, an algorithm, and some parameter tuning. The user will then be able to test our pre-trained models with their own data and download the model along with the code. Thus, the key to making CV accessible to everyone, effective, and user-friendly is to keep things straightforward.

2.3 Project Activities

Developing the user interface and implementing algorithms using Python libraries like TensorFlow, OpenCV, SciKit Learn, and PyTorch, as well as integrating them into Streamlit to connect to the web application, are the organized project activities for AutoVision as shown in Figure 2.1.

Table 2.1: Project Activities

Phase	**Activity**	**Outcome**
Project Initiation	Define project goals and initial requirements	Clarity on the project scope and objectives
Planning	Create detailed project plan and schedule	Well-defined project road map
Development	Implement features and functionalities	Seamless progress in task completion and issue resolution
Testing	Verify and validate features	Tested and verified features
Documentation	Create and update project documentation	Comprehensive project documentation
Review and Feedback	Demonstrate work, gather feedback	Stakeholder input and assessment of progress
Refinement	Adjust plans and features based on feedback	Enhanced project alignment with goals
Deployment	Deploy the final product	Fully functional tool ready for use

2.4 Activities Estimation

Activity estimation is the process of projecting how long different tasks and activities will take to finish at different stages of the development lifecycle. It's a crucial component of planning because it aids in resource allocation, goal setting, and improving project visibility overall for teams. The project's primary tasks are listed in the table that is being presented. Every row denotes a particular task along with related information such as the task's expected scope, output, and days-to-complete estimate as shown in Figure 2.2.

Table 2.2: Activities Estimation

Activity	**Scope**	**Expected Output**	**Estimated Duration (Days)**
Project Planning	Define project goals and schedule	Detailed project plan	10
Research	Investigate technologies and models	Research reports and recommendations	15
Front-end Development	Develop user interface with React	Functional and responsive UI	30
Model Integration	Integrate ML models for computer vision tasks	Integrated and tested models	25
Testing	Conduct thorough testing of all features	Verified and validated features	20
Documentation	Create and update project documentation	Comprehensive project documentation	15
Feedback and Review	Demonstrate work and gather feedback	Stakeholder feedback and improvement plan	10
Deployment	Deploy the final product	Fully functional tool	5

2.5 Process Model

Agile methodology is a project management framework that breaks down projects into small pieces called "user stories," and we work on these stories, commonly known as "sprints" [8].

Figure 2.1 shows that in the proposed system Agile model is used. And created the idea of a web application for CV tasks purposes, then collected some requirements about the idea. Then design a prototype of our web application the following steps are sketching the concept, wire framing, and UI/UX designing. After this, another step is implementation (coding and programming). After the implementation, the second last phase is testing the web application and models and then figuring out errors and doing some maintenance.

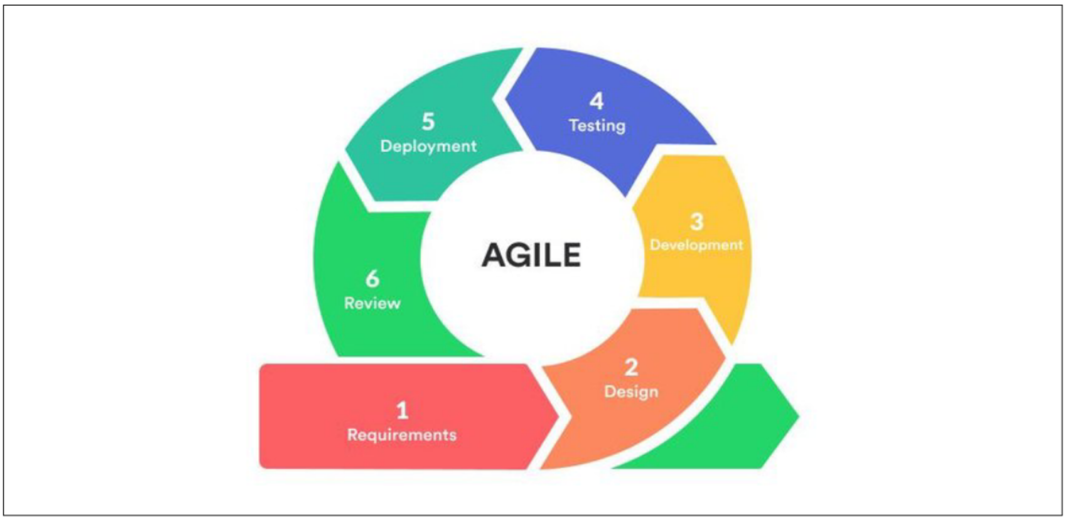

Figure 2.1: Agile methodology

The web App AutoVision, will be develop using agile methodology, which has multiple features. Using this methodology the developer divides each task into small pieces first of all it starts with planning phase which means it mentions which features they need and after requirement gathered developer can start user interface design phase. Whenever, they complete design , next phase is able to development phase which means that to set the features functioning. Whenever the development completed so that time developer start the testing phase to check the features are working correctly or not in testing phase the result is not correct so the developer restart the loop from the start and analyze it. Afterwards, testing is passed the developer finalized development phase and they get review. If review is positive the project should be launch and if the review is not satisfactory the developer restart the loop and make efficient project. By using agile method the team members can also change and improve project according to their needs.

2.5.1 Roles

The following roles are defined for our process model:

- **Product Owner**: The responsibility of defining project requirements and maintaining the Product Backlog is shared between team members, aligning with the project goals.

- **Development Team**: In the project plan, both team members take on the roles of designers and developers. They collaborate on creating detailed designs and implementing them. From planning to testing, their combined efforts ensure a comprehensive approach to the project's success.

2.5.2 Events

The following are the events of the project.

- **Sprint**: The duration of Sprints is defined as 2 months for each sprint.
- **Sprint Planning**: Both team members actively participate in Sprint Planning meetings to select tasks for the upcoming Sprint.
- **Daily Stand-up**: Daily stand-up meetings synchronize activities, discuss progress, and identify challenges or impediments.
- **Sprint Review**: At the end of each Sprint, completed work is reviewed, and feedback gathered, which is presented to the supervisor for review.
- **Sprint Retrospective**: Thinking about what went well and where it could be done better, and then taking actions to make things even better in the next stages of the project, guides how things can be done better in the next parts.

2.5.3 Supervisor Involvement

Supervisor plays a role as a stakeholder, offering valuable input during Sprint Reviews and guidance throughout the project. Regular check-ins with the supervisor ensure alignment with academic requirements.

2.6 Activities Risk

Table 2.3 shows the number of risks that can occur in project development. To avoid or minimize the risk, some risk activities include identification of risk, While creating an application, keep the end-user in mind the whole time. It is a fallacy to ignore end-users while designing an interface. The plan for project risk and removal of project risk are given and explained in below table.

2.6.1 Identify and Quantify Risks for Activities

It is the process of assessing identified risks to obtain data that can be used to decide how to respond to the risk in question. The purpose of quantifying project risks is to prepare for unexpected events in terms of costs, time and human resources. The Table 2.3 illustrates Identify and Quantify Risks for Activities.

Table 2.3: Identify and Quantify Risks for Activities

Risks	**Risk Description**
UI Design Problems	Challenges in creating effective and appealing designs.
Programming Difficulties	Difficulties in programming logic and implementation.
Schedule Risk	Delays may occur if resources or team members are not allocated properly, affecting the project timeline.
Project Delay	Risk of the project extending beyond the planned schedule.
Team Management	Poor team management can lead to project delays and impact overall progress.
Error in a Working Application	Errors may occur during actions like uploading images or text, changing passwords, or using the mobile back button.

2.6.2 Risk Reduction: Activity to Stop Risk Occurring

Risk reduction is about mitigating potential losses by reducing the likelihood and severity of a possible loss. Table 2.4 illustrates the risk reduction.

Table 2.4: Risk Reduction

Risk	**Risk Reduction**
UI Design Problems	Designed it through brainstorming.
Programming Difficulties	Take help from online tutorials or post questions on social communities related to the domain.
Schedule Risk	Negotiated with the supervisor to address the risk.
Project Delay	Group members will work on the project regularly to avoid delays in development activities.
Team Management	Ensure each individual is assigned a specific task for better management.
Error in a Working Application	Make the application system efficient and test it every time a change is made.

2.7 Inception

In the initial phase, the primary objective is to established for groundwork by clearly defining its goals and scope. The aim is to comprehend the purpose and direction of the project, ensuring a robust starting point for the work ahead. The following factor related to the inception phase:

- Identifying stakeholders

2.7.1 Identifying Stakeholders

Stakeholders are categorized into two main groups (direct and indirect). Direct stakeholders actively contribute to project execution, while indirect stakeholders are entities influenced by the project's outcomes.
The following stakeholders are identified for the proposed system:

a. Direct Stakeholders
The following are the direct stakeholders involved in our project:

- **Supervisor**: Providing valuable guidance and feedback, the supervisor helps the team navigate challenges and ensures the project aligns with academic requirements. Also, overseeing internal dynamics, ensuring the team operates efficiently.

- **Developer**: Engaging hands-on with coding and implementation, developers bring the project to life. Responsibility lies in writing quality code, collaborating with the team, and meeting project deadlines.

- **Designer**: Focused on the visual and user experience aspects, designers contribute to the project's aesthetics. Responsibility is to create an engaging user interface and collaborate with developers to integrate design elements seamlessly.

b. Indirect Stakeholders
The following are the indirect stakeholders involved in our project:

- **End-User**: Representing the ultimate beneficiaries, end-users are individuals or groups who will interact with and derive benefits from the final project. responsibility is to understand user needs and provide a user-friendly experience, shared among the entire team with a special focus on designers and developers.

- **Students and academic individuals**: Because of the tool's potential future use, impact on research and education, potential for collaboration, and important validation and feedback, students and academics are indirectly involved in the "AutoVision" project. Their involvement and input can impact the tool's evolution and adoption, even if they were not directly involved in its development. As such, they play a significant role in ensuring the tool's success and continued relevance in academic and research communities.

2.8 Resources Allocation

Resource allocation plan as shown below in Table 2.5. The team comprises two members, both designers and developers, along with a supervisor. A structured plan called Sprints is followed, during which the team works on specific tasks. Resource allocation involves giving each team member the right amount of time and energy to perform their job well. Team members spend their time creating and building, while the supervisor checks everything at the end of each Sprint to ensure it meets standards. This way, ensures everyone has enough time to do their part, and the supervisor ensures work meets standards before moving on to the next Sprint.

Table 2.5: Resource Allocation

Task	**Duration**	**Start Date**	**End Date**	**Assigned To**
Sprint 1 Planning	1 week	Oct 2023	Oct 2023	Team
Sprint 1 Development	3 weeks	Oct 2023	Nov 2023	Member 1
Sprint 1 Review and Documentation	1 week	Nov 2023	Nov 2023	Team
Sprint 2 Planning	1 week	Dec 2023	Dec 2023	Team
Sprint 2 Development	3 weeks	Dec 2023	Jan 2024	Member 2
Sprint 2 Review and Documentation	1 week	Jan 2024	Jan 2024	Team
Sprint 3 Planning	1 week	Feb 2024	Feb 2024	Team
Sprint 3 Development	3 weeks	Feb 2024	Mar 2024	Member 1
Sprint 3 Review and Documentation	1 week	Mar 2024	Mar 2024	Team
Sprint 4 Planning	1 week	Apr 2024	Apr 2024	Team
Sprint 4 Development	3 weeks	Apr 2024	May 2024	Member 2
Sprint 4 Review and Documentation	1 week	May 2024	May 2024	Team
Testing and QA	4 weeks	May 2024	Jun 2024	Team
User Feedback and Iterations	2 weeks	Jun 2024	Jun 2024	Team
Finalization	2 weeks	Jun 2024	Jul 2024	Team

2.8.1 Gantt Chart

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. In Figure 2.2 on the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar, the position and length of the bar reflect the start date, duration, and end date of the activity. This Gantt chart is for the 2023 and 2024 years.

Process	Duration in Months							
	Oct-23	Nov-23	Dec-23	Jan-24	Feb-24	Mar-24	Apr-24	May-24
Sprint 1								
Requirement Gathering								
Designing								
Development								
Testing								
Documentation								
Sprint 2								
Requirement Gathering								
Designing								
Development								
Testing								
Documentation								
Sprint 3								
Requirement Gathering								
Designing								
Development								
Testing								
Documentation								
Sprint 4								
Requirement Gathering								
Designing								
Development								
Testing								
Documentation								

Figure 2.2: Gantt Chart

Chapter 3

Software Requirements Specification

With AutoVision, users may analyze images and videos using advanced machine learning skills. The application is web-based. In order to make activities like object segmentation, plant disease detection, brain tumor classification, bone fracture detection, and bird species identification easier, the system attempts to offer an intuitive user interface. With their own datasets, users can also train custom machine learning models that make use of algorithms like image classification and regression. Important features include easy-to-use interfaces for training models and pre-processing datasets, extensive documentation to assist users, and smooth integration with GitHub and Google for user identification. Built on top of cutting-edge web technologies like Next.js, Tailwind CSS, and TypeScript, AutoVision interfaces with machine learning frameworks based on Python, such as TensorFlow and PyTorch, for dependable performance.

3.1 Requirement Elicitation

The goal of the "AutoVision" project is to create an intuitive web tool for CV tasks. In order to do this, a thorough requirements elicitation process was used to identify important parties, such as end users, developers, supervisors, and designers. The functional requirements of the tool focus on a variety of algorithms, datasets, model selection, and robust customization features. It also includes the ability to download images and generate Python code for the trained model. Dependencies concentrate on browser compatibility and server uptime, whereas assumptions emphasize the need for reliable internet access and accurate data entry. The underlying web technologies and the requirement for a steady internet connection to guarantee smooth operation are the main constraints of the project. The group cooperatively addressed possible obstacles using brainstorming as the elicitation technique, making sure the application satisfies user needs and project goals. The two major perspectives of System Requirements are Functional Requirements and Non-Functional Requirements.

3.2 Functional Requirements

The purpose of the AutoVision approach in adoption is to have a computer vision function that can provide multiple use cases. Users should also be allowed to upload image datasets for the tasks selected to perform. It allows users to customize parameters and settings which are very important as customization can directly work in the favor of achieving best possible outputs. Note: additionally the users should be able to save the results of their computer vision projects as well as share them.

The functional requirements are as follows:

3.2.1 Various Categories of CV Tasks

It means that the autovision system must have a rich set of computer vision operations among them are object detection, image classification (this includes facial recognition), pose estimation, semantic segmentation etc. A model variety, and variants of each malting category a meeting needs for implementation and data analysis.

3.2.2 Selectable Computer Vision Task Categories

A user should be able to navigate the diverse computer vision task categories with an intuitive menu or interface. These descriptions or snapshots should enable users to identify the categories best suited for their data and analytical needs.

a. Model Catalog (Select Pre-trained Model)

The Model Catalog feature allows users of AutoVision to select from a range of pre-trained models for various computer vision applications. Some of these tasks are identifying bird species, classifying brain tumors, detecting plant diseases, object segmentation and bone fracture detection. Users choose an appropriate task from the library, and then apply a trained model for that particular type of analysis on their data.

b. Train Custom Model (Select Custom Model Training)

Train customized Model With this feature the user can train their own model based on a requirement by User. With some datasets and methodologies, we can train a classification or regression model or image recognition. With the systems flexibility, users can choose pre-existing datasets or upload their own to train custom models. Users can now build models that are specialized to their data and analytic requirements.

3.3 Non-Functional Requirements

The AutoVision system's non-functional requirements (NFR) cover a number of important areas.

3.3.1 Performance

For fast reaction times, the Auto Vision system must alter its settings to make sure that computer vision jobs complete on time in spite of large datasets. The system should be able to work with multiple users at the same time without any major performance lag for user experience. This means things like fine-tuning your server responses, handling data processing with algorithms that are as effective and efficient in real-world conditions.

3.3.2 Usability

The user interface of AutoVision needs to be easy and intuitive in order for all kinds of users with different level of technical background can extract value out of it. This should be common image of design features, standardized labeling and immediate previews of the results for seamless execution. For users to learn about how they can use the application, contextual help options and tool tips might prove beneficial. These characteristics have been designed so as users can engage the systems seamlessly whether you are a first-time teacher or an experienced trainer.

3.3.3 Reliability

And there shall be no downtime for the AutoVision application. We detect the erroneous inputs and unexpected user actions using error-handling mechanisms. The system must be able to provide accurate results again this is true with the handling of whatever inputs it might grab.

3.3.4 Scalability

AutoVision system has to be scalable in order to support the growing number of users. This is possible when using scalable architecture and the appropriate technology to process an ever-growing number of requests as well large amounts of data, while maintaining velocity. It will need load balancing and efficient resource management to make the scalability seamless.

3.3.5 Compatibility

The AutoVision application should work with every major web browser, such as Edge, Firefox and Chrome. It also needs to be responsive (works on all platforms: PC, tablet and smartphones) so as that anyone can view it in their best comfort. The system has to provide more functionality when using modern browsers and have a smooth transition towards old ones. Through interoperability, the system is easily accessible by users on all platforms and devices.

3.4 Use-Cases/ Hierarchical Input Process Output

Based on the Use-Case Diagram as shown in Figure 3.1 for AutoVision users can perform any type of computer vision task and how they are going to interact with the system, using this. You pick your favorite of a handful activities, such as object annotation, plant disease recognition or brain tumor classification. The 2nd step is the built-in system of verification uploaded dataset of pictures for compliance with selected task and it can be processed without errors. Once the validation is done, it uses algorithms for computer vision analysis as per your request. The parameters of the model can also be altered by users as required in order to make the analysis suitable for them. Users then download the results for offline storage or reference once their process is complete. Having the underlying code in a larger context allows us to understand everything more completely and could allow for augmenting it if necessary. The broad platform formed by AutoVision is the answer to various computer vision researches as it efficiently meets its customer's needs.

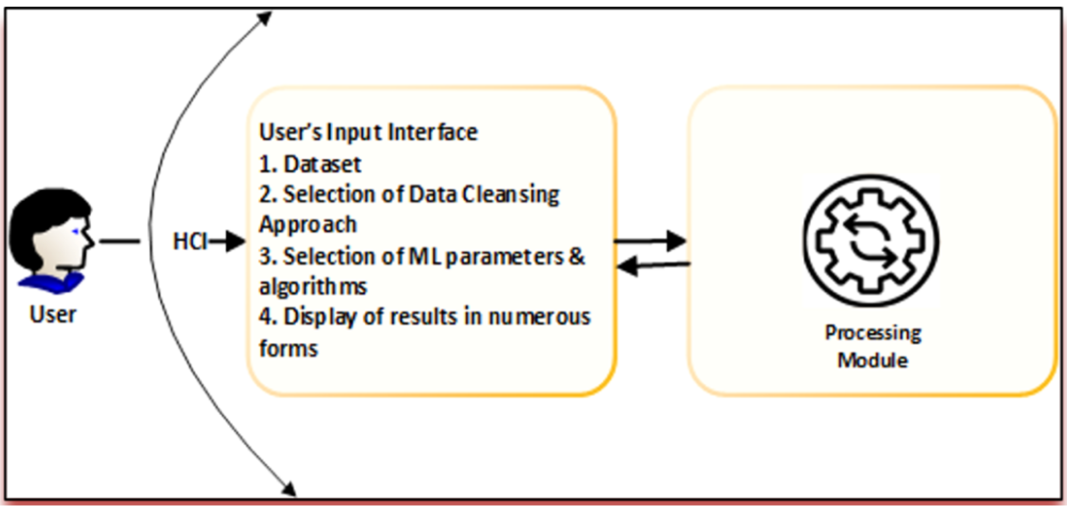

Figure 3.1: Use-Case Diagram

3.5 Assumptions and Dependencies

Some of the key assumptions and dependencies on which AutoVision project hinges upon includes aspects that are fundamental to ensure their success in functioning properly as well user satisfactions. The online app is best served to users who can consistently access the internet on their laptops or mobiles, no operating system limitations, it is built for the web but you need a server that always online to use this. The assumption is that users will use many different web browsers with the system. In addition, the system will work better if users enter correct data. These assumptions point to the app's reliance on an internet connection and server being online while simultaneously reiterating how important correct user input is for this program to work.

3.5.1 Assumptions

The following are the assumptions:

- Users have internet access available on their mobile/computer devices.
- There are no limitations regarding the operating system, but AutoVision operates on a server that needs to be always online.
- Users can access the system through any internet browser.
- Users will enter valid data into the system.

3.5.2 Dependencies

The following are the dependencies:

- The availability and reliability of internet connectivity are essential for users to access the web application.
- The server hosting AutoVision must be constantly online to ensure uninterrupted access to the application.
- The web application must be compatible with various internet browsers to accommodate users accessing the system from different devices.
- The functionality and accuracy of AutoVision depend on users inputting valid data into the system. Any invalid data could affect the performance and output of the application.

3.6 Overall System Description

With a uniform home page interface, AutoVision's entire system is made to offer every user a consistent and easy-to-use experience from the moment they launch the application. Users can choose from a variety of computer vision tasks within the application, providing a wide range of analysis choices.
Whether a user opens the app for the first time, or lands on AutoVision's home page layout - each element of their system is designed to provide an uniformed (and easy-to-navigate) UX. Within the application, users have access to a number of computer vision tasks they can task multitude -wise analysis away.
In addition, the system supports upload of image datasets that users can use to perform analysis on their own data. Users can customize the parameters of model to meet their own needs and preferences. It also simplifies adding or copying the code beneath, to make it easy for you to include in other projects/workflows. This offers even greater flexibility and accessibility to the program's analyses as it allows users, among other things, download analysis results for use at a later time or to others.

The Features Of The Application are:

- For a user, all of them open to the same Home Page.
- The choice to pick a different computer vision job.
- For particular tasks, the system allows for pre-trained model selection.
- Also, the system provides a functionality where you can select model pretrained for specific tasks.
- Unique model training may be achieved by utilizing either such methods or custom data from the user.
- Users have the ability to customize the model parameters to suit their own tastes through customization choices.
- The system facilitates easy generation and copying of underlying code.
- Users can download analysis results for further use.

3.7 System Constraints

The main limitations of AutoVision's system are related to its reliance on different technologies for both front-end and back-end functions. The application's functionality and user interface are supported by these technologies taken together. For smooth operation, having a reliable internet connection is a basic necessity. Users may suffer disruptions in their experience if they are unable to access the features and services of the program due to unreliable internet connections. Furthermore, network latency and browser compatibility may have an impact on the program's performance; therefore, it is necessary to make sure the application works with a variety of web browsers and to fix any network-related problems. Through the prioritization of these system limitations, the application can endeavor to provide a dependable and uniform user experience throughout a range of usage scenarios.

There are two types of system constraints which are given below:

3.7.1 Software Constraints

The suggested application is constructed using a number of different technologies. ESLint, TypeScript, Tailwind CSS, and Next.js are used in the front-end development process. Along with Google Colab, Jupyter Notebooks (ipynb), and machine learning/deep learning packages like TensorFlow, PyTorch, and Scikit-learn, Python is used to train algorithms. Libraries like as OpenCV and PIL are utilized for image handling, and Matplotlib and Seaborn are used for plotting. An interface, deployment management, and resource management are all handled by Streamlit. On Vercel, the website is operational. Therefore, users must have access to a working internet connection in order for the application to function properly. The functions and services of the application might not work or function poorly without a dependable internet connection. Furthermore, because of the application's dependence on these technologies, network latency and browser compatibility could have an impact on its performance. Providing a consistent user experience requires addressing potential network difficulties and guaranteeing compatibility across various web browsers.

3.7.2 User Constraint

The application is made to accommodate a wide range of users, including both novice and expert users. Users who want to use the application's capabilities and carry out different computer vision tasks need to have a web browser that can render the application's content. This stipulation guarantees that users, irrespective of their technological proficiency, can engage with the application. The application also attempts to make itself easy to use by offering a user-friendly experience to all users. The program is designed to be simple to use regardless of the user's level of knowledge with technology. In addition, taking into account probable variations in user preferences and accessibility requirements, the application should include customizable options and conform to accessibility guidelines to satisfy a range of user needs.

Chapter 4

Software Design

In this chapter, we will provide an extensive analysis on the latter by discussing system architecture as a whole with specific emphasis placed upon its various structural parts and their relationships. The chapter then decomposes the architecture into its lowest-level components and full explanations on how they interact with each other to provide functionality for applications that rely on this system. In this chapter, to describe dynamic features of the system it used different models including activity diagrams, data flow diagram(DFDs) and sequence diagram (use-case-diagram DataAccessOperation ActivityDiagram Sequence Diagam) [9] [10] [11] [12]. These tools are crucial as they help you visualize the process flows, data flows and between-component interactions over time. Those diagrams explain not only how the system reacts on certain scenario, but also helps to understand at higher level what is happening underlining on logic and operation flow which valuable mainly for developers without programming knowledge or just read a bit of it. Beyond the architectural overview, there is a lot of insightful commentary on designing interfaces in this chapter; specifically making certain user interaction with the system. This section transcends just appearance, centring on the usability and accessibility as well as overall user experience in which the interface design should reach. The chapter presents a complete idea of how the interface will enable user interactions through discovering the worked out design decisions and guarantees an intuitive and friendly product. The discussion here stresses the significance of pampering interface with users want them to interact, symbolizes in whether it is capable enough for target activity or features that assist end-users to get done what extremities they anticipate.

4.1 Features Diagram

Following are the four attributes mentioned in Figure 4.1

- **Register**
  This is used for registering a new user using Clerk [13].

- **Authentication**
  This is used by the admin to verify the details of registered users and whether to

approve or reject them.

- **Login**
  User enter login details and press login and application will check data in database and fetch the profile.

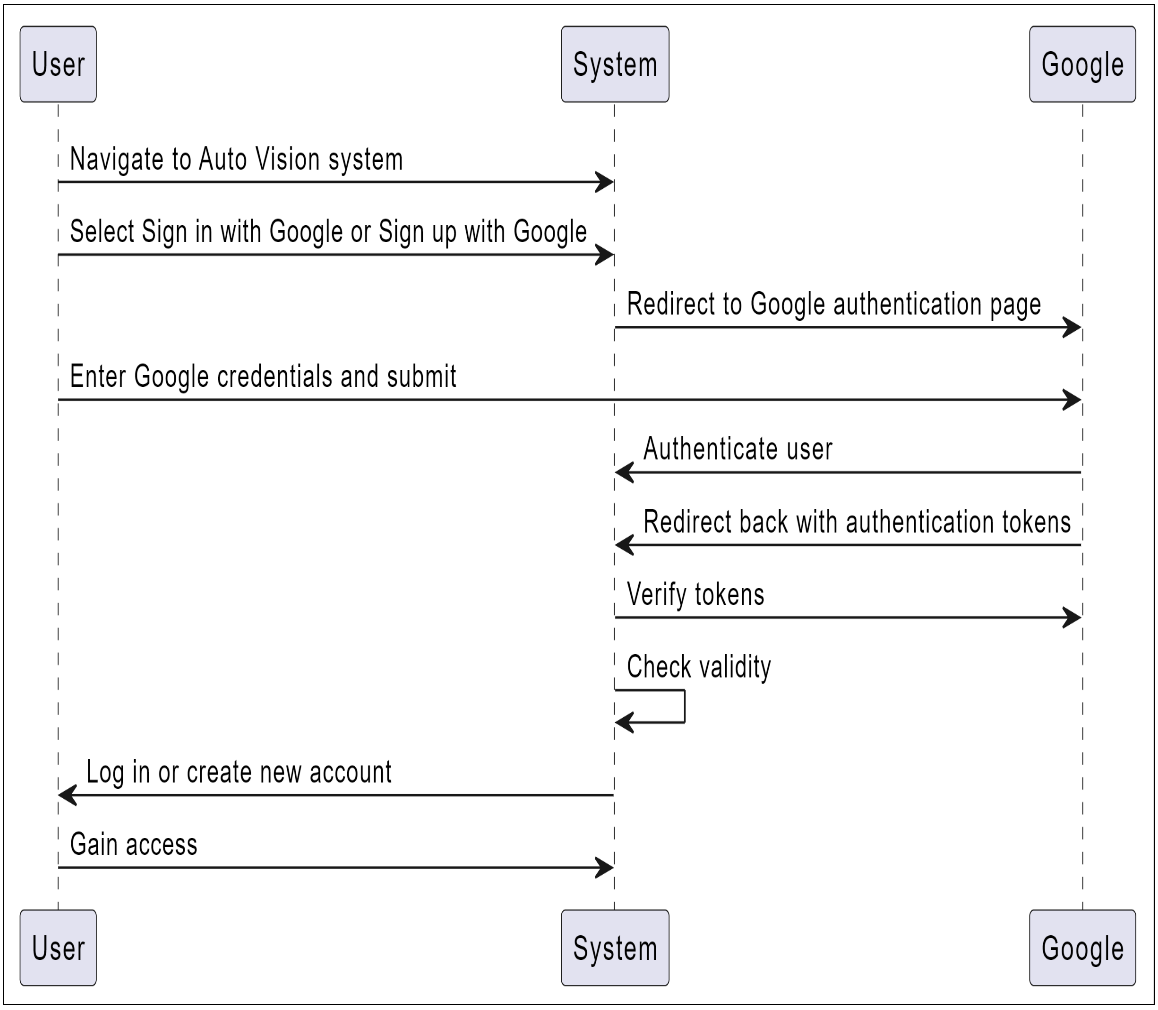

Figure 4.1: Feature Diagram

4.1.1 Activity diagram for uploading Image/Video file

The activity diagram for a user to upload a picture or video file for the AutoVision system to analyze is shown in the Figure 4.2. By using the file upload function, the user starts the procedure. After that, they use the online interface to submit the chosen file that they have chosen from their local storage. After confirming the file's format, the system accepts the file and stores it in the relevant data repository. The user receives a notification from the system after a successful upload, and the file is ready for additional processing, like model training or prediction. The sequential exchanges that take place between the user and the system during the file upload process are depicted in this diagram.

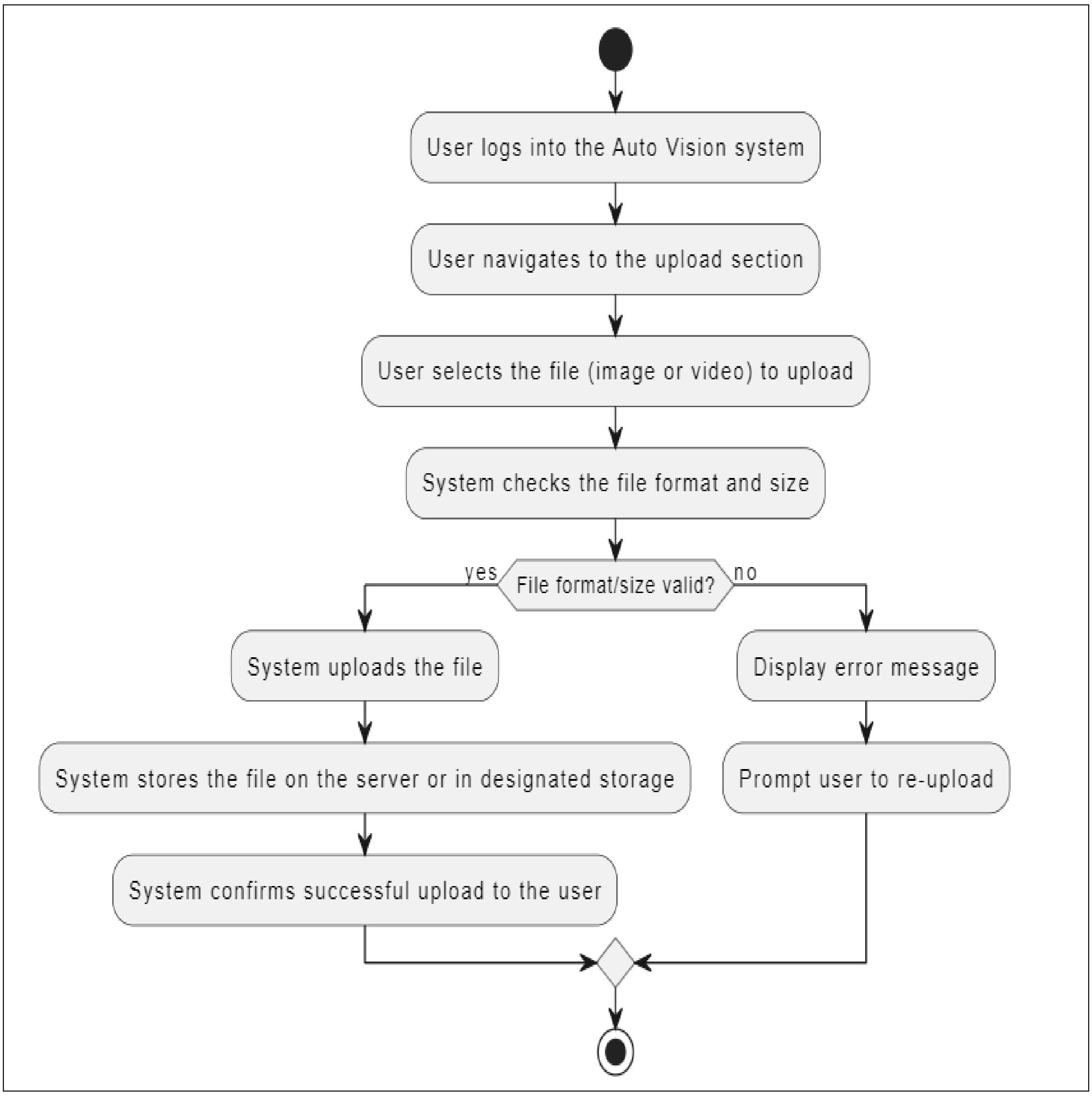

Figure 4.2: Activity diagram for uploading Image/Video file

4.1.2 Activity Diagram for Running Model Inference

Figure 4.3 illustrates how to select a pre-trained model in the AutoVision application. After accessing the model selection interface, the user looks through the pre-trained models that are accessible and groups them based on the features they have, such as object segmentation, plant disease detection, bone fracture diagnosis, and bird species identification. When a user selects to the desired model, then system will get method information and present it back as API documentation for usage. After the user confirmed their selection, the model is loaded by system and can be used to custom training or forecasts etc. The diagram shows how the user uses a model selection function and what happens in the system.

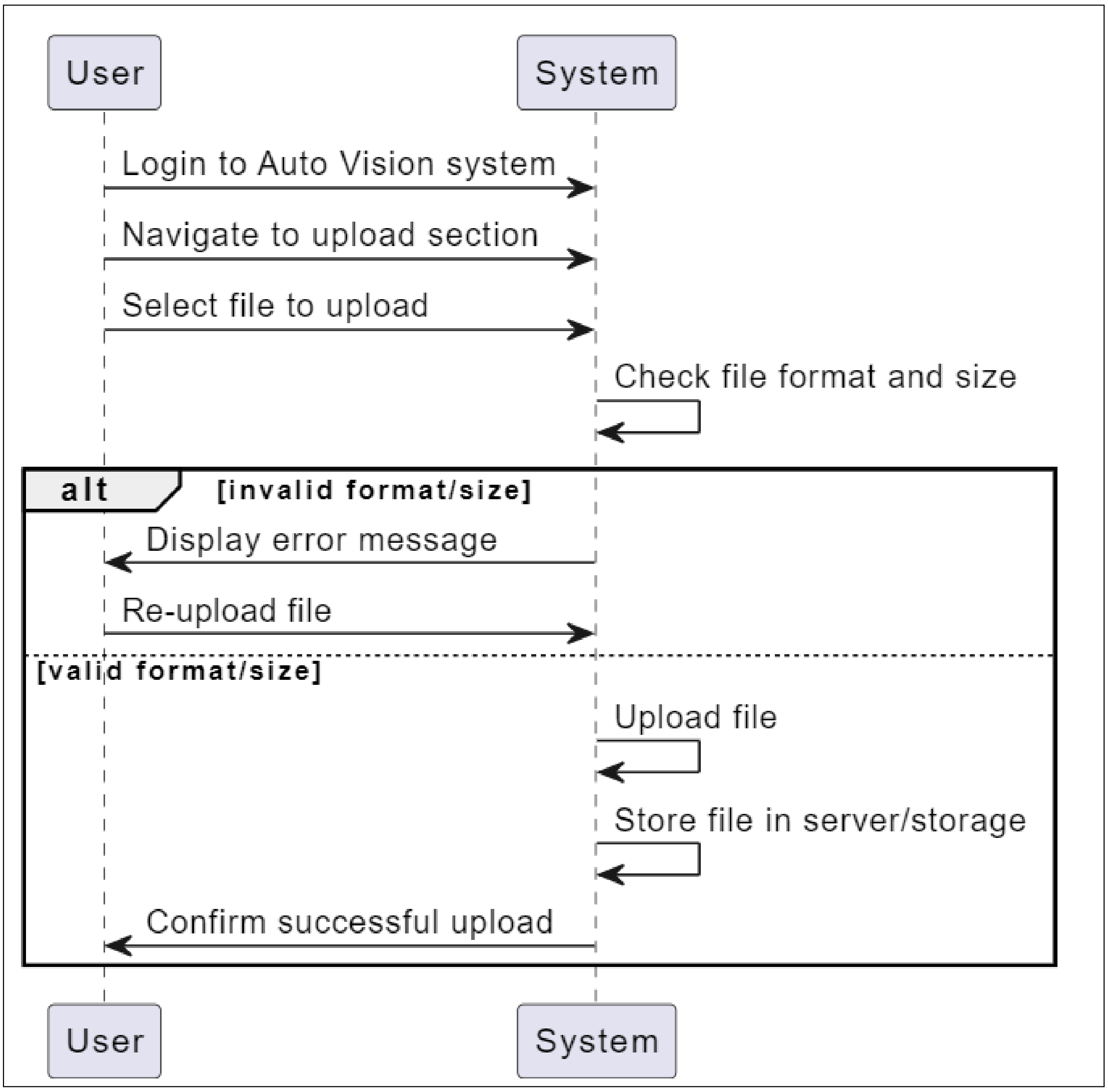

Figure 4.3: Activity Diagram for Running Model Inference

4.1.3 Activity Diagram for Training Custom Model

The process of training a custom model using autovision technology is illustrated in figure 4.4. User starts train a new method then, they bring a dataset that allows them to perform image classification or regression generalization - whatever the task was planned for. The dataset is verified by the system, after which various algorithm options are presented to choose from for better predictions. The user selects an algorithm, chooses the hyper parameters for training and starts. The data is prepared and then the system trains a model, providing you with real-time updates on its progress. Once the training is finished, a notification tells you that your trained customizations will be saved. The following diagram shows the entire process in training custom models.

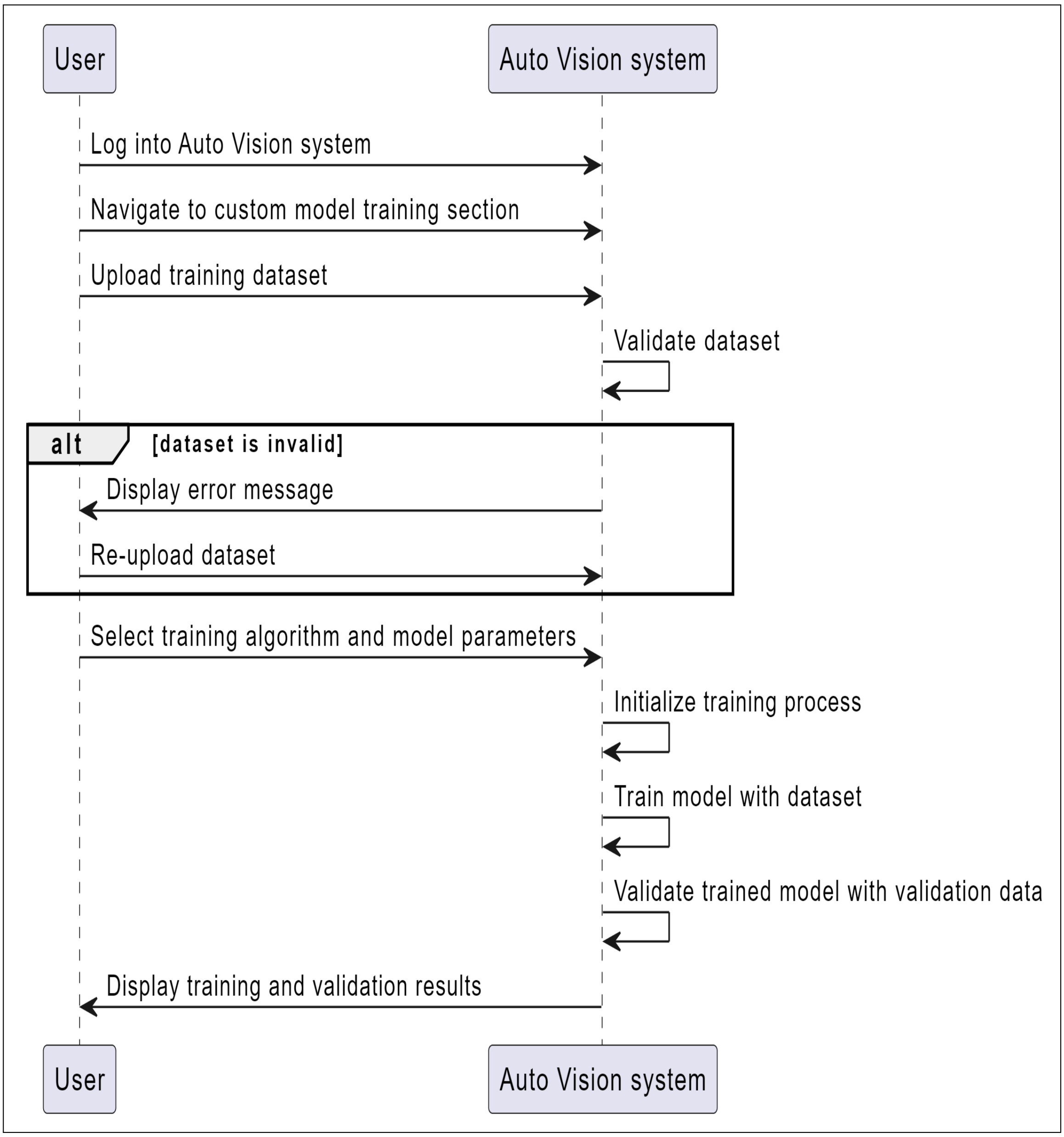

Figure 4.4: Activity Diagram for Training Custom Model

4.1.4 Activity Diagram for Selecting Pre-trained Models

Figure 4.5 shows the pre-trained model selection process. In the model selection interface, different pre-trained models are opened according to which features each has like object segmentation, plant disease detection etc. Once a desired model has been clicked upon by the user, we get information about that model and show it to the user. When a user makes the selection, The system loads the model into memory and performs actions like custom training or forecasting using that selected model.

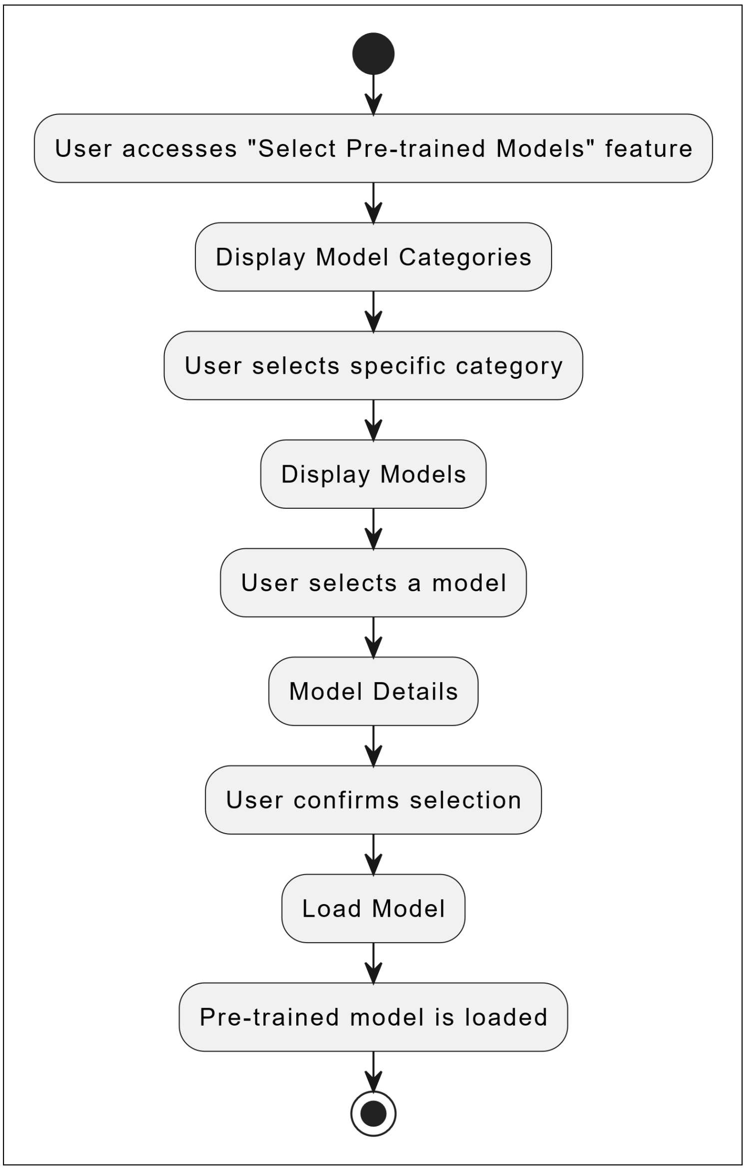

Figure 4.5: Activity Diagram for Selecting Pre-trained Models

4.1.5 Activity Diagram for Error Handling

Figure 4.6 shows how to sign in with GitHub or Google when running AutoVision. This user is accessing the sign-in/sign-up page and when given an option to choose an authentication method, they are selecting either Google or GitHub. System redirects the user to login page of respective provider for entering credentials. It returns to AutoVision with a token between successful authentication. The system creates or updates their account, validates the token and grants access to use the application. This flowchart shows the security team a safe and usable way for third-party services to authenticate.

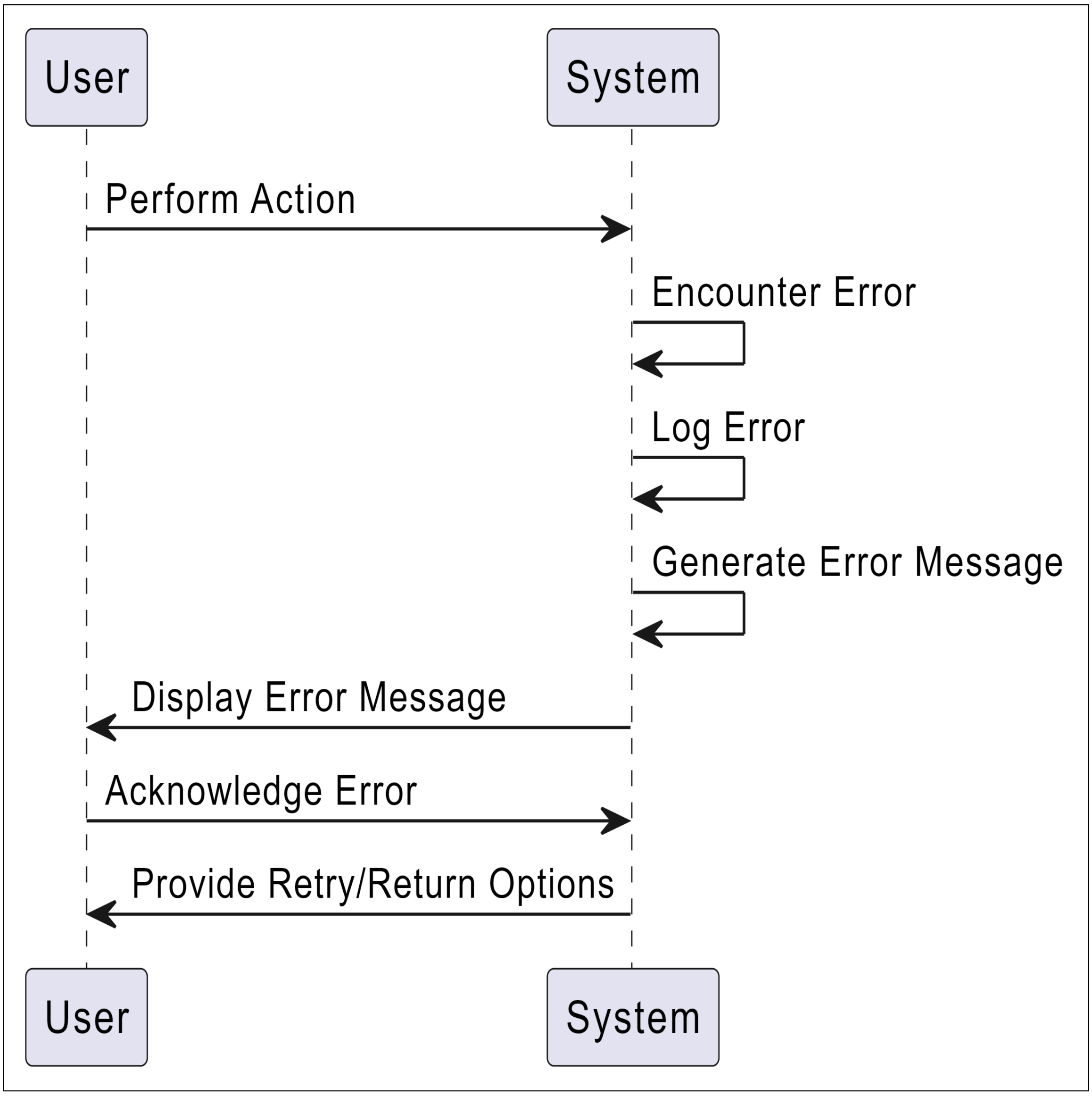

Figure 4.6: Activity Diagram for Error Handling

4.1.6 Activity Diagram for Downloading Results

Figure 4.7 is an action diagram that shows a user who uploads an image or video file to have it seen by the AutoVision system. It is instigated by user, making use of file upload option. The users the chooses the local file and submits it to them using a web interface. When the system confirms your file format on arrival, it takes in and saves data records an appropriate repository. Where system triggers then prepare the file for further operations like model training or prediction once it gets successfully uploaded by user. Following is the sequence diagram illustrating a series of conversations happening between user and system to upload file.

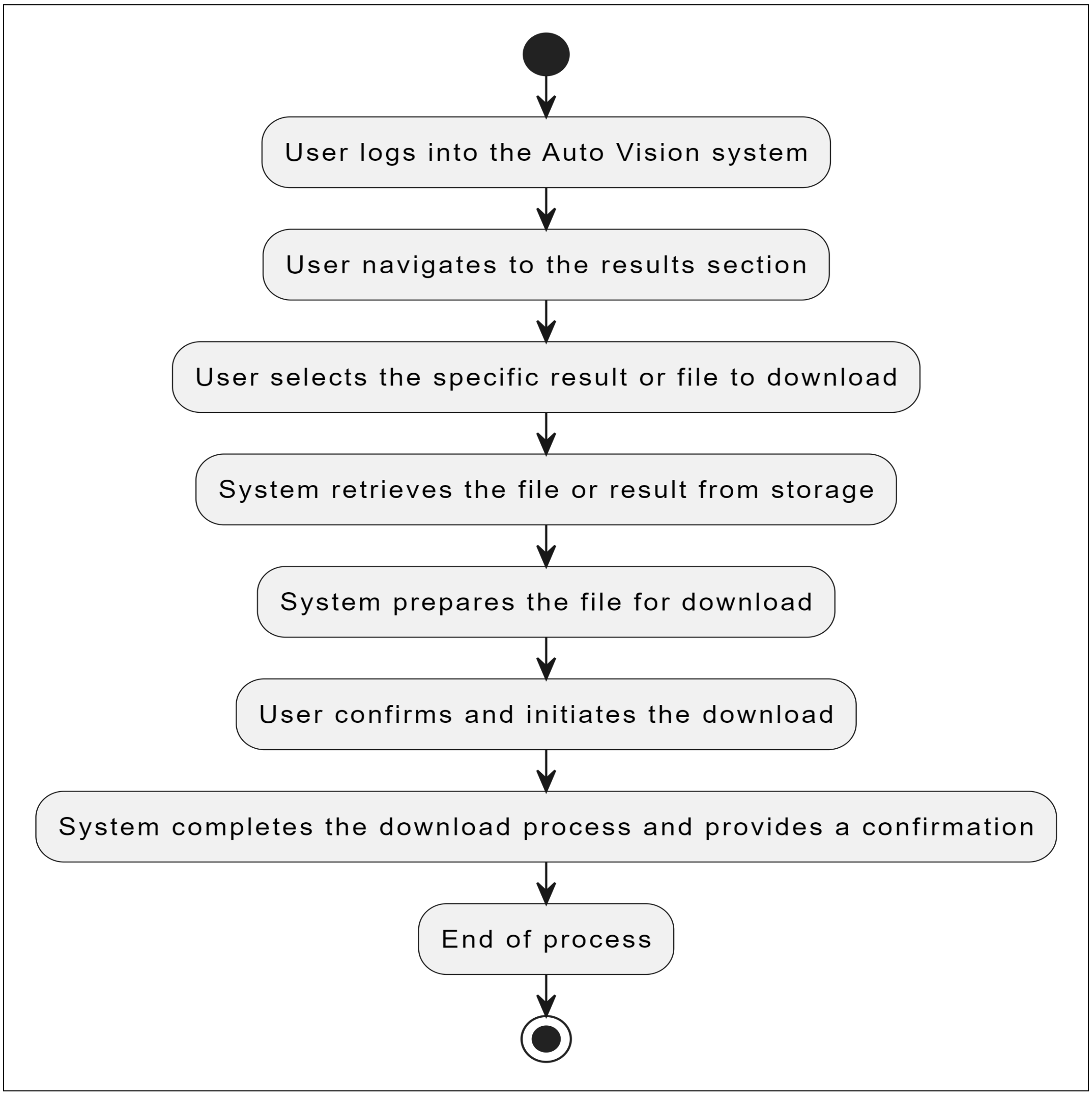

Figure 4.7: Activity Diagram for Downloading Results

4.2 Data Flow Diagram (DFD)

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and non-technical audiences, from developer to CEO [14].

4.2.1 Level 0 DFD

The AutoVision system is given a high-level perspective shown in figure 4.8, which portrays it as a single process interacting with outside entities. The User, the Authentication Provider (Google/GitHub), and the Model Storage are the main entities. To upload photos or videos, choose models, and train or predict using models, the user interacts with the autovision system. Requests from users to log in or sign up are handled by authentication providers, and trained models are stored and retrieved via model storage. This diagram establishes the basic data flow between the system and external entities.

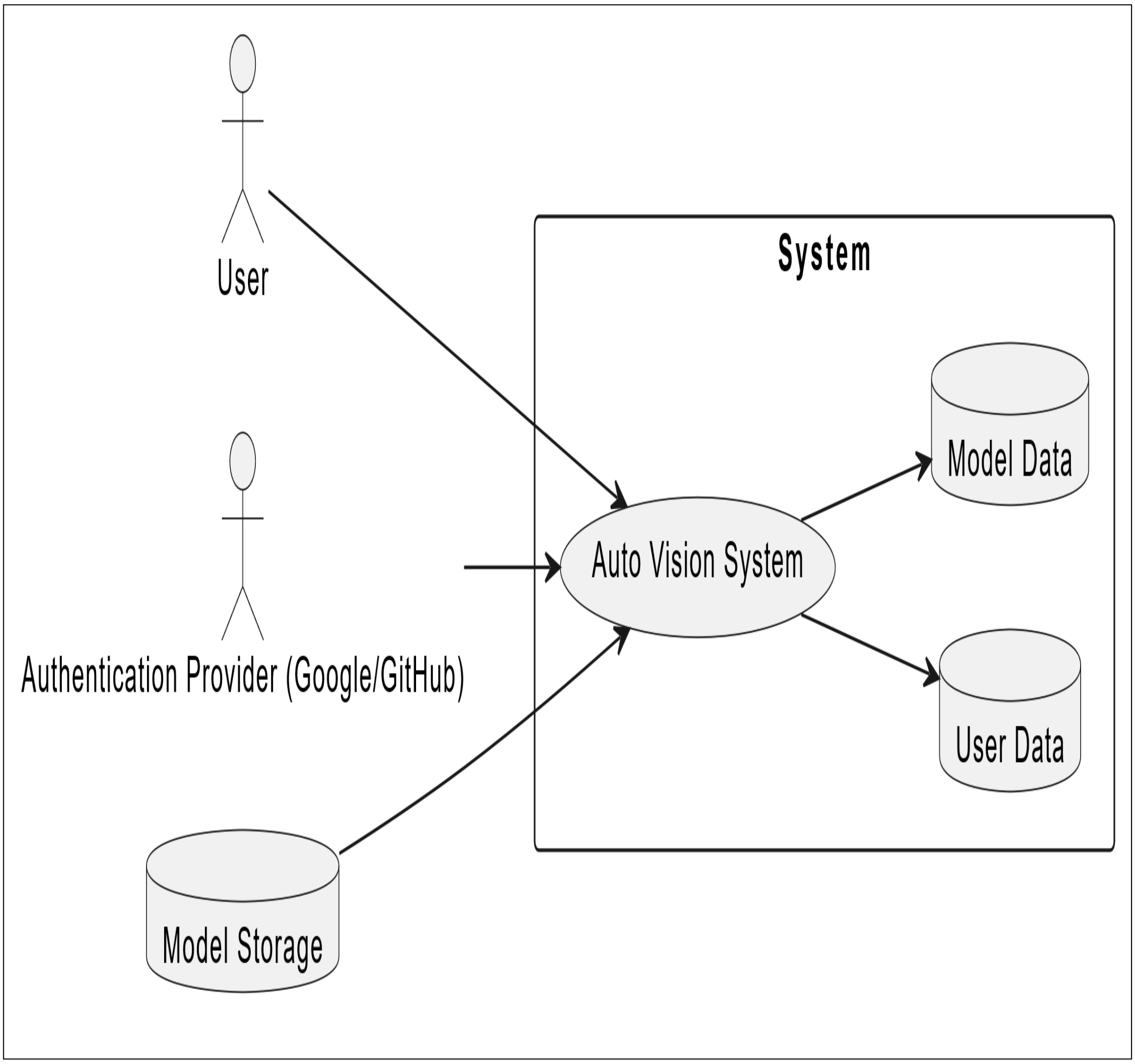

Figure 4.8: Level 0 Data Flow Diagram

4.2.2 Level 1 DFD

Figure 4.9 breaks down the main process into four sub-processes: User Authentication, Model Selection, Model Training, and Model Prediction at DFD Level 2. User Authentication needs to sign in using Google or GitHub. Model Selection Offers the ability to browse over pre-trained models Data upload and Training new model. These processes are included under the training of a model Model predictions - Taught models can predict new data. These in turn rely on data stores that handle saving and retrieving the user states eg User Data, Model state, etc. That DFD will give a more detailed view of the main functionalities that your system can perform.

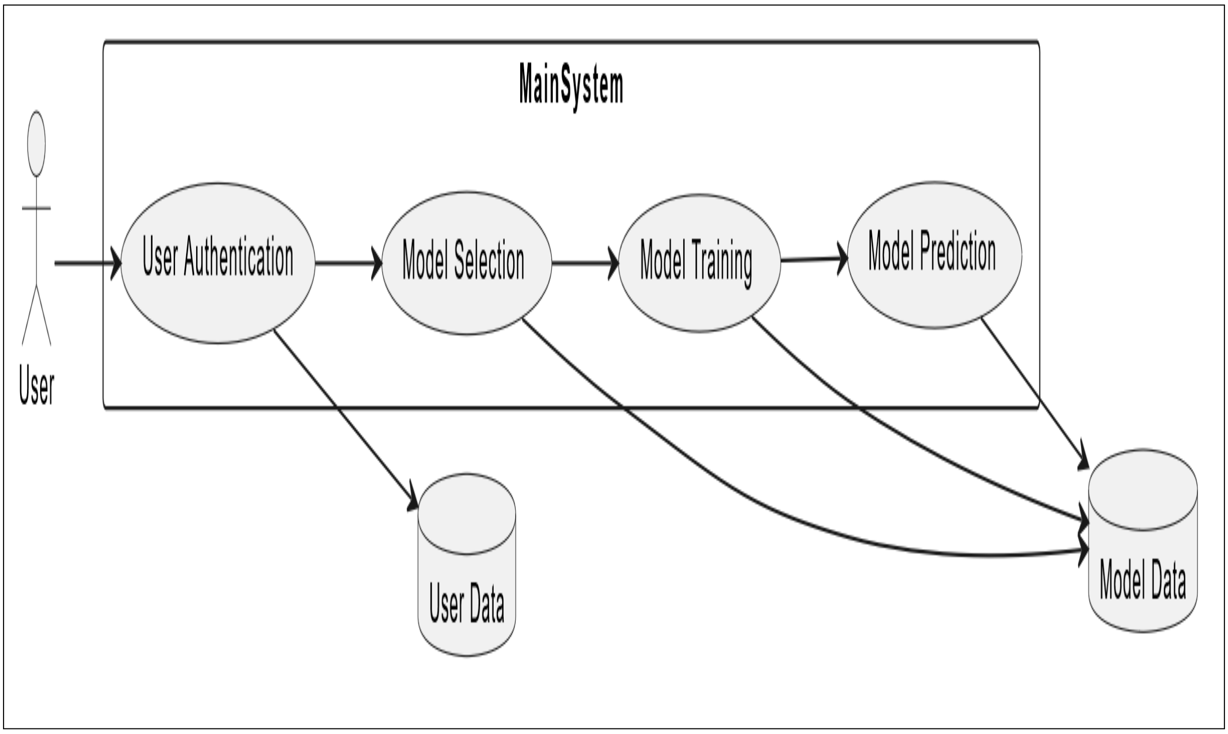

Figure 4.9: Level 1 Data Flow Diagram

4.2.3 Level 2 DFD

Level 1 DFD can be divided into more focused actions in the next level shown in figure 4.10. User authentication. This can be broken down into two parts: Register both the Google type and the GitHub one. Using pre-trained Models is just another part of Model selection. Model training, which involves uploading images and videos using particular algorithms. Model prediction is using trained models to make predictions. Each operation is related to exactly one data exchange so that we can guarantee a correct thread of available data and system operation. This graphic dives deep into the connections and relationships between each of these components.

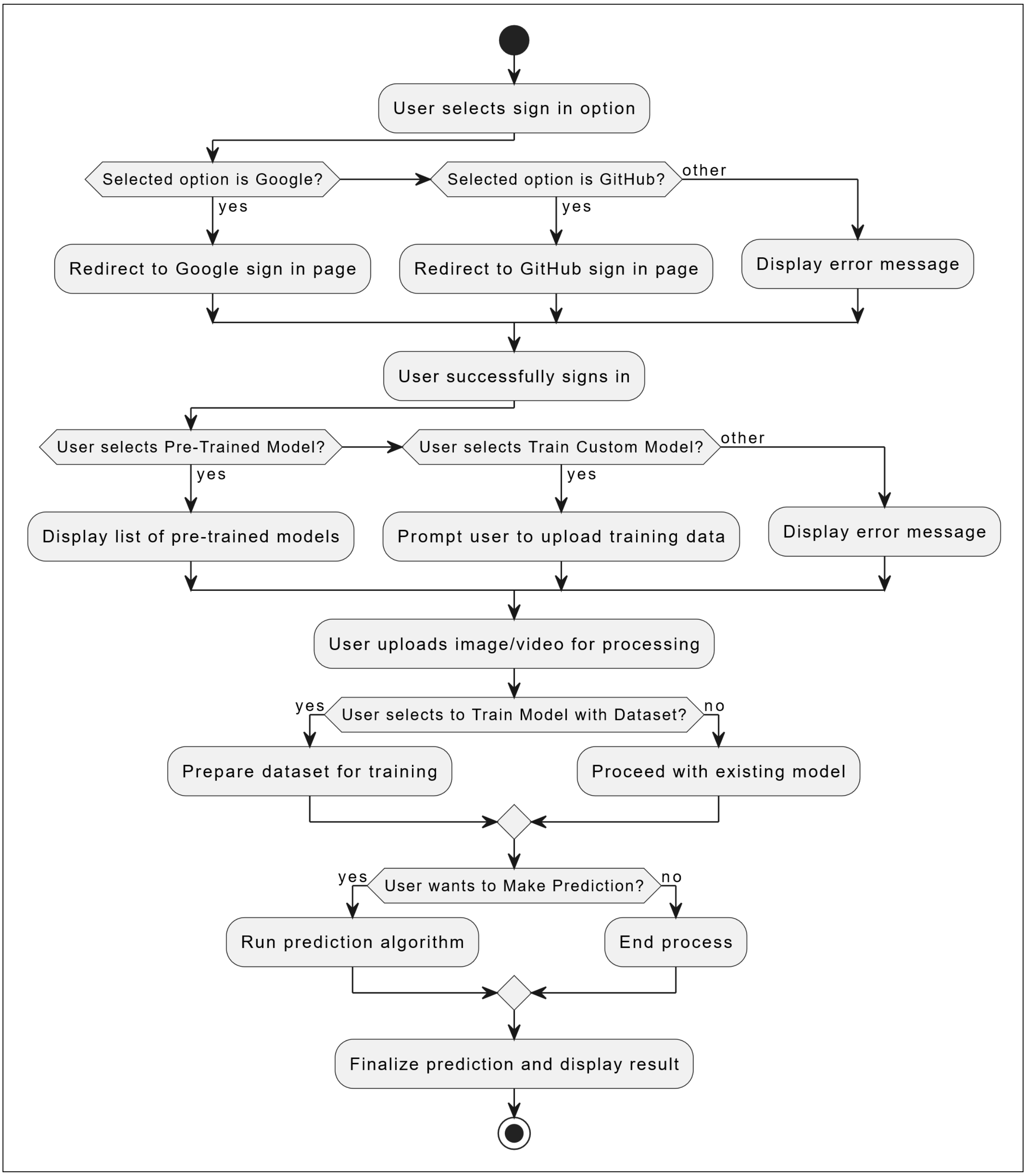

Figure 4.10: Level 2 Data Flow Diagram

4.3 Interfaces of the Application

The interface of AutoVision should be very familiar and intuitive for dealing with its functionalities. AutoVision introduces users with a neat and clean structured home-

based overview page to centralize your navigation. The simple-workflow user interface gives top billing to the options for training custom models and selecting a pre-trained model.
Under pre-trained models, the interface provides sorted options like Object Segmentation, Plant Disease Detection, Brain Tumor Classifier, Bone Fracture Detection and Birds Species Identification. The items also have specific, detailed descriptions it in each category for customers to get peruse. create or select the model that fits their requirement. In addition, since it allows users to upload datasets and sample methods for tasks like regression or image classification - the interface is designed to aid in training custom models. The blog gives you various options as well, as a step-by-step guide on how to choose settings for training data designing datasets.

4.3.1 Landing Page

Figure 4.11 which gives access to AutoVision and has a very user-interactive interface. This makes the main features of this software accessible by users with its obvious options to choose a pre-trained model selection and train your unique models.

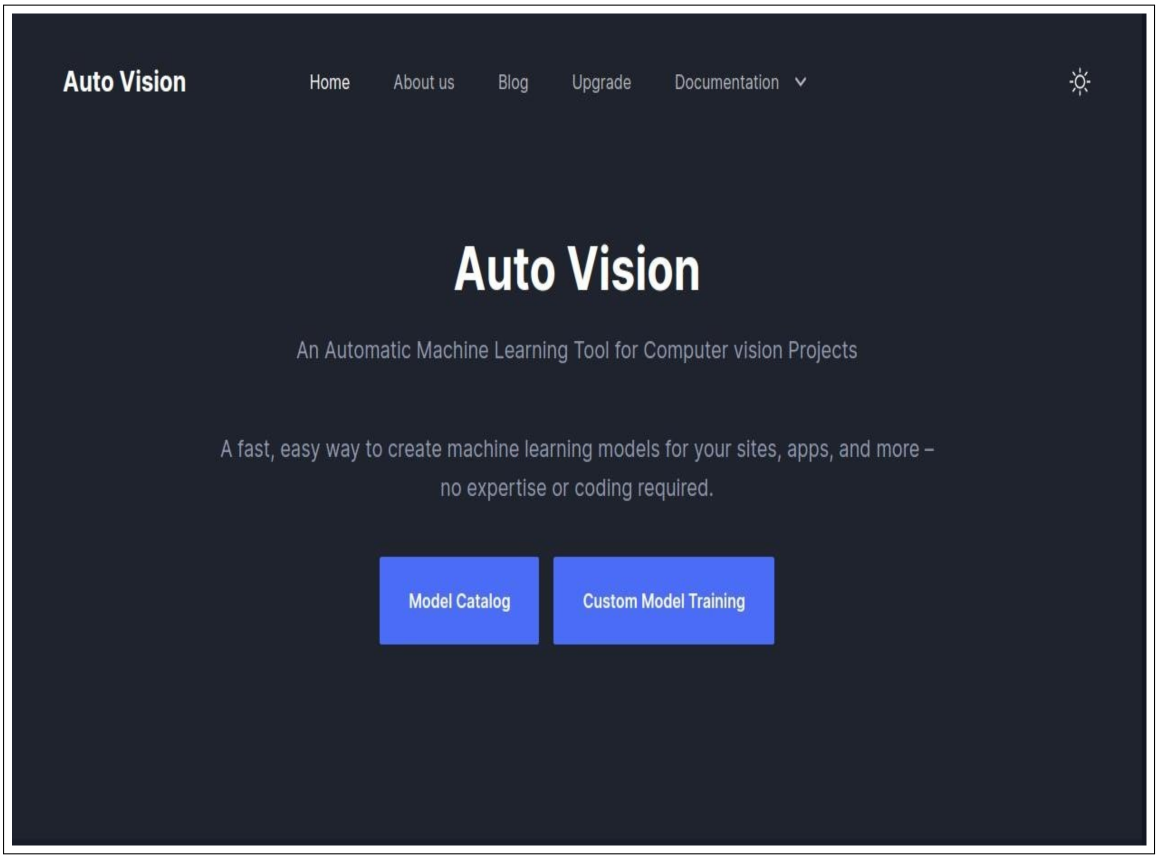

Figure 4.11: Landing Page

4.3.2 Login/ SingUp

Login/ SingUp UI that allows users to log in with GitHub or Google. Figure 4.12 Included popular social login options improve user comfort and security while making the authentication process faster.

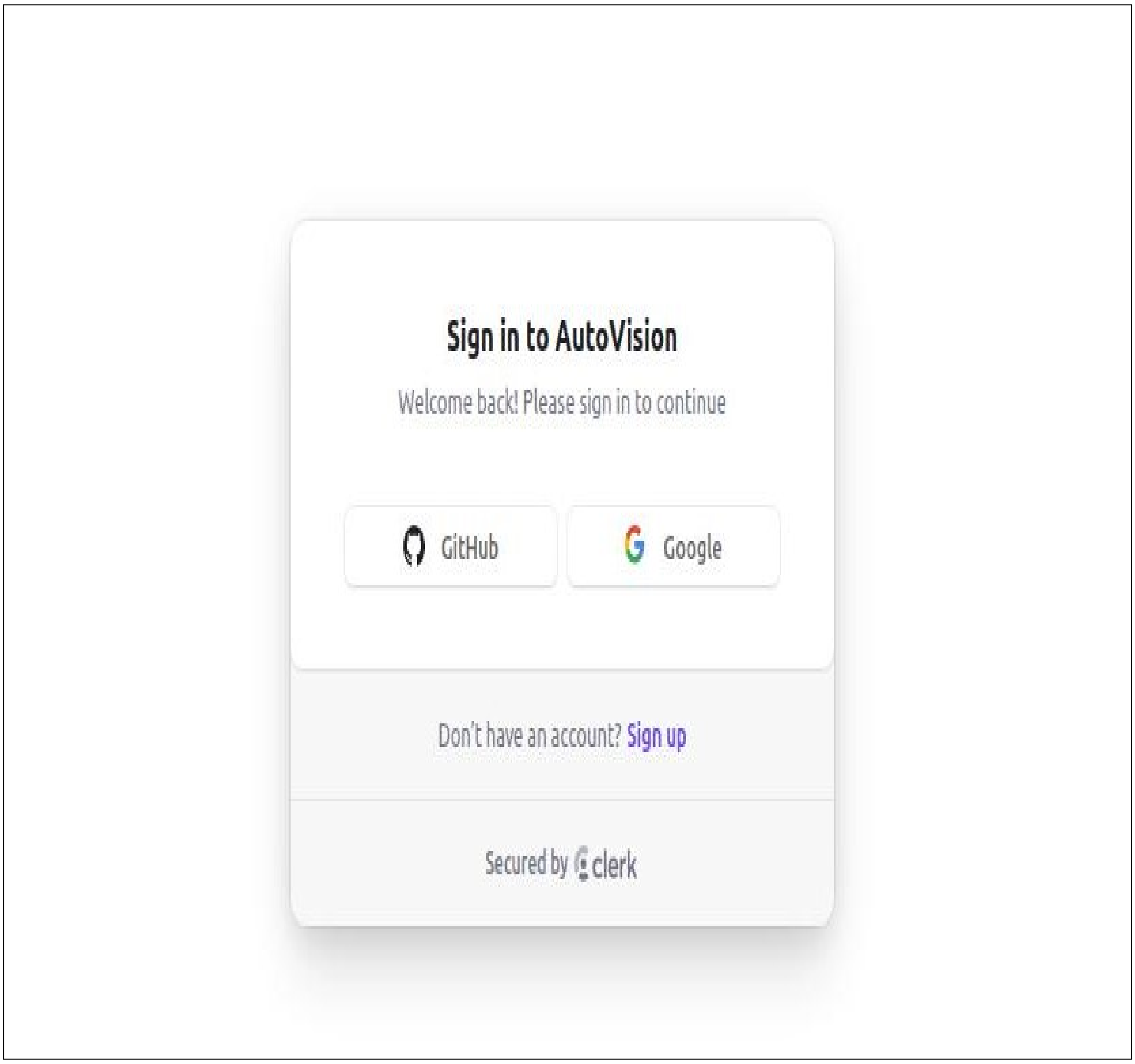

Figure 4.12: Login/ SingUp

4.3.3 Community

The platform supports a community feature as shown in figure 4.13 that enables users to interact and collaborate. Users can share experiences rate and compare as well as discuss the web application. With respect to all this, the community allows open communication and assist in making functionalities improved and user-friendly by considering every single requirement of different type users.

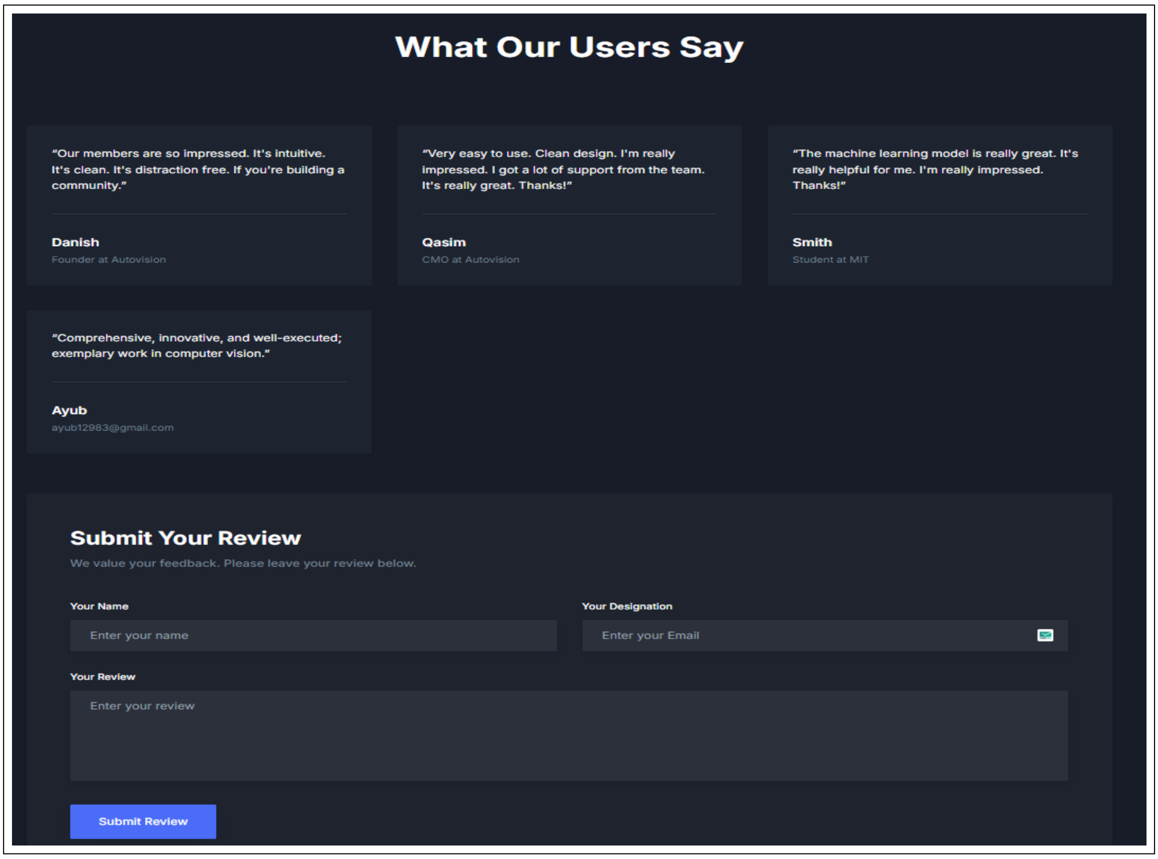

Figure 4.13: Community

4.3.4 About

Figure 4.14 For more information about AutoVision, its Mission, Vision and Team for the users. It makes an application more informative, transparent and builds trust by detailing the people and organizations behind its creation. By reading the Mission and Vision, users get to understand what AutoVision stands for as an organization in terms of values it believes in and long term vision which can help build up trust with its end customers. You will also find here a section on the Team that profiles some of the best and brightest who are accelerating this project. The items also have specific, detailed descriptions it in each category for customers to get peruse. To create or select the model that fits their requirement. This reassures users of the standards of the app but also shows who helps and gives their experiences to create it customer stories that are genuine go a long way, showing users not just the product but confident use cases which is essentially what people will buy into. Plus, it speaks to a standard of ethics and responsible marketing that is only becoming more critical in the digital age.

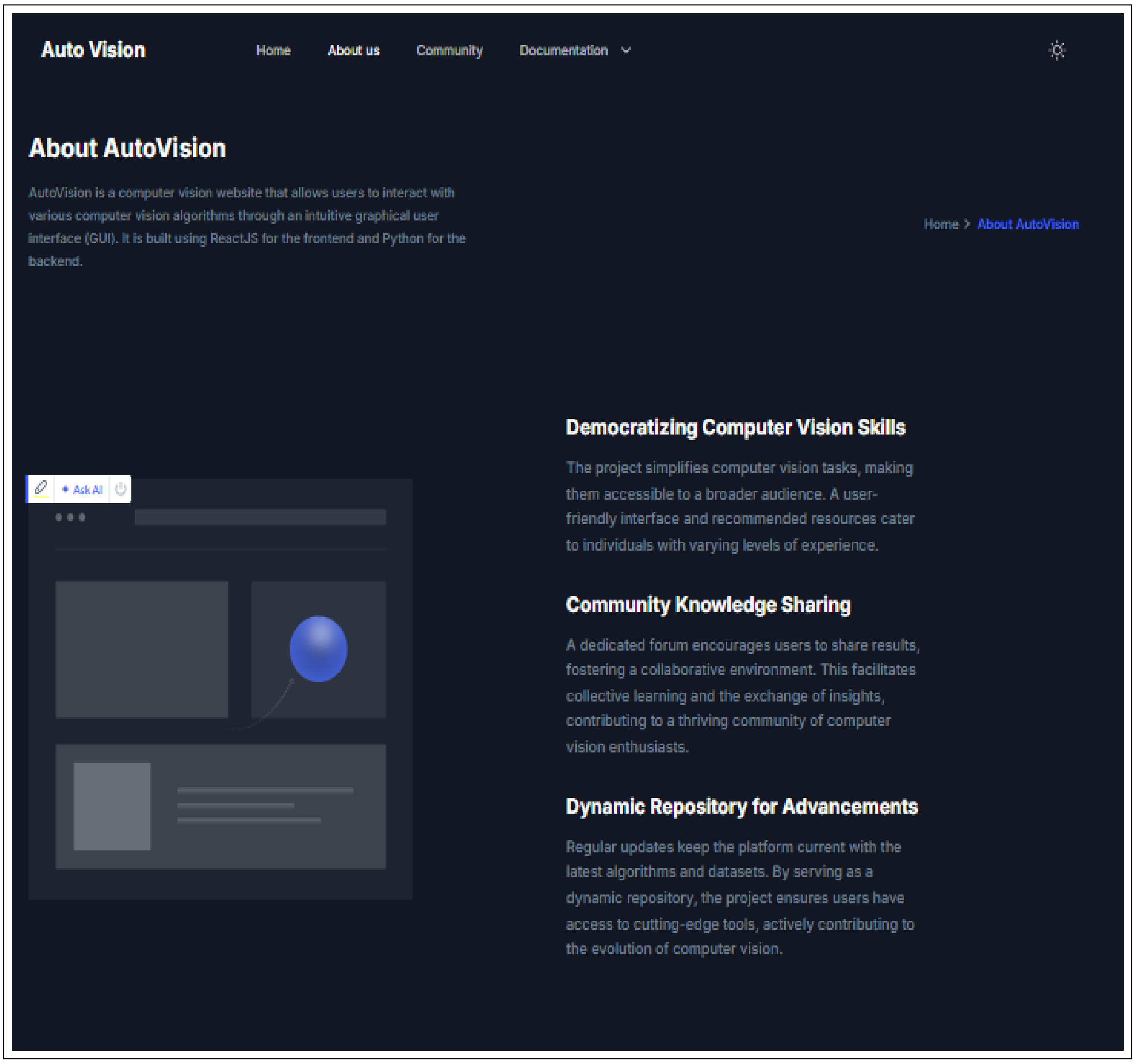

Figure 4.14: About Us

4.3.5 Model Catalog

Figure 4.15 shows a few pre-trained AutoVision models of specific categories detection, plant disease detection, brain tumor classification, bird species identification and bone fracture detection. The elaborate explanations and alternatives in each group allow users to investigate and choose models suitable for their specific requirements. AutoVision collaborated on a wide range of use cases with AutoVision to provide multiple pre-trained models, so we target users in various categories. There are detailed descriptions corresponding to each model, which clear the air about what a given model can and cannot do well, making it easier for users of these models to make informed decisions. Additionally, the alternatives within each category provide flexibility to allow users choose a model that fits their project goals and tech constraints.

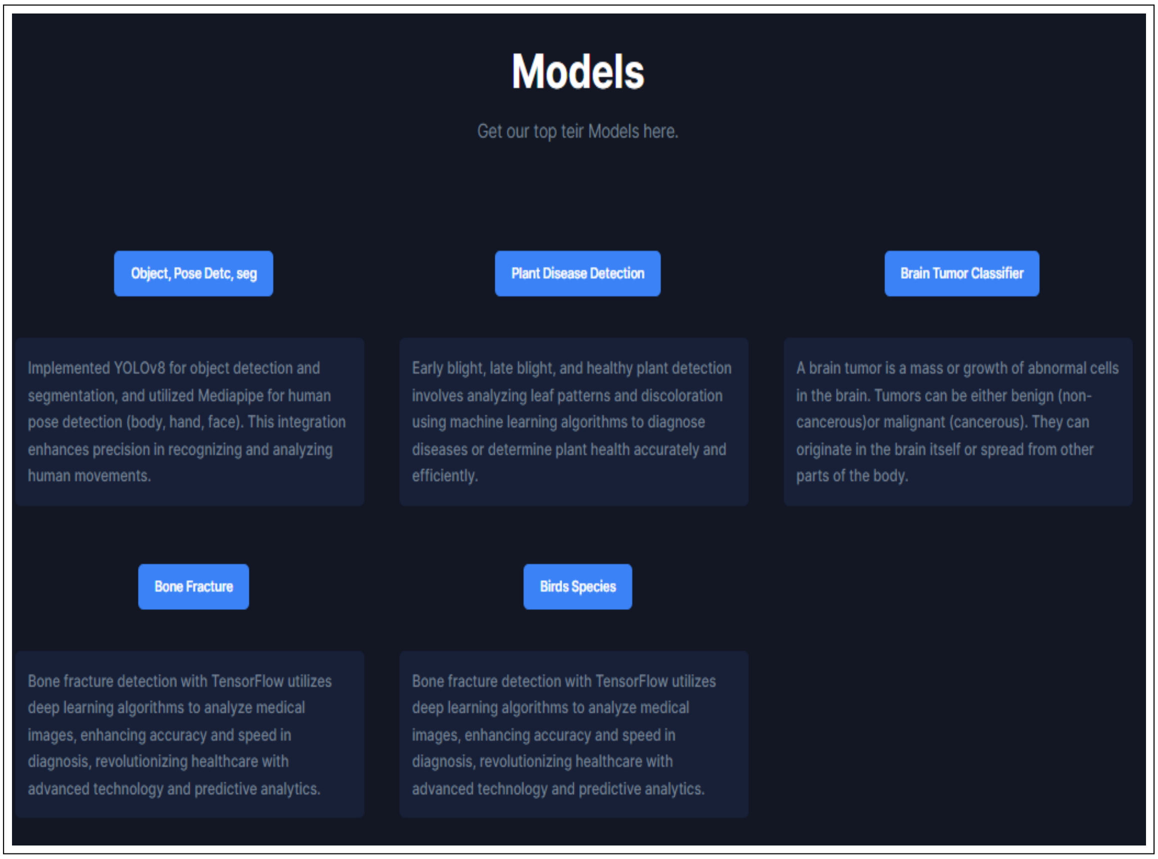

Figure 4.15: Model Catalog

a. Object Detection, Object Segmentation, Pose Estimation

Specialized models are available in the Model Catalog area for applications like Pose Estimation, Object Segmentation, and Object Detection shown in figure 4.18. This interface allows specialty units This, used in conjunction with computer vision applications makes sure the users can select between models that are closest to their needs which have been defined above, and get a more accurate as well efficient visual data processing done. The elaborate explanations and alternatives in each group allow users to investigate and choose models suitable for their specific requirements In addition to this, the models that exist inside of these catalogs are updated as well and made better over time so users always get access to cutting edge computer vision capabilities. This dedication to innovation and UX design solidifies AutoVision as the leading platform in visual data analysis. Furthermore, the transfer of these model solutions with other supporting tools and features within AutoVision ensures smooth functioning thereby eliminating third party infrastructure dependency and multiplying productivity. Thus users are able to deliver high-quality results in a fraction of the time, rendering Model Catalog invaluable for professionals within various industries.

Figure 4.16: Object Detection and Segmentation

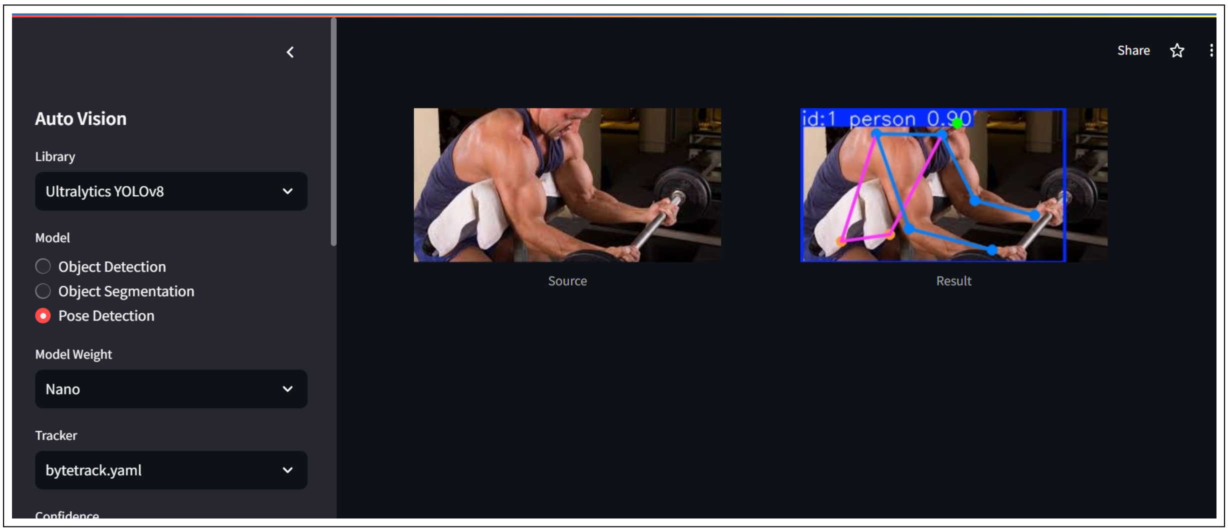

Figure 4.17: Pose Estimation

Figure 4.18: Object Detection, Object Segmentation, Pose Estimation

b. Brain Tumor

Brain tumor model interface can be found and classified from medical imaging, as shown in figure 4.19. The concept of deep learning has just begun to gain attention in computational neuroscience and neurology due to its large potential for a precise diagnosis and analysis of brain pathologies with the use of state-of-the-art machine-learning methods.

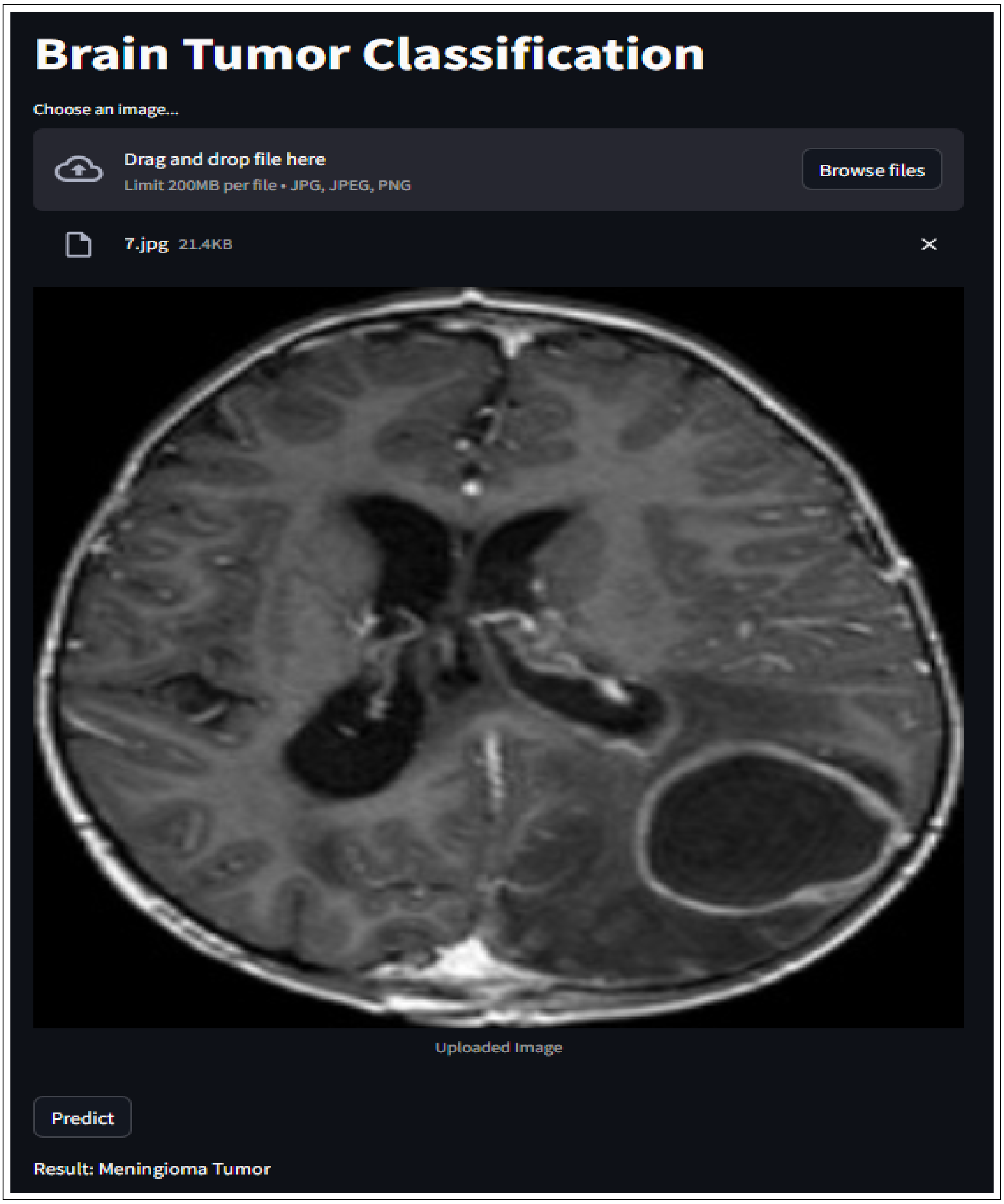

Figure 4.19: Brain Tumor

c. Plant Disease Detection

Figure 4.20 plant disease detection model interface of the framework can detect and identify plant diseases from photographs. Apart from its role in helping farmers and agricultural experts to monitor crop health, this technology also provides the early prediction and prevention of diseases for best farm productivity.

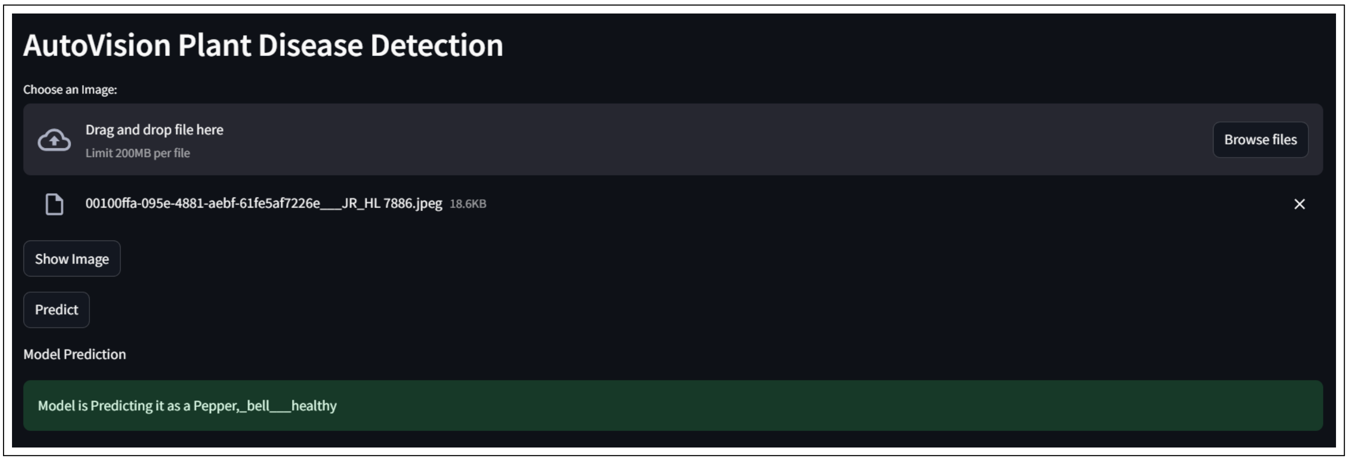

Figure 4.20: Plant Disease Detection

d. Birds Species

The Birds Species model interface offers ways to recognition and characterize birds species, as shown in figure 4.21 from pictures. To aid the research community, ornithologists, and wildlife enthusiasts in their studies and conservation efforts on avian biodiversity, it employs machine learning techniques to consistently detect and identify species.

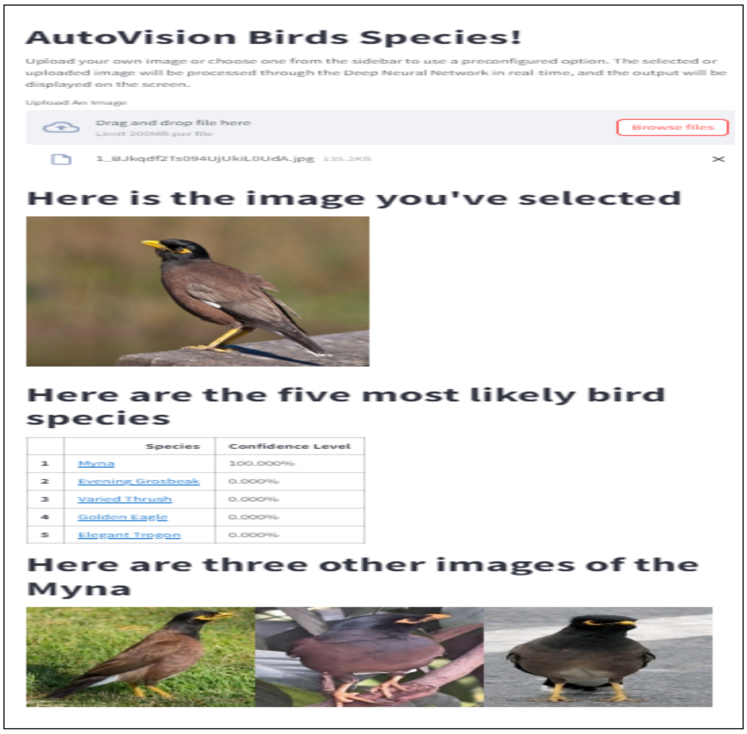

Figure 4.21: Birds Species

e. Bone Fracture

The Bone fracture model interface with the ability to identify and classify bone fractures as shown in figure 4.22 in medical images. This in return, simplifies accurate diagnosis of bone injury by physicians which is crucial for efficient patient care and treatment planning.

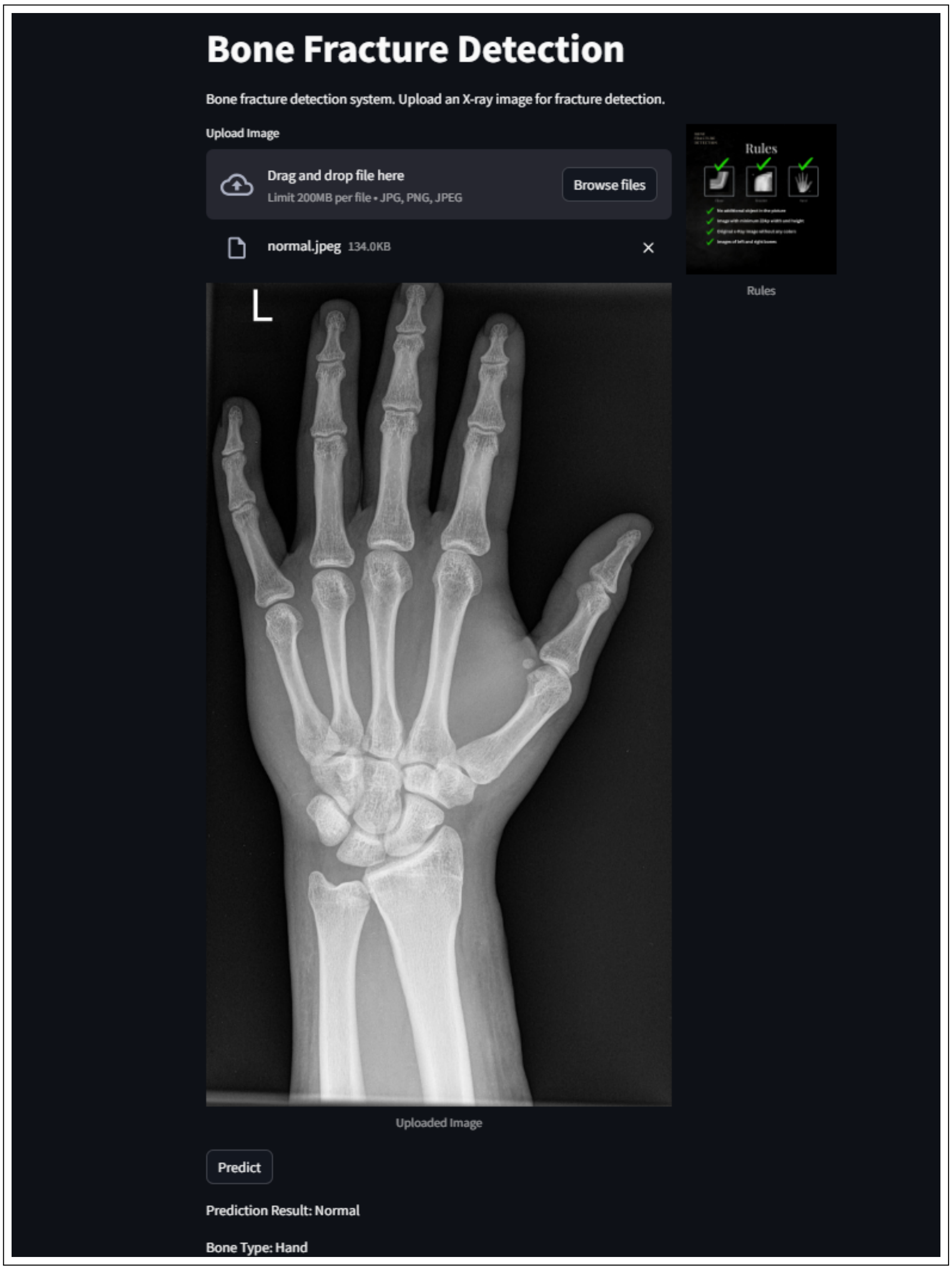

Figure 4.22: Bone Fracture

4.3.6 Custom Model Training

With custom model training interface, users can create task and dataset specific SOTA machine learning models 4.23. It provides various pre-processing tools, methods for training and analyzing models and has huge customizing options such as image classification or regression into your own model.

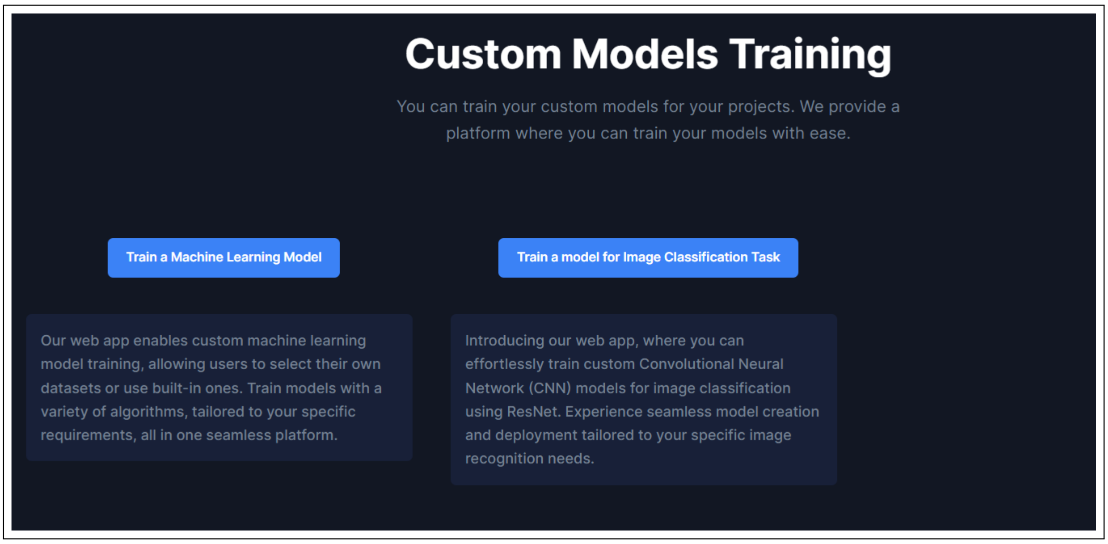

Figure 4.23: Custom Model Training

a. Train ML model

AutoVision allows users to upload multiple datasets and techniques for custom model training as shown in figure 4.24. It steps users through the process of choosing algorithms, setting training parameters and monitoring model performance to yield optimized models.

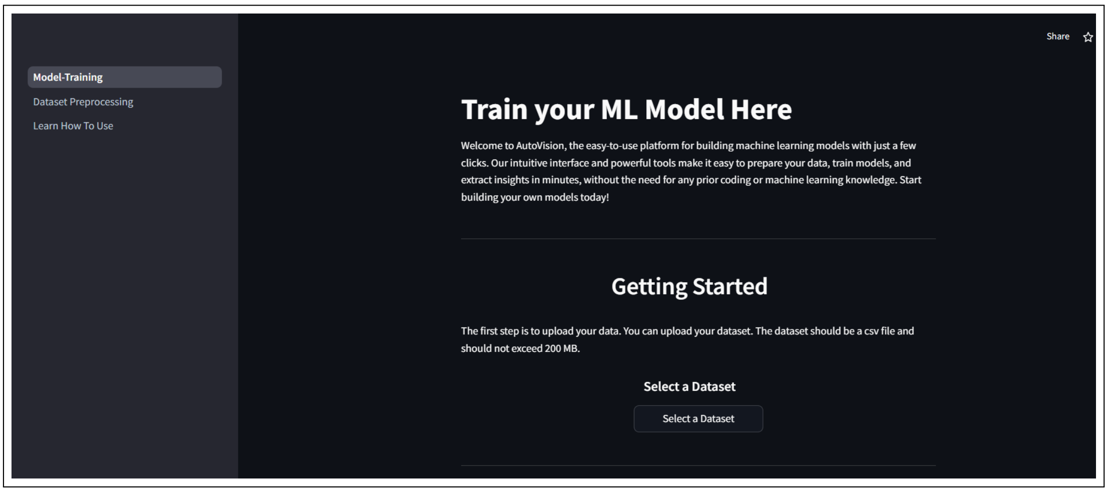

Figure 4.24: Train ML model

b. Dataset Pre processing

Dataset pre-processing is the process of preparing raw data as shwon in figure 4.25, for machine learning model training by procedures like, Feature extraction Augmentation Normalization Here Keras library is used to prepare a dataset for image recognition. By ensuring that the datasets are well-formatted for the model training, it enhances efficiency and trust in machine learning predictions.

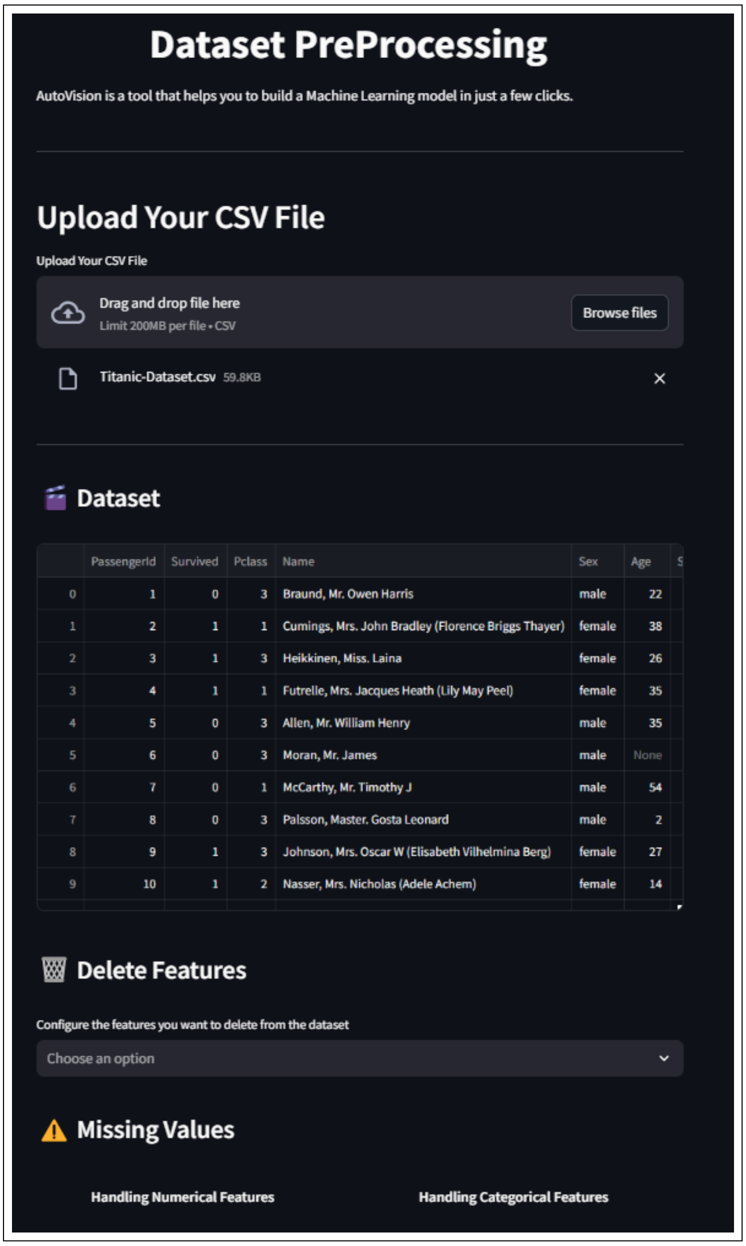

Figure 4.25: Dataset Pre processing

c. Train Image Classification Model

Train machine learning models for image classification as shown in figure 4.26. Adding labeled picture datasets towards the training pipeline, tweaking model architecture and home improvement parameters, testing in order for appropriate classification consequences.

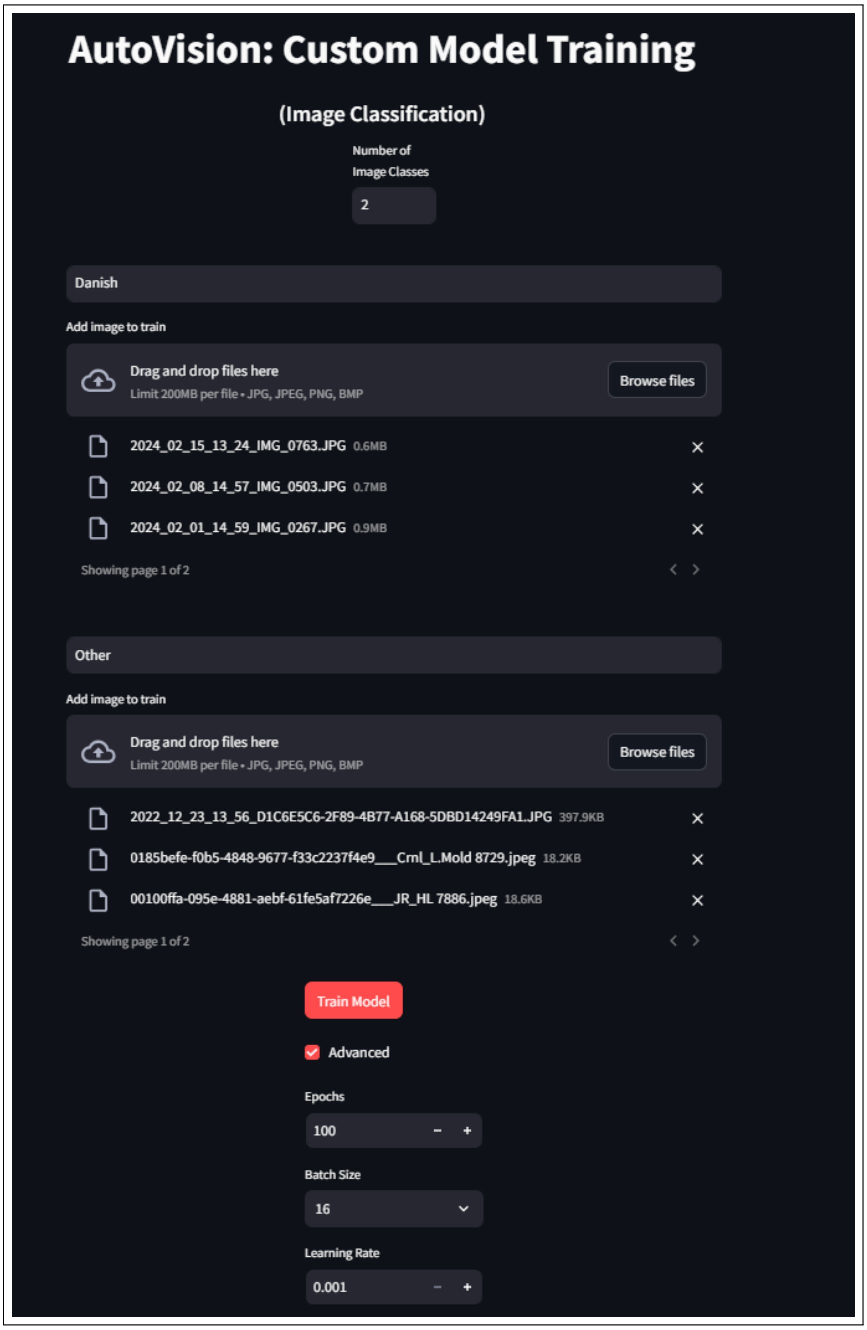

Figure 4.26: Train Image Classification Model

Chapter 5

Software Testing

This chapter describes the software development testing strategies for AutoVision, a PaaS-based computer vision utility that ensures dependability, usability, and efficiency. Unit Testing - This is the process of checking if a particular system unit or component works correctly and meets its dependability requirements. This part consists of the full table with unit tests for post-processing checks and errors dealing with take a look at, each anticipated vs precise outcomes.
Integration Testing: Integration testing is done after the unit test and it checks how individual modules within a system interact together or communicate. The third category is Integration level tests which record interactions like frontend-backend calls, ml models interaction, etc, and most importantly validate whether the components are functioning as they intended to be integrated.
This chapter also touches on Black Box Testing - a way of describing the technique in which the tester knows nothing about how things work under the hood. It focuses on testing the system environment as a whole functioning with expected behavior by users without any need for technical implementation. For example, test cases like "Upload Image File" and "Download Results" represent an in-depth Black box testing approach in comparison to other tests through which the functionalities at different layers are validated keeping User Perspective as a priority.

Test Cases Generation

- Test Case ID
- Test Case Description
- Test Data
- Expected output
- Actual output
- Test Result

5.1 Unit Testing

Unit-Level Testing; The unit-level tests were done for testing every module of the Auto-Vision system to ensure that each software part works as expected respectively as shown in Table 5.1. Each test case is assigned a certain capability - eg, verify image upload; choose from some ready-to-use models, or predict based on user input. Test Data: it is the input with which we have tested our system, Expected Result What ideally should a test do to result? Expected output is the predicted outcome, it should exactly match with actual outputs (outcomes of testing) to mark a test case pass. The test result column compares the expected and actual output to determine if a test is passed or failed.

Table 5.1: Unit Level Testing

S.No	**Test Case Description**	**Test Data**	**Expected Output**	**Actual Output**	**Test Result**
U1	Verify parsing of valid input data for model training.	Valid input data	Parsed data object	Parsed data object	Pass
U2	Generate a classification model based on user input.	Selected algorithm, Dataset	Trained model	Trained model	Pass
U3	Test error handling for invalid input data.	Invalid data input	Error message	Error message	Pass
U4	Measure performance with a large dataset.	Large dataset	Model training completion within 1 hour	Model training completion within 1 hour	Pass
U5	Verify boundary conditions with empty or null input.	Empty dataset	Error message	Error message	Pass
U6	Test parsing of invalid input data for model training.	Invalid data input	Error message	Error message	Pass
U7	Attempt model training with missing data fields.	Dataset with missing fields	Prompt to complete missing fields	Prompt to complete missing fields	Pass

5.2 Integration Testing

To evaluate the cooperation and interaction between various AutoVision system modules or components, integration-level tests are documented in Table 5.2. An example of an integration scenario is provided in each test case description. Examples include data processing, integrating with other libraries for image handling, and communicating between the front end and the ML/DL models. The input data provided for integration is known as the test data, and the expected output denotes the outcome that the integration process is meant to produce. While the test result column indicates whether or not the integration was successful, the actual output mirrors the observed result during testing. With the help of these tests, you can be sure that all the system's parts function together as planned.

Table 5.2: Integration Level Testing

S.No	**Test Case Description**	**Test Data**	**Expected Output**	**Actual Output**	**Test Result**
I-001	Front end-Model Communication for Image Processing	Uploaded image	Front end sends image to the model, receives processed data	Front end successfully receives processed data from the model	Pass
I-002	Generate a classification model based on user input.	Selected algorithm, Dataset	Trained model	Trained model	Pass
I-003	Test error handling for invalid input data.	Invalid data input	Error message	Error message	Pass
I-004	Measure performance with a large dataset.	Large dataset	Model training completion within 1 hour	Model training completion within 1 hour	Pass
I-005	Verify boundary conditions with empty or null input.	Empty dataset	Error message	Error message	Pass
I-006	Test parsing of invalid input data for model training.	Invalid data input	Error message	Error message	Pass
I-007	Attempt model training with missing data fields.	Dataset with missing fields	Prompt to complete missing fields	Prompt to complete missing fields	Pass

5.3 System Testing

Table 5.3 describes the system-level test results for AutoVision in terms of performance, functionality, and reliability. For each input-output, the test cases verify the major system features of the other end. Results are used to verify whether the system meets expectations, such as working in practice for users.

Table 5.3: System Level Testing

S.No	Test Case Description	Test Data	Expected Output	Actual Output	Test Result
S-001	End-to-End Image Processing Workflow	Uploaded image	System processes image and displays results accurately	Image processed and results displayed correctly	Pass
S-002	Model Training and Inference	Training dataset	Model trains successfully and makes accurate predictions	Model trained successfully and made accurate predictions	Pass
S-003	System Availability Assessment	Continuous operation scenario	System remains available and responsive over extended period	System remained available and responsive	Pass
S-004	User Interface Responsiveness	Various user interactions	UI responds promptly to user inputs	UI responded promptly to user inputs	Pass
S-005	Error Handling Mechanisms	Invalid data inputs	System gracefully handles errors and provides meaningful feedback	System handled errors gracefully and provided feedback	Pass
S-006	Integration with External Libraries	Image processing with OpenCV	System successfully integrates and uses external libraries	External libraries integrated and used successfully	Pass
S-007	Scalability Under Load	High volume of user requests	System scales effectively and maintains performance	System scaled effectively and maintained performance	Pass

5.4 Black Box Testing

Black box testing is a software testing technique where the internal workings of the system are not known to the tester. Instead, the tester focuses on the system's external behavior and functionality [15]. Black box testing, in the context of the AutoVision web application test cases supplied, guarantees that the application functions as intended from the user's point of view, independent of its internal operations. In the "Upload Image Functionality Test Case," for instance, the tester confirms that the system manages uploaded photos successfully without requiring knowledge of its internal processing techniques. Comparably, the tester verifies that several chart kinds may be correctly generated using user-provided data in the "Chart Generation Test Case." The "Download Image Test Case" verifies that the system successfully downloads the chart as an image file, and the "Get Code Test Case" verifies that the system displays a code snippet for embedding the generated chart. As part of AutoVision's user-oriented approach, black box testing is there to ensure that the system works and behaves accordingly without the need for knowledge about its internal architecture or programming.

5.4.1 Upload Image Functionality

Table 5.4 is the to make sure that the system supports different image file formats, also test the capability to upload photos for AutoVision project. This is a test of correctly and without any pixel errors uploading pictures in the system, on this aspect. The system needs to be able to work with multiple image file types such as JPEG or PNG and the images need accurate processing. The upload of photographs to the system for further processing and analysis will be available, only after passing said test.

Table 5.4: Upload Image Functionality Test

S.No	Test Case Description	Test Data	Expected Output	Actual Output	Test Result
1	Attempt to upload an image file in different formats	Various image files in different formats	System successfully uploads and processes the image	Image is uploaded and processed without errors	Pass

5.4.2 Model Inference Functionality

The table 5.5 shows that its purpose is to test the inference power of the model. It tests a system on how well it can predict based on input images given to the model. This test case is where the system is provided with an image that contains certain objects to be identified as well as classified. This means the model must be good at determining and also analyzing what kind of data is in our input image if it would like to make accurate predictions. With the test passing, it assures us that this feature understands model inference for MLOps users favor confidence in system prediction.

Table 5.5: Model Inference Functionality Test

S.No	Test Case Description	Test Data	Expected Output	Actual Output	Test Result
1	Input an image into the system for model inference	A sample image containing recognizable objects	Model makes accurate predictions about the objects in the image	Model provides accurate predictions based on the input image	Pass

5.4.3 Custom Model Training

It tests that clients can appropriately train their machine learning models by assessing the custom model training ability of AutoVision project as shown in table 5.6. On this test, you evaluate how well your model does in practice (on validation data), depending on the performances of its ancestor transformers and an adjusted and labeled dataset sitting in a holds-out way things are done training set. We want enough accuracy to know that the structure of our trained model is working (so it can go on and learn from data) but not too much so we don't overfit. It confirms that technology allows customers to train a custom model successfully at the desired level of accuracy and so implement their machine-learning solutions.

Table 5.6: Custom Model Training Test

S.No	Test Case Description	Test Data	Expected Output	Actual Output	Test Result
1	Train a custom machine learning model using provided dataset and algorithms	A labeled dataset for training	Trained model achieves satisfactory accuracy on validation data	Trained model achieves desired accuracy on validation data	Pass

5.4.4 System Responsiveness

Measuring the system responsiveness of the System as shown in table 5.7. To complete this test case you will need to interact with the system using buttons, type text, etc (few UI elements). The amount of time it takes for the system to respond has also been noted. The user interface (UI) should be quick to respond with no significant disruption. There are a lot of benefits when we follow the practices above in addition to enabling better isolation between concerns, it also guarantees that user interface behavior can be verified quickly - without any architectural layer slowing us down.

Table 5.7: System Responsiveness Test

S.No	Test Case Description	Test Data	Expected Output	Actual Output	Test Result
1	Interact with the system UI and assess responsiveness	Various user interactions (e.g., clicking buttons, entering text)	System UI responds promptly to user inputs	System UI responds without noticeable delays	Pass

5.4.5 Error Handling

The test shown in table 5.8 is a system-wide error-handling capability testing, the purpose here is to verify if faults are detected and managed correctly. You need to test the system by providing bad input data or wrong file format and evaluate its reaction. The system is supposed to detect the errors and deliver meaningful error messages letting users know how they can fix them. The system is very good at Error Management, guiding the user through mistakes to aid them in their debugging or resolution process keeping the experience predictable and learnable.

Table 5.8: Error Handling Test

S.No	Test Case Description	Test Data	Expected Output	Actual Output	Test Result
1	Provide invalid input data and assess system's error handling	Incorrectly formatted data or unsupported file formats	System gracefully handles errors and provides informative error messages	System detects and handles errors appropriately, guiding the user to rectify the issue	Pass

5.4.6 External Library Integration:

Table 5.9 shows that the integration process was successfully tested to be seamless within AS for other tools used by image processing tasks of AutoVision project. This test case utilizes external libraries for different image processing operations like Scaling images and cropping them to verify how well they fit within the system. It was assumed the external libraries would function properly, executing what they must in a seemingly perfect relationship with our system. The test comes back successfully concluding that the system is capable of achieving functionality and flexibility by helping external resources.

Table 5.9: Integration with External Libraries Test

S.No	Test Case Description	Test Data	Expected Output	Actual Output	Test Result
1	Utilize external libraries for image processing tasks	Images requiring various processing tasks (e.g., resizing, cropping)	External libraries are seamlessly integrated and produce desired image processing results	External libraries perform required tasks accurately within the system	Pass

Conclusion

AutoVision represents an important step forward in democratizing complex machine learning algorithms for a wider audience. AutoVision facilitates the utilization of AI at large scale across industries by providing native support to pre-trained models for use-cases like object segmentation, plant disease detection, brain tumor classification etc. The system is possibly infinitely versatile because it could come close to meeting individual needs perfectly contingent on employing user-supplied datasets and willingness to train custom models. Driven by a user-oriented approach, the design is minimalistic and backed up with modern web technologies; it provides an uncomplicated experience for users to get any data they require within seconds rendering seamless sign-ins using GitHub or Google. It is an easy to use, well documented (like really comprehensive documentation) library so both beginner and experts can work with. AutoVision is highly scalable, extremely flexible and provides a solid backbone for real life machine learning applications that fuel innovation to make data-driven decisions.

References

[1] M. Ergen *et al.*, "What is artificial intelligence? technical considerations and future perception," *Anatolian J. Cardiol*, vol. 22, no. 2, pp. 5–7, 2019.

[2] I. El Naqa and M. J. Murphy, *What is machine learning?* Springer, 2015.

[3] G. Sinha and V. Bajaj, "18 - data deduplication applications in cognitive science and computer vision research," in *Data Deduplication Approaches* (T. T. Thwel and G. Sinha, eds.), pp. 357–368, Academic Press, 2021.

[4] N. Pinnis, "Modern website development with strapi and next. js," 2022.

[5] G. Van Rossum and F. L. Drake, *An introduction to Python.* Network Theory Ltd. Bristol, 2003.

[6] I. Streamlit, "Streamlit: A faster way to build and share data apps." `https://streamlit.io/`, 2024. Accessed: 2024-07-03.

[7] I. Vercel, "Vercel: Build and deploy the best web experiences with the frontend cloud." `https://vercel.com/`, 2024. Accessed: 2024-07-03.

[8] F. K. Chan and J. Y. Thong, "Acceptance of agile methodologies: A critical review and conceptual framework," *Decision support systems*, vol. 46, no. 4, pp. 803–814, 2009.

[9] Visual Paradigm, "Use case diagram," 2022. Accessed: 2022-08-18.

[10] Visual Paradigm, "What is a data flow diagram," 2022. Accessed: 2022-08-18.

[11] Visual Paradigm, "Activity diagram: A graphical representation of workflows of step-wise activities and actions with support for choice, iteration, and concurrency," 2023. Accessed: 2023-10-10.

[12] Visual Paradigm, "Sequence diagram: A type of interaction diagram that shows how processes operate with one another and in what order," 2023. Accessed: 2023-10-10.

[13] Clerk.com, "Clerk dashboard," 2023. Accessed: 2023-10-10.

[14] Lucidchart, "Data flow diagram," 2023. Accessed: 2023-10-10.

[15] S. Nidhra and J. Dondeti, “Black box and white box testing techniques-a literature review,” *International Journal of Embedded Systems and Applications (IJESA)*, vol. 2, no. 2, pp. 29–50, 2012.