BREAST CANCER DETECTION

A PROJECT REPORT
for
Project (KCA451)
Session (2024-25)

Submitted by

DRISHTY
(University Roll No : 2300290140061)
EKATA GUPTA
(University Roll No : 2300290140062)

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Mr. Rabi N. Panda
Associate Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(MAY 2025)

DECLARATION

We hereby declare that the work presented in this report entitled **“BREAST CANCER DETECTION"**, was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

We have given due credit to the original authors/sources for all the words, ideas, diagrams, computer programs, experiments that are not my original contribution.

We affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, we shall be fully responsible and answerable.

DRISHTY (2300290140061)
EKATA GUPTA (2300290140062)

CERTIFICATE

Certified that Name of student **Drishty (2300290140061), Ekata Gupta (2300290140062)** have carried out the project work having **“Breast Cancer Detection”** (Project-KCA451) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Mr.Rabi N. Panda
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr.Akash Rajak
Dean
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Breast Cancer Detection

Drishty

Ekata Gupta

ABSTRACT

Breast cancer is one of the leading causes of mortality among women globally, and early detection plays a vital role in increasing the survival rate. This project focuses on the development of an AI/ML-powered system aimed at the early diagnosis of breast cancer using patient data and clinical parameters.

The proposed system leverages machine learning techniques such as Logistic Regression, Decision Tree, Random Forest, and Support Vector Machine (SVM) to analyse features derived from breast cancer datasets. These features include tumor radius, texture, perimeter, area, and smoothness among others. By training and validating these models, the system is able to classify tumors as benign or malignant with high accuracy.

An additional layer of innovation is introduced through a user-friendly web interface where users can input symptom-related data via a dynamic Symptom Checker module. This module collects information about lifestyle, personal history, and physical symptoms, and computes a risk score to provide an early indication of potential malignancy.

The project not only aids in decision support for healthcare providers but also empowers individuals with a tool for proactive health monitoring. By combining medical expertise with technological advancements, the system strives to reduce diagnostic delays and improve outcomes through accessible, scalable, and intelligent screening solutions.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. Our deepest gratitude goes to project supervisor, **Mr. Rabi N. Panda (Associate Professor),** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Akash Rajak, Dean, Department of Computer Applications,** for his insightful comments and administrative help on various occasions.

Fortunately, we have many understanding friends, who have helped me a lot on many critical conditions.

Finally, our sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without them support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

DRISHTY (2300290140061)
EKATA GUPTA (2300290140062)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PROJECT DESCRIPTION

The **Breast Cancer Detection System** is an AI/ML-powered platform designed to help users detect the potential presence of breast cancer by offering three distinct methods of diagnosis: **Manual Entry**, **CSV Upload**, and **Symptom Checker**. This project uses trained machine learning models to classify cancer types as **benign** or **malignant** based on the user input. It aims to simplify early detection and increase accessibility for individuals and medical professionals alike.

1. **Manual Entry:** This method allows users to input key diagnostic parameters manually (e.g., radius mean, texture mean, perimeter mean, etc.). These values are then fed into the ML model to generate a prediction.
2. **CSV Upload:** In this mode, users can upload a medical report or dataset in CSV format. The system parses the uploaded file and uses the included data to make a prediction.
3. **Symptom Checker:** A more interactive, user-friendly feature where users answer a short series of questions regarding symptoms and lifestyle. The system analyses the responses and calculates a risk score, categorizing the risk level as low, moderate, or high.

   This multifaceted approach caters to users with different levels of technical expertise and access to medical data. It combines user-centric design, robust backend logic, and advanced predictive modelling to offer a comprehensive early detection tool for breast cancer.

1.2 PROJECT SCOPE

1. To provide an accessible, user-friendly, and reliable system for breast cancer prediction.
2. To integrate multiple methods of diagnosis to cater to diverse user needs.
3. To support healthcare practitioners by offering a quick, data-driven second opinion.
4. To encourage early detection and increase awareness of breast cancer symptoms.
5. To use AI/ML technology effectively for real-world healthcare applications.

1.3 FUTURE SCOPE

1. Integration of deep learning models for improved accuracy and real-time predictions.
2. Expansion of the symptom checker with dynamic questioning and NLP-based response analysis.
3. Mobile application version for better accessibility in rural and underserved areas.
4. Incorporation of patient history records for more personalized predictions.
5. Connecting with healthcare providers for real-time diagnosis and teleconsultation.

1.4 IDENTIFICATION OF NEED

With the rising number of breast cancer cases globally, early detection is critical. Unfortunately, access to diagnostic tools and expert consultations is not always available, especially in remote or underserved areas. Many individuals also struggle to interpret their own health data or symptoms.

Key Needs Identified:

1 A centralized, AI-based diagnostic system that users can operate independently.

2 Support for multiple diagnostic inputs (manual, file upload, symptom-based).

3 A system that educates while diagnosing — enhancing user awareness.

4 Reliable and interpretable results that can be shared with healthcare professionals.

5 Accessible design that caters to both tech-savvy users and beginners.

1.3 PROBLEM STATEMENT

Breast cancer, if detected early, is highly treatable. However, delays in diagnosis due to lack of awareness, high costs, or limited access to healthcare facilities lead to late-stage detection and reduced survival rates. The current gap in accessible and automated diagnostic support needs to be filled with intelligent, user-friendly solutions.

This project aims to bridge this gap by providing a platform that uses machine learning to predict breast cancer likelihood based on multiple user inputs. The system simplifies the diagnostic process, making it available to users without requiring direct access to healthcare institutions.

1.4 SOFTWARE/TECHNOLOGY USED IN PROJECT

A. Python & Machine Learning Libraries

Python serves as the core language for the project due to its simplicity and vast ecosystem of libraries suitable for data science and machine learning.

- **scikit-learn**: Used for model development, including data splitting, scaling, and training algorithms like Random Forest and Support Vector Machines (SVM).
- **pandas**: Employed for handling and processing structured data, especially during CSV upload and analysis.
- **joblib**: Utilized to save and load the trained machine learning model efficiently.
- **NumPy**: Supports numerical operations involved in calculations and model inputs.

B. Flask

Flask is a lightweight and flexible web framework for Python that powers the backend of the application.

- It handles HTTP requests and responses, routes different modules (Manual Input, CSV Upload, Symptom Checker), and communicates with the ML model to fetch predictions.

- Flask also serves the web pages and processes the forms submitted by users.

C. HTML , CSS , JavaScript

These core web technologies form the structure, style, and interactivity of the frontend:

- HTML: Provides the structure of the web pages used for input forms, result displays, and navigation.
- CSS: Styles the interface to make it visually appealing and consistent across devices.
- JavaScript: Adds interactivity, like dynamic form handling and instant validations.

D. Bootstrap

Bootstrap is a popular frontend toolkit used to design responsive, mobile-first web pages.

- Enhances user experience with a grid system, buttons, cards, modals, and navigation bars.
- Helps maintain design consistency across the site, improving usability and visual appeal.

E. CSV & Pandas for Upload Functionality

The Upload CSV module leverages Python's pandas library to handle and validate structured data from user-uploaded .csv files.

- Reads the CSV file and extracts required diagnostic features.
- Ensures column formats match the model’s expectations before feeding the data into the ML model for prediction.

F. Jupyter Notebook

Jupyter Notebook is used during the development and training phase of the machine learning model.

- Enables exploratory data analysis (EDA), visualization (using matplotlib and seaborn), and iterative model tuning.
- Also serves as a documentation tool for experiments and comparisons of different algorithms.

G. Visual Studio Code (VS Code)

The entire project is developed using VS Code, a popular integrated development environment (IDE).

- Offers extensions for Flask, Python, Git, and live server preview.
- Makes debugging, code formatting, and testing more efficient.

H. GitHub

GitHub is used for version control and code hosting.

- Allows sharing of the project source code with others for collaboration and contribution.
- Ensures version tracking and backup of all the code and documentation.

1.4.1 NON- FUNCTIONAL REQUIREMENTS

The non-functional requirements define the quality attributes and overall performance expectations for the Breast Cancer Detection System. These ensure that the application is not only functional but also efficient, scalable, and user-friendly.

1. **Performance**
   The system should provide fast and accurate predictions, minimizing the delay between input submission and result display. The use of lightweight frameworks like Flask ensures minimal overhead and fast response times.
2. **Scalability**
   The architecture must support scalability to handle an increasing number of users, datasets, and prediction requests without compromising performance.
3. **Usability**
   The interface should be intuitive and user-centric, allowing users with minimal technical expertise to operate the system with ease.
4. **Security**
   Input data from users should be handled securely, with validation mechanisms to prevent malicious data entry. If hosted online, the system should include encryption (HTTPS), authentication (if needed), and secure storage.
5. **Reliability**
   The system must be consistently available and operate without crashes or significant downtime. It should handle errors gracefully and recover from unexpected failures.
6. **Maintainability**
   The codebase should be modular and well-documented to simplify future enhancements, debugging, or the integration of new modules (e.g., mammogram image analysis).
7. **Portability**
   The system should work across different platforms including Windows, macOS, and Linux. It should also function smoothly on mobile devices when accessed via a web browser.
8. **Interoperability**
   The system should allow integration with third-party data platforms (like hospital databases or government screening portals) in future expansions.

9. **Accessibility**
   The interface should comply with basic accessibility standards (such as WCAG), supporting screen readers and clear navigation for visually impaired users.
10. **Data Integrity**
    Data submitted by users must be accurately processed and predictions must reflect the input without corruption or unintended alteration.

1.4.2 FUNCTIONAL REQUIREMENTS

The functional requirements define the specific features and capabilities of the Breast Cancer Detection System. These functionalities support both the technical workflow and the end-user experience.

1. **Manual Data Entry & Prediction**

   Users can manually enter relevant medical data (e.g., mean radius, texture, smoothness) into a form. Upon submission, the data is processed and a prediction is returned (Malignant or Benign).

2. **CSV Upload and Batch Prediction**

   Users can upload .csv files containing multiple rows of patient data. The system will validate the file format, process all entries, and return predictions in a structured format.

3. **Symptom Checker with Risk Scoring**

   An interactive questionnaire collects information about symptoms, lifestyle, family history, and other health indicators. Based on this, a risk score is computed and displayed.

4. **Navigation Bar with Modular Access**

   Users can easily navigate through sections like **Home**, **Predict**, **About**, **Support**, and **Login**. The **Predict** section expands to show diagnosis options via visually designed cards.

5. **Project Information and Support Page**

   The **About** and **Support** pages contain information about breast cancer awareness, system usage, contact details, and FAQs.

6. **Responsive Design**

   The system’s interface adapts to various screen sizes, providing a seamless experience on desktops, tablets, and smartphones.

7. **Model Integration**

A pre-trained machine learning model is loaded in the backend, and it performs real-time predictions based on user input.

8. **Error Handling and Alerts**

   The system provides real-time feedback for invalid inputs, missing fields, or unsupported file formats.

9. **Result Display and Summary**

   After prediction, users are shown results in a clear format, possibly with recommendations or links to further reading.

10. **Future Expandability Hooks**

    The architecture supports the addition of new modules like AI-driven report analysis, image-based diagnostics, or integration with medical APIs.

CHAPTER 2

FEASIBILITY STUDY

2.1 INTRODUCTION

Feasibility of the system in an important aspect, which is to be considered. The system needs to satisfy the law of economic, which states that the maximum output should be yielded in minimum available resources.

A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions.

There are five types of feasibility study separate areas that a feasibility study examines, described below:

1. **Technical Feasibility**

This assessment focuses on the technical resources available to the organization. It helps organizations determine whether the technical resources meet capacity and whether the technical team is capable of converting the ideas into working systems. Technical feasibility also involves the evaluation of the hardware, software, and other technical requirements of the proposed system. As an exaggerated example, an organization wouldn't want to try to put Star Trek's transporters in their building currently, this project is not technically feasible.

2. **Economic Feasibility**

This assessment typically involves a cost/ benefits analysis of the project, helping organizations determine the viability, cost, and benefits associated with a project before financial resources are allocated. It also serves as an independent project assessment and enhances project credibility— helping decision-makers determine the positive economic benefits to the organization that the proposed project will provide.

3. **Legal Feasibility**

This assessment investigates whether any aspect of the proposed project conflicts with legal requirements like zoning laws, data protection acts or social media laws. Let's say an organization wants to construct a new office building in a specific location. A feasibility study might reveal the organization's ideal location isn't zoned for that type of business. That organization has just saved considerable time and effort by learning that their project was not feasible right from the beginning.

4. **Operational Feasibility**

This assessment involves undertaking a study to analyze and determine whether and how well the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development.

5. **Scheduling Feasibility**

This assessment is the most important for project success; after all, a project will fail if not completed on time. In scheduling feasibility, an organization estimates how much time the project will take to complete.

When these areas have all been examined, the feasibility analysis helps identify any constraints the proposed project may face, including:

- Internal Project Constraints: Technical, Technology, Budget, Resource, etc.
- Internal Corporate Constraints: Financial, Marketing, Export, etc.
- External Constraints: Logistics, Environment, Laws, and Regulations, etc.

2.2MAIN ASPECTS

There are three aspects of feasibility to be considered namely.

1. Technical
2. Operational
3. Economical

1. **TECHNICAL**:

In the technical aspects one may consider the hardware equipment for the installation of the software. The system being centralized will required very little hardware appliances. Hence this helps the system to work smoothly with limited amount of working capitals.

2. **OPERATIONAL**:

In the operational aspects may think of the benefits of the workload that many a personal may have to share. This is eased out and the required output may be retrieved in a very short time. Thus there is accuracy in the work on time is also saved there will be very little work that needs to be performed.

3. **ECONOMICAL**:

Economical system is definitely feasible because the software requirement is less and the operational working for the system requires less number of recruits. This help introduction over-staffing and wastage funds.

We studied on the position to evaluate solution. Most important factors in this study were tending to overlook the confusion inherent in Application Development the constraints and the assumed studies. It can be started that it the feasibility study is to serve as a decision document it must answer three key questions.

1. Is there a new and better way to do the job that will benefit the user?

2. What are the costs and savings of the alternatives?

3. What is recommended?

2.3 Technical feasibility:

This centers on the existing computer system (hardware, software etc.) and to what extent it can support the proposed additional equipment .in this stage of study, we have collected information about technical tools available by which I could decide my system design as the technical requirements.

2.4 Operational Feasibility:

In this stage of study, we have checked the staff availability. I concentrate on knowledge of end users that are going to use the system. This is also called as behavioral feasibility in which I have studied on following aspects; people are inherently resistant to change, and computers have been known to facilitate change. An estimate has been made to how strong a reaction the user staff is having toward the development of a computerized system. It is common knowledge that computer installations have something to do with turnover. I had explained that there is need to educate and train the staff on new ways of conducting business.

2.5 Economical feasibility:

Economic analysis is the most frequently used method for evaluating the effectiveness of candidate system. More commonly known as cost\benefit analysis, the procedure is to determine the benefits and savings that benefits outweigh costs. The decision was to design and implement system because it is for having chanced to be approved. This is an on- going effort that improves the accuracy at each phase of the system life cycle.

In developing cost estimates for a system, I need to consider several cost elements. Among these is hardware personal facility. Operating and supply costs.

2.6 BENEFITS

Benefits of conducting a feasibility study:

- Improves project teams' focus
- Identifies new opportunities
- Provides valuable information for a "go/no-go" decision

- Narrows the business alternatives
- Identifies a valid reason to undertake the project
- Enhances the success rate by evaluating multiple parameters
- Aids decision-making on the project
- Identifies reasons not to proceed

2.7 SYSTEM REQUIREMENT SPECIFICATION

Any system can be designed after specifies the requirement of the user about that system. For this first of all gathered information from user by the preliminary investigation which is starting investigation about user requirement.

The data that the analysts collect during preliminary investigation are gathered through the various preliminary methods.

1) Documents Reviewing Organization

The analysts conducting the investigation first learn the organization involved in, or affected by the project. Analysts can get some details by examining organization charts and studying written operating procedures.

Collected data is usually of the current operating procedure:

- The information relating to clients, projects and students and the relationship between them was held manually.
- Managing of follow-ups was through manual forms.
- Complaints require another tedious work to maintain and solve.
- Payments details had to be maintained differently.

2) Gathering Information By Asking Questions

Interviewing is the most commonly used techniques in analysis. It is always necessary first to approach someone and ask them what their problems are, and later to discuss with them the result of your analysis.

3) Questionnaires

Questionnaires provide an alternative to interviews for finding out information about a system. Questionnaires are made up of questions about information sought by analyst. The questionnaire is then sent to the user, and the analyst analyzes the replies.

4) Electronic Data Gathering

Electronic communication systems are increasingly being used to gather information. Thus, itis possible to use electronic mail to broadcast a question to a number of users in an organization to obtain their viewpoint on a particular issue. In our project, with the help of Marg software solutions, I have sent questionnaire through electronic mail to twenty employees of the company and retrieved the information regarding the problem faced by existing system.

5) Interviews

Interview allows the analysts to learn more about the nature of the project request and reason of submitting it. Interviews should provide details that further explain the project and show whether assistance is merited economically, operationally or technically.

One of the most important points about interviewing is that what question you need to ask. It is often convenient to make a distinction between three kinds of question that is:

- Open questions
- Closed question
- Probes

Open questions are general question that establish a person's view point on a particular subject.

CHAPTER 3

DESIGN

3.1 INTRODUCTION

System is created to solve problems. One can think of the systems approach as an organized-way of dealing with a problem. In this dynamic world, the subject system analysis and design, mainly deals with the software development activities. Since a new system is to be developed, the one most important phases of software development life cycle is system requirement gathering and analysis. Analysis is a detailed study of various operations performed by a system and their relationship within and outside the system. Using the following steps, it becomes easy to draw the exact boundary of the new system under consideration.

All procedures, requirements must be analyzed and documented in the form of detailed DFDs, logical data structure and miniature specifications.

System analyses also include sub-dividing of complex process involving the entire system identification of data store and manual processes.

3.1.1 SYSTEM ANALYSIS

A **system** is created to solve problems. The systems approach can be viewed as an organized way of addressing a problem. In the dynamic world of software development, **System Analysis and Design** primarily deals with the activities involved in developing software solutions.

Since a new system is to be developed, one of the most important phases of the **Software Development Life Cycle (SDLC)** is **System Requirement Gathering and Analysis**.

Analysis involves a detailed study of the current system, leading to the specification of the new system. It is a detailed study of various operations performed by the system and their relationship both within and outside the system. Using the following steps, it becomes easier to define the exact **boundary** of the new system under consideration.

Steps for Drawing the Boundary of the New System:

- **Identify Problems and New Requirements:**

  Analyze the issues with the current system and define new requirements that need to be addressed.
- **Work Out the Pros and Cons:**

  Evaluate the strengths and weaknesses of the current system and propose solutions.
- **Documenting Procedures and Requirements:**

  All procedures and requirements must be analyzed and documented, typically in the form of **Data Flow Diagrams (DFDs)**, **logical data structures**, and **miniature specifications**.
- **Sub-dividing Complex Processes:**

  Break down complex processes and identify data stores and manual processes to be incorporated in the new system.

System Analysis Process

System analysis is conducted with the following key steps:

1. **Information Gathering**
2. **The Tools of Structured Analysis**
3. **Identification of Need**
4. **System Planning and Initial Investigation**
5. **Feasibility Study**

Initial Investigation:

- **Problem Definition and Project Initiation:**

  Define the core problems and initiate the project by gathering necessary information.

- **Determining the Requirements:**

  Identify and document the requirements for the system.

- **Needs Identification:**

  Identify the needs that the system must address.

- **Dimension of Planning:**

  Define the planning scope and dimension for the system.

- **Feasibility Study:**

  Conduct a feasibility study to determine if the system is practical and achievable.

- **System Performance Definition:**

  Define the performance metrics and objectives for the new system.

- **System Objectives Identification:**

  Identify the objectives the system needs to fulfill.

- **Description of Outputs:**

  Define the expected outputs of the system.

Preliminary Investigation

- The **Preliminary Investigation** is a major part of the system analysis phase, focusing on evaluating the merits of the project request and determining the feasibility of the proposed project.
- The purpose of this investigation is to collect information that helps the project team evaluate whether the project should move forward and assess its viability.
- During the preliminary investigation, **system analysts** engage with various stakeholders to gather facts about business operations and system requirements.

Parts of Preliminary Investigation:

1. **Request Clarification:**

The first step is to clearly define the needs and requirements of the system.

2. **Feasibility Study:**

Conduct a feasibility study to assess the practicality of the system development.

3. **Request Approval:**

After clarifying the request and determining feasibility, seek approval to move forward with the project.

4. **Determining the Requirements:**

The requirements must be properly defined. Often, requests from employees and users are not clearly specified, making it essential for system analysts to clarify and verify the project request.

5. **Collecting Information from Various Users:**

Information can be obtained by reviewing existing organizational documents, such as data on sales, complaints, and operational processes. Observing onsite activities provides further insight into the real system's operations.

3.2 SDLC

Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.

SDLC defines the complete cycle of development i.e. all the tasks involved in planning, creating, testing, and deploying a Software Product.

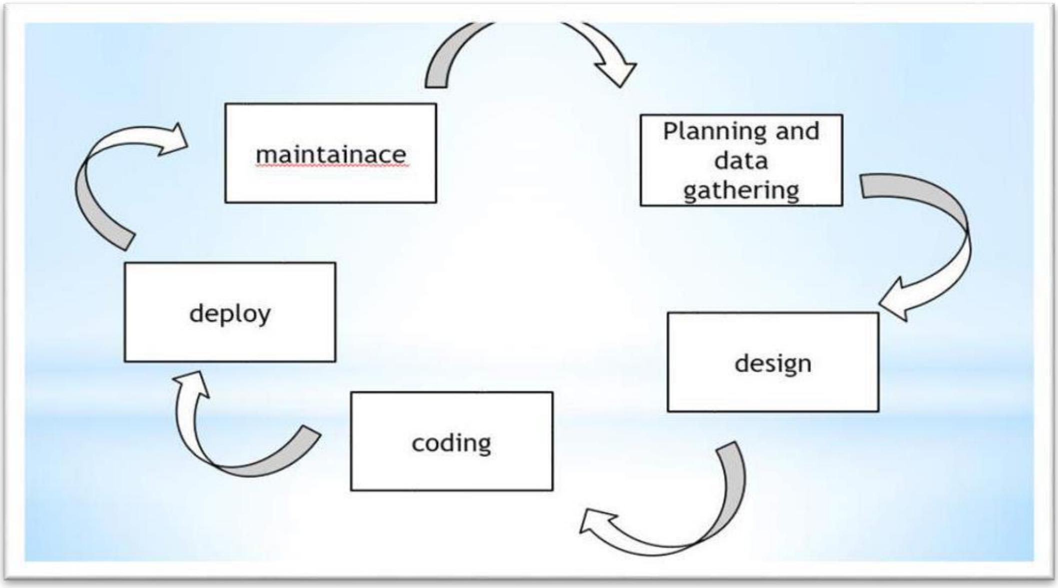


Figure 3.1: Above image depicting the planning step

SDLC Phases

Given below are the various phases:

- Requirement gathering and analysis
- Design
- Implementation or coding
- Testing
- Deployment
- Maintenance

1) Requirement Gathering and Analysis

During this phase, all the relevant information is collected from the customer to develop a product as per their expectation. Any ambiguities must be resolved in this phase only.

Business analyst and Project Manager set up a meeting with the customer to gather all the information like what the customer wants to build, who will be the end-user, what is the purpose of the product. Before building a product, a core understanding or knowledge of the product is very important.

For Example, A customer wants to have an application which involves money transactions. In this case, the requirement has to be clear like what kind of transactions will be done, how it will be done, in which currency it will be done, etc.

Once the requirement gathering is done, an analysis is done to check the feasibility of the development of a product. In case of any ambiguity, a call is set up for further discussion. Once the requirement is clearly understood, the SRS (Software Requirement Specification) document is created. This document should be thoroughly understood by the developers and also should be reviewed by the customer for future reference.

2) Design

- In this phase, the requirements gathered and documented in the
- **Software Requirements Specification (SRS)** are used as the primary input.
- Based on these requirements, a **software architecture** is designed to serve as a blueprint for the system's development.
- The design phase defines the **overall system structure**, including its components, data flow, control flow, interfaces, and interactions.
- It provides detailed specifications for each module, including **functional behavior**, **database structure**, and **external interfaces**.
- The objective is to ensure that the system design meets all technical and operational requirements and supports efficient and maintainable code development.
- The outcome of this phase includes **High-Level Design (HLD)** and **Low-Level Design (LLD)** documents, which guide the subsequent implementation phase.

3) Implementation or Coding

- Implementation or Coding is the phase where the actual development of the software begins.
- This phase starts once the design documents are finalized and shared with the development team.
- The software design is translated into source code using appropriate programming languages and development tools.
- Each component or module specified in the design phase is developed and implemented during this stage.
- Developers follow predefined coding standards and guidelines to ensure consistency, readability, and maintainability of the code.
- Version control systems are used to manage code versions and enable collaborative development.
- Unit testing is often performed during this phase to verify the correctness of individual modules as they are implemented.
- Proper code documentation is maintained to support future maintenance and updates.

4) Testing

Testing starts once the coding is complete and the modules are released for testing. In this phase, the developed software is tested thoroughly and any defects found are assigned to developers to get them fixed.
Retesting, regression testing is done until the point at which the software is as per the customer's expectation. Testers refer SRS document to make sure that the software is as 35per the customer's standard.

6) Deployment

Once the product is tested, it is deployed in the production environment or first UAT (User Acceptance testing) is done depending on the customer expectation.

In the case of UAT, a replica of the production environment is created and the customer along with the developers does the testing. If the customer finds the application as expected, then sign off is provided by the customer to go live.

5) Maintenance

After the deployment of a product on the production environment, maintenance of the product i.e., if any issue comes up and needs to be fixed or any enhancement is to be done is taken care by the developers.

3.3 SOFTWARE ENGG. PARADIGM A

Software engineering is a layered technology. The foundation for software engineering is the process layer. Software engineering processes the glue that holds the technology layers together and enables ratios and timely development of computer software. Process defines a framework for a set of key process areas that must be established for effective delivery of software engineering technology.
Method compasses a broad array of tasks that include requirements analysis, design, program construction, testing and support. Software engineering tools provide automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another tool, a system for the support of software development, called computer-aided software engineering is established.

The following paradigms are available:

1. The Waterfall Model
2. The Prototyping Model
3. The Spiral model Etc.

3.4 ARCHIETECTURE OF THE SYSTEM

An Architecture of a system in which the principal parts or functions are represented by blocks connected by lines that show the relationships of the blocks.

Its for higher level, less detailed descriptions that are intended to clarify overall concepts without concern for the details of implementation.

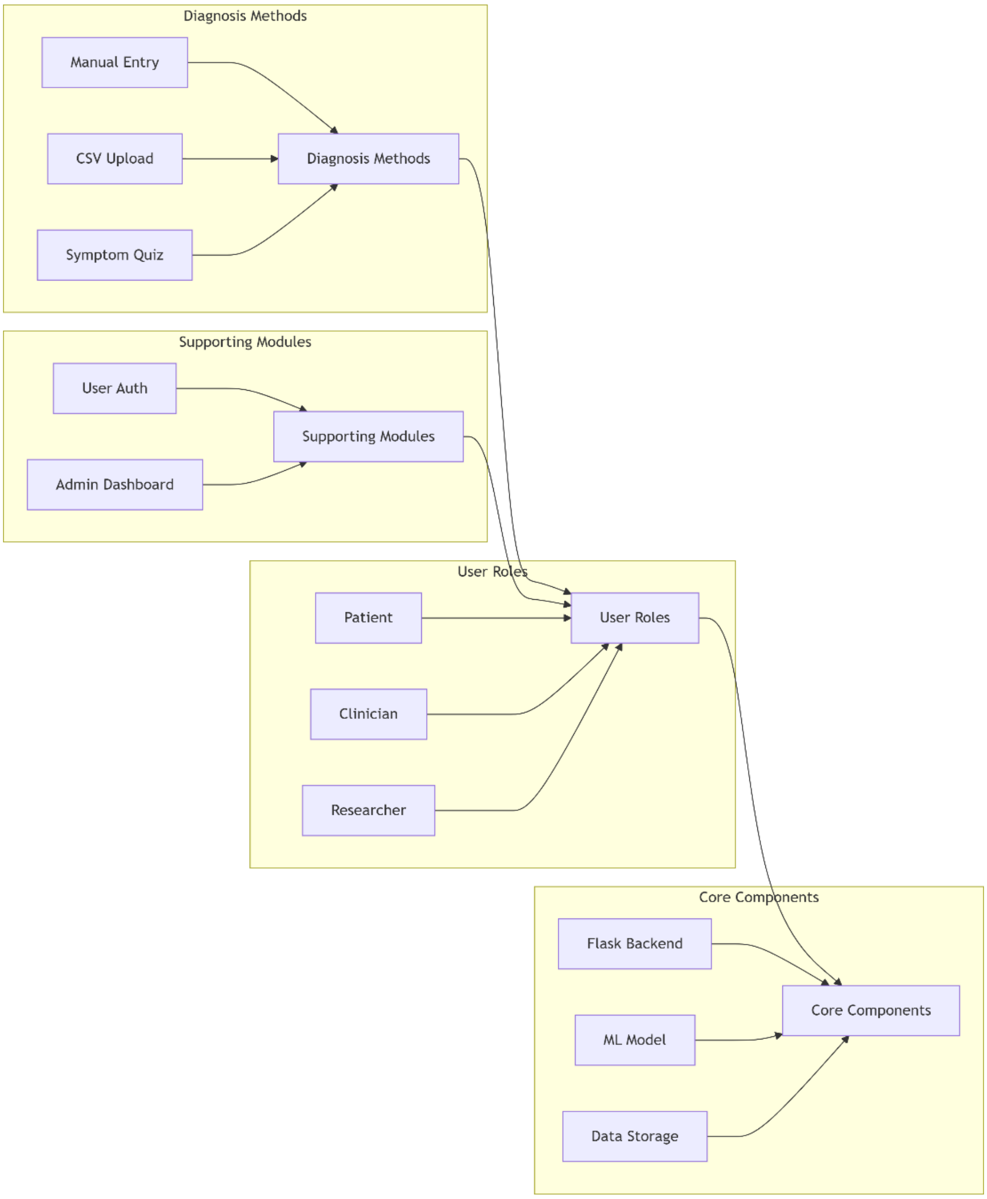


Figure 3.2: Architecture of the system

3.5 CONTROL FLOW GRAPH

A **Control Flow Graph (CFG)** is a graphical representation that illustrates the flow of control or computation during the execution of a program or a specific module. It is primarily used in **static analysis**, **compiler design**, and **software testing** to understand and analyze the control structure of a program. The CFG captures how the control moves from one part of the code to another, helping identify logical paths, unreachable code, and potential decision points. Each node in a CFG represents a **basic block** – a straight-line sequence of code with no branches (except into the entry and out of the exit). Each edge represents a **possible control transfer** between the basic blocks during execution.

Characteristics of Control Flow Graph:

- **Process-Oriented & Directed Graph**:

  The CFG is a process-oriented directed graph, where the direction of the edges indicates the flow of control.
- **Shows All Possible Execution Paths**:

  The CFG displays **all the paths** that can be traversed during program execution, helping to analyze program logic and behavior.
- **Nodes Represent Basic Blocks**:

  Each **node** in the CFG represents a **basic block** – a sequence of instructions with a single entry and exit point.
- **Edges Represent Control Flow Paths**:

  Edges in the CFG represent the **flow of control** from one basic block to another, showing the possible paths the program can take during execution.

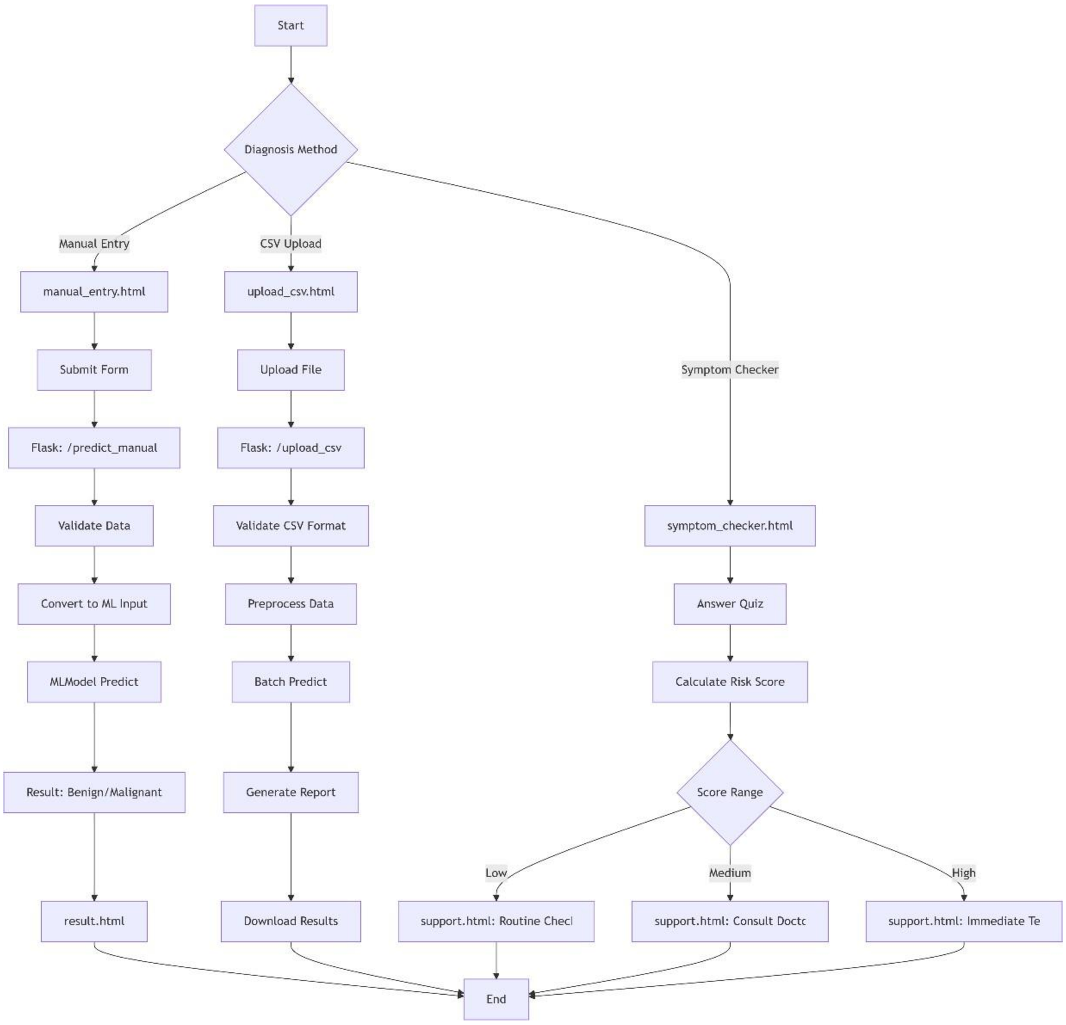


Figure 3.3: Control Flow Graph

CHAPTER 4

REPORT

4.1 GIST

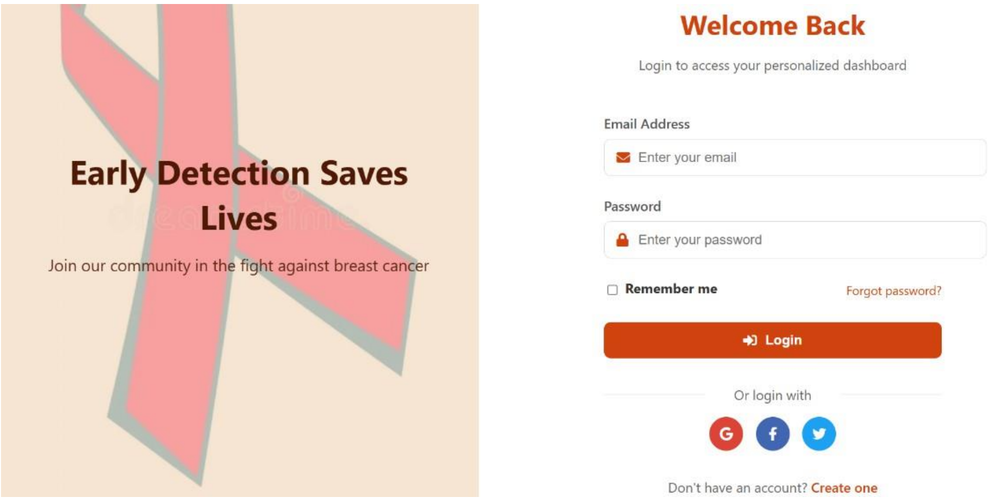
Breast Cancer Detection is an AI/ML-powered diagnostic platform designed to assist users in early identification and assessment of breast cancer risks. Whether you're a medical researcher, healthcare enthusiast, or someone concerned about personal health, this project offers an intelligent solution by leveraging data-driven insights. The platform simplifies the diagnosis process through three user-friendly modules: **Manual Entry**, **CSV Upload**, and **Symptom Checker**.

Users can manually input test parameters, upload clinical reports in CSV format, or take a short symptom-based quiz to predict the likelihood of benign or malignant outcomes. The system ensures that early warning signs are not overlooked and helps in taking preventive steps on time. With this project, making informed decisions about health and early detection becomes more accessible, reliable, and efficient.

4.2 SOME SNIPPETS

4.2.1 Web App

1. Sign In Page



Early Detection Saves Lives

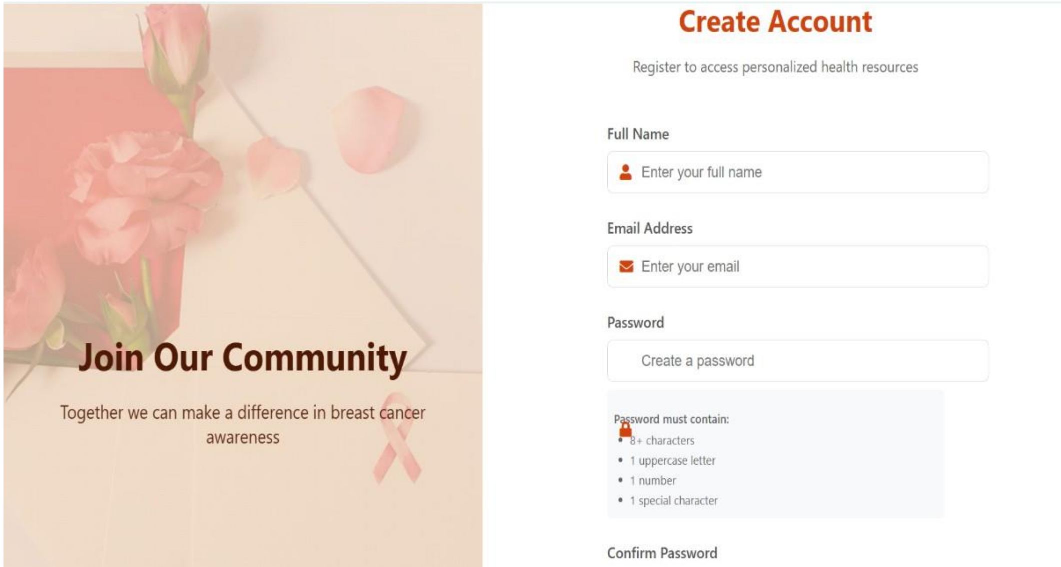
Join our community in the fight against breast cancer

Welcome Back

Login to access your personalized dashboard

The Sign In page is the entry point for users to access the Breast Cancer Detection system. It ensures that only authorized users can proceed further by verifying credentials through a secure login mechanism. Designed with a clean and user-friendly interface, the sign-in form accepts email and password inputs, guiding users smoothly into the platform.

2. Account Creation



Join Our Community

Together we can make a difference in breast cancer awareness

Create Account

Register to access personalized health resources

Full Name

Enter your full name

Email Address

Enter your email

Password

Create a password

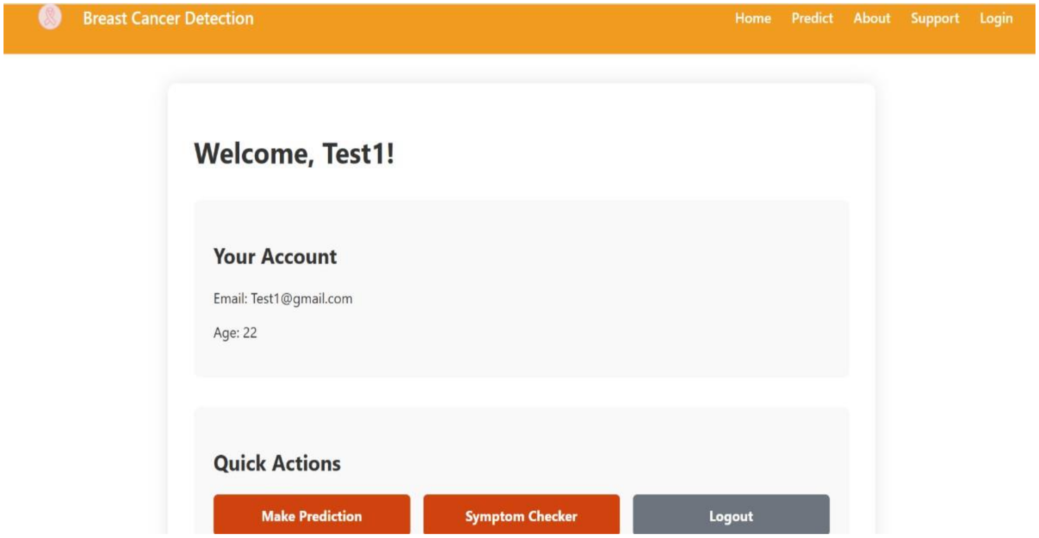
Password must contain:

- 8+ characters
- 1 uppercase letter
- 1 number
- 1 special character

Confirm Password

New users can register on the platform through the Account Creation page. It collects basic user information such as name, email, and password to securely create a user profile. The registration process is seamless and ensures that user data is securely stored and accessible only to authenticated users.

3. Dashboard



Breast Cancer Detection

Home Predict About Support Login

Welcome, Test1!

Your Account

Email: Test1@gmail.com

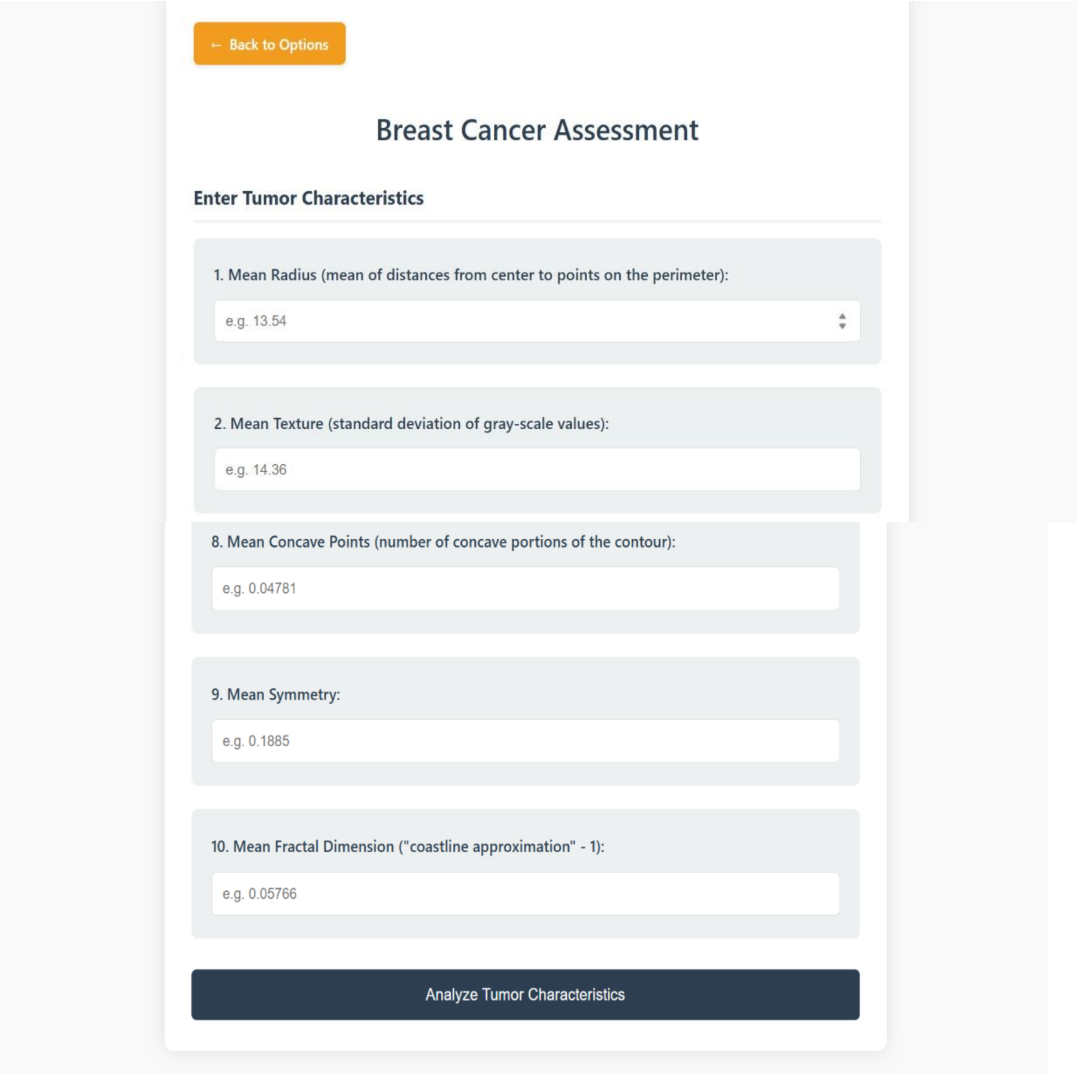
Age: 22

Quick Actions

Make Prediction Symptom Checker Logout

After a successful login, users are redirected to the Dashboard — a central hub that provides easy navigation to all major functionalities including Manual Entry, CSV Upload, and Symptom Checker. The dashboard offers a personalized space where users can manage their predictions, access insights, and receive alerts.

4. Manual Entry Form Interface



Breast Cancer Assessment

Enter Tumor Characteristics

1. Mean Radius (mean of distances from center to points on the perimeter):

e.g. 13.54

2. Mean Texture (standard deviation of gray-scale values):

e.g. 14.36

8. Mean Concave Points (number of concave portions of the contour):

e.g. 0.04781

9. Mean Symmetry:

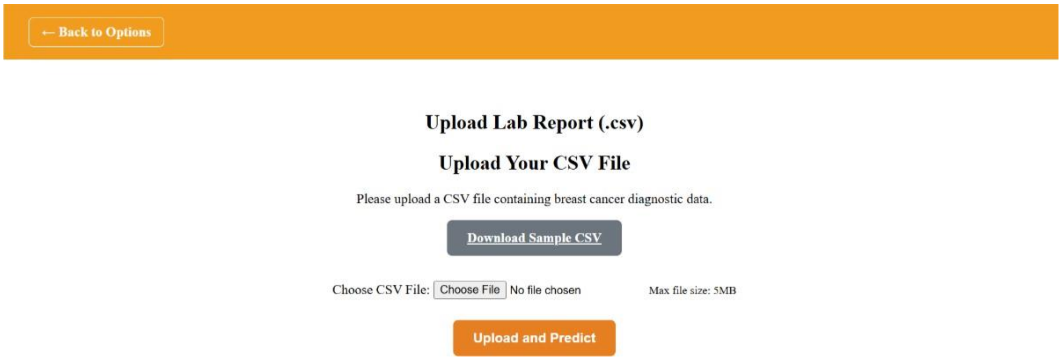
e.g. 0.1885

10. Mean Fractal Dimension ("coastline approximation" - 1):

e.g. 0.05766

This module allows users to manually input medical parameters (such as mean radius, texture, perimeter, etc.) to check for the likelihood of breast cancer. The form is designed to be simple and intuitive, with dropdowns or number fields validating entries. Upon submission, the system processes the inputs using a trained ML model to predict either benign or malignant status.

5. CSV Upload Interface



Upload Lab Report (.csv)

Upload Your CSV File

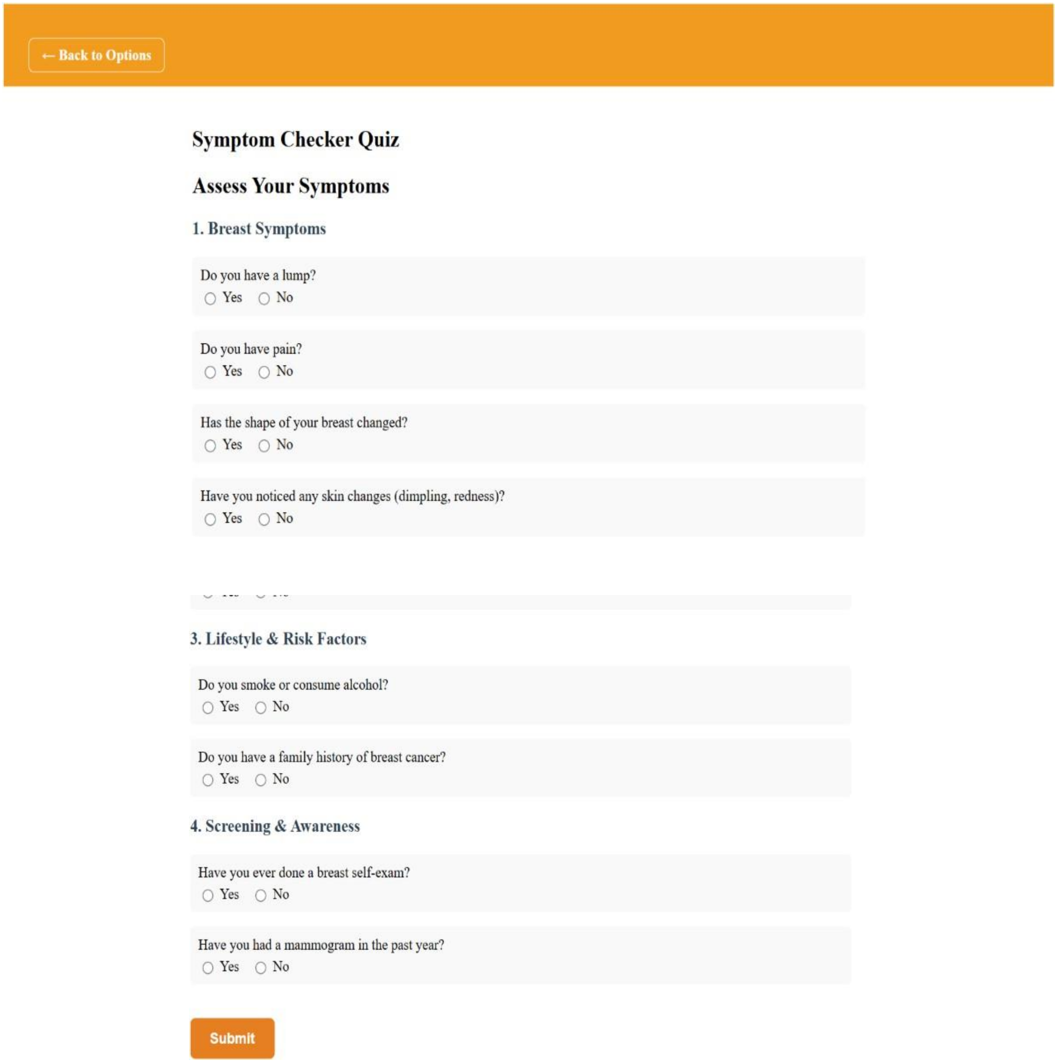
Please upload a CSV file containing breast cancer diagnostic data.

Choose CSV File: No file chosen

Max file size: 5MB

In the CSV Upload interface, users can directly upload a pre-formatted CSV file containing medical records. This feature is ideal for doctors or researchers handling batch patient data. Once uploaded, the system parses the file, extracts relevant features, and predicts outcomes in real-time for each entry.

6. Symptom Checker Quiz Page



Symptom Checker Quiz

Assess Your Symptoms

1. Breast Symptoms

Do you have a lump?
○ Yes ○ No

Do you have pain?
○ Yes ○ No

Has the shape of your breast changed?
○ Yes ○ No

Have you noticed any skin changes (dimpling, redness)?
○ Yes ○ No

3. Lifestyle & Risk Factors

Do you smoke or consume alcohol?
○ Yes ○ No

Do you have a family history of breast cancer?
○ Yes ○ No

4. Screening & Awareness

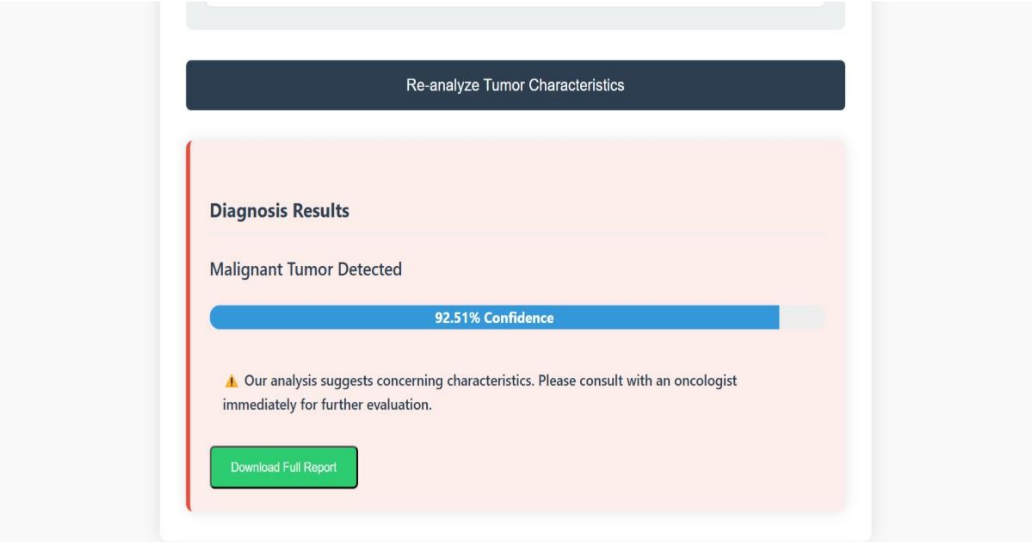
Have you ever done a breast self-exam?
○ Yes ○ No

Have you had a mammogram in the past year?
○ Yes ○ No

This module provides an interactive quiz for users who may not have medical reports but want to assess their risk. It includes questions related to personal history, symptoms, and lifestyle. Based on the responses, the system generates a risk score and categorizes the user as Low, Moderate, or High risk. This feature is helpful for awareness and early caution.

7. Prediction Output

After using any of the prediction modules, users are shown the results on the Prediction Output page. This includes the classification result (Benign or Malignant) or Risk Level (Low/Moderate/High), accompanied by a brief message suggesting next steps such as clinical consultation. The output page is visually informative and easy to interpret.

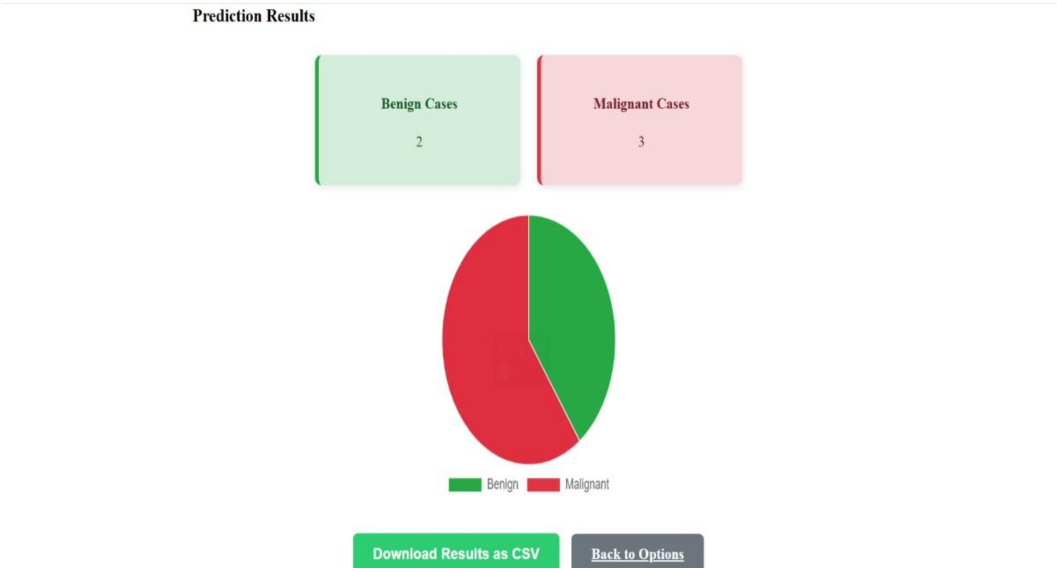


Diagnosis Results

Malignant Tumor Detected

92.51% Confidence

⚠ Our analysis suggests concerning characteristics. Please consult with an oncologist immediately for further evaluation.



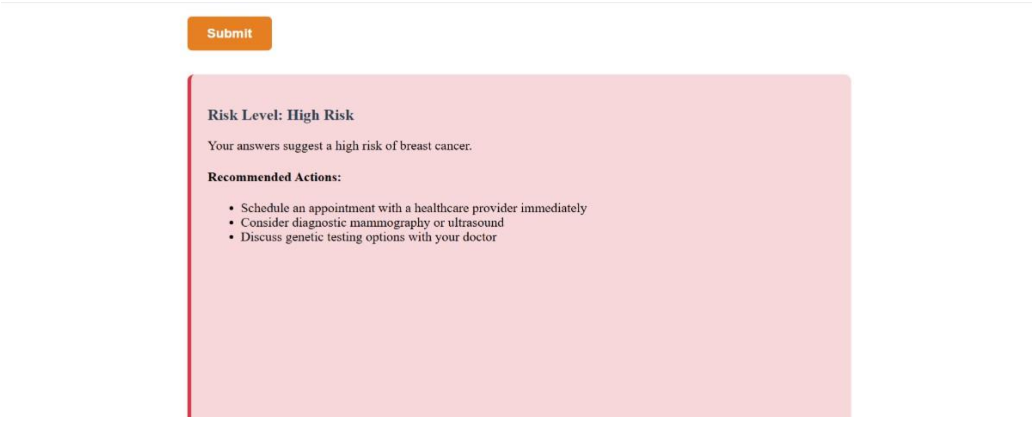
Prediction Results

Benign Cases

2

Malignant Cases

3



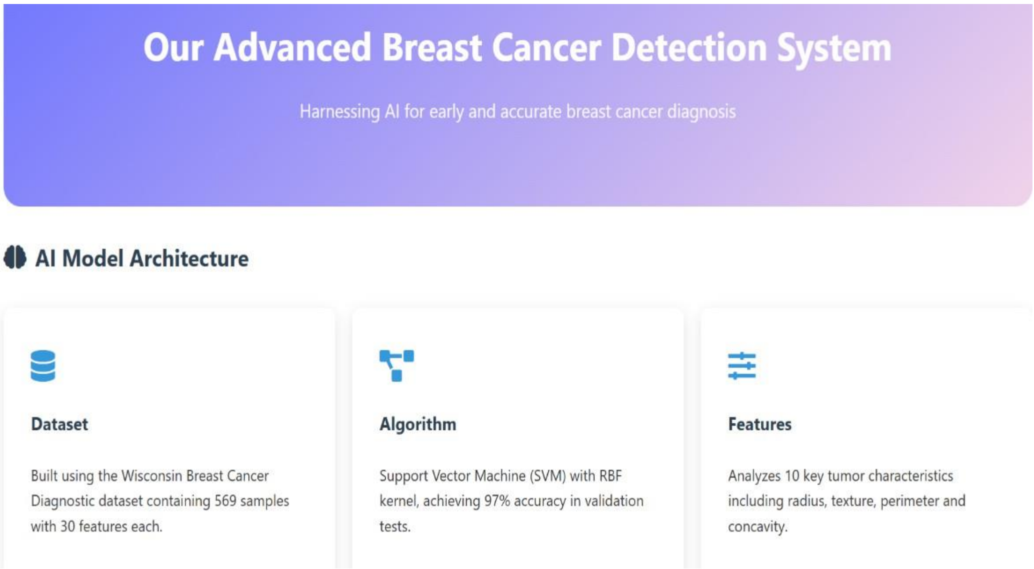
Risk Level: High Risk

Your answers suggest a high risk of breast cancer.

Recommended Actions:

- Schedule an appointment with a healthcare provider immediately
- Consider diagnostic mammography or ultrasound
- Discuss genetic testing options with your doctor

8. About Section



Our Advanced Breast Cancer Detection System

Harnessing AI for early and accurate breast cancer diagnosis

AI Model Architecture

Dataset

Built using the Wisconsin Breast Cancer Diagnostic dataset containing 569 samples with 30 features each.

Algorithm

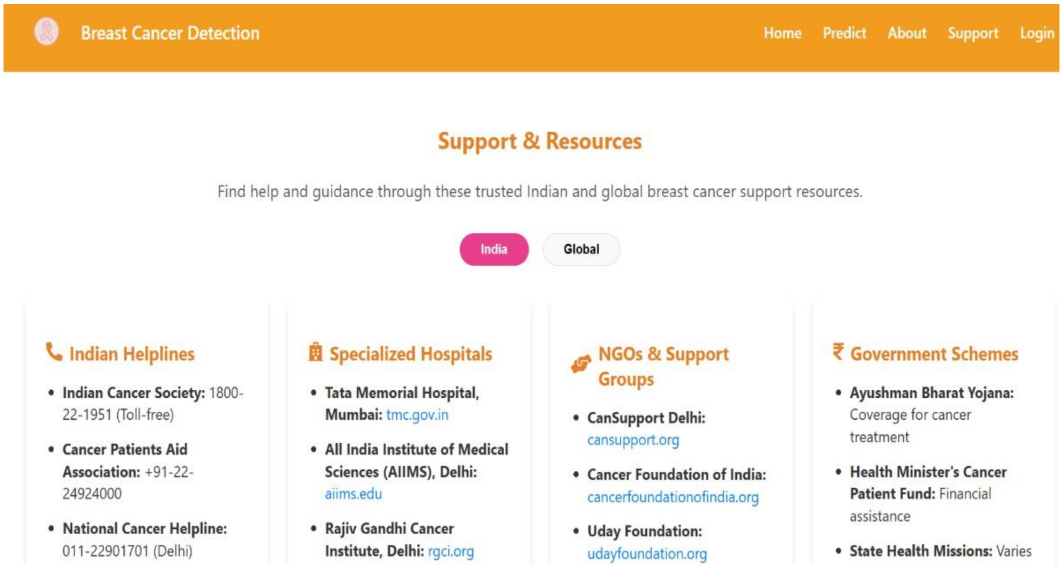
Support Vector Machine (SVM) with RBF kernel, achieving 97% accuracy in validation tests.

Features

Analyzes 10 key tumor characteristics including radius, texture, perimeter and concavity.

The About section gives users insight into the purpose and background of the project. It highlights the team members, the motivation behind building the platform, and the technologies used. This section adds credibility and transparency to the project.

9. Support Section



Support & Resources

Find help and guidance through these trusted Indian and global breast cancer support resources.

☎ Indian Helplines

- **Indian Cancer Society:** 1800-22-1951 (Toll-free)
- **Cancer Patients Aid Association:** +91-22-24924000
- **National Cancer Helpline:** 011-22901701 (Delhi)

Specialized Hospitals

- **Tata Memorial Hospital, Mumbai:** tmc.gov.in
- **All India Institute of Medical Sciences (AIIMS), Delhi:** aiims.edu
- **Rajiv Gandhi Cancer Institute, Delhi:** rgci.org

NGOs & Support Groups

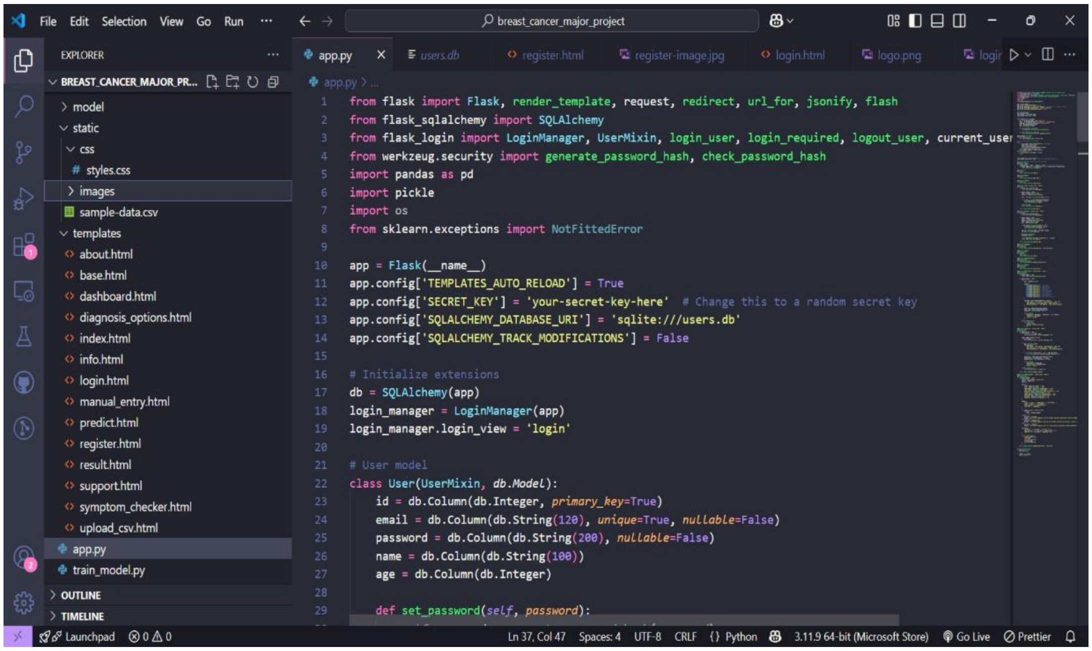
- **CanSupport Delhi:** cansupport.org
- **Cancer Foundation of India:** cancerfoundationofindia.org
- **Uday Foundation:** udayfoundation.org

₹ Government Schemes

- **Ayushman Bharat Yojana:** Coverage for cancer treatment
- **Health Minister's Cancer Patient Fund:** Financial assistance
- **State Health Missions:** Varies

The Support section serves as a help center for users needing guidance or assistance while using the platform. It includes FAQs, contact forms, and support email details to ensure that users can reach out in case of technical issues or queries related to breast cancer information and predictions.

4.2.2 VISUAL STUDIO CODE



```python
from flask import Flask, render_template, request, redirect, url_for, jsonify, flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user, login_required, logout_user, current_user
from werkzeug.security import generate_password_hash, check_password_hash
import pandas as pd
import pickle
import os
from sklearn.exceptions import NotFittedError

app = Flask(__name__)
app.config['TEMPLATES_AUTO_RELOAD'] = True
app.config['SECRET_KEY'] = 'your-secret-key-here'  # Change this to a random secret key
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

# Initialize extensions
db = SQLAlchemy(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'

# User model
class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    name = db.Column(db.String(100))
    age = db.Column(db.Integer)

    def set_password(self, password):
```

The screenshot shows the backend setup of the Breast Cancer Detection web application in VS Code. The app.py file configures the Flask app, sets up the database and user authentication, and defines the User model. The project is well-structured with separate folders for templates, static files, models, and scripts, making the application organized and scalable.

CHAPTER 5

CODING

This chapter contains some codes of the project. The goal of the coding is to translate the design of the system into code in a given programming language. For a given design, the aim of this phase is to implement the design in the best possible manner. The coding phase affects both testing and maintenance profoundly.

Some Codes are as Written below:

App.py :

```
from flask import Flask, render_template, request, redirect, url_for, jsonify,
flash
from flask_sqlalchemy import SQLAlchemy
from flask_login import LoginManager, UserMixin, login_user,
login_required, logout_user, current_user
from werkzeug.security import generate_password_hash,
check_password_hash
import pandas as pd
import pickle
import os
from sklearn.exceptions import NotFittedError
```

```
app = Flask(__name__)

app.config['TEMPLATES_AUTO_RELOAD'] = True
app.config['SECRET_KEY'] = 'your-secret-key-here'  # Change this to a random secret key
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

# Initialize extensions
db = SQLAlchemy(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'

# User model
class User(UserMixin, db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    name = db.Column(db.String(100))
    age = db.Column(db.Integer)

    def set_password(self, password):
        self.password = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password, password)

@login_manager.user_loader
def load_user(user_id):
    return db.session.get(User, int(user_id))

# Load the trained model and metadata
try:
    with open('model/breast_cancer_model.pkl', 'rb') as f:
        model_data = pickle.load(f)
        model = model_data['model']
        FEATURE_ORDER = model_data['feature_order']
    print("✅ Model loaded successfully")
    print(f"Model features: {FEATURE_ORDER}")
except Exception as e:
    print("❌ Failed to load model:", str(e))
    raise e

# login_manager = LoginManager(app)
login_manager.login_view = 'login'  # This sets the default login view

# Add this decorator to enforce login for all routes
@app.before_request
def require_login():
    allowed_routes = ['login', 'register', 'static']  # Routes that don't need auth
    if request.endpoint not in allowed_routes and not current_user.is_authenticated:
        return redirect(url_for('login'))

@app.route('/')
def home():
    return redirect(url_for('login'))

@app.route('/about')
def about():
    return render_template('about.html')

@app.route('/support')
def support():
    return render_template('support.html')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')
        user = User.query.filter_by(email=email).first()

        if not user or not user.check_password(password):
            flash('Invalid email or password', 'error')
            return redirect(url_for('login'))

        login_user(user)
        return redirect(url_for('dashboard'))

    return render_template('login.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if current_user.is_authenticated:
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')
        name = request.form.get('name')
        age = request.form.get('age')

        if User.query.filter_by(email=email).first():
            flash('Email already registered', 'error')
            return redirect(url_for('register'))

        new_user = User(email=email, name=name, age=age)
        new_user.set_password(password)

        db.session.add(new_user)
        db.session.commit()

        flash('Registration successful! Please log in.', 'success')
        return redirect(url_for('login'))
    return render_template('register.html')

@app.route('/dashboard')
@login_required
def dashboard():
    return render_template('dashboard.html', user=current_user)

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('home'))

@app.route('/diagnosis-options')
@login_required
def diagnosis_options():
    return render_template('diagnosis_options.html')

@app.route('/predict')
@login_required
def predict():
    return render_template('predict.html')

@app.route('/manual-entry', methods=['GET', 'POST'])
@login_required
def manual_entry():
    if request.method == 'GET':
        return render_template('manual_entry.html')

    if request.method == 'POST':
        try:
            data = request.get_json()

            features = [
                float(data['feature_1']),  # mean radius
                float(data['feature_2']),  # mean texture
                float(data['feature_3']),  # mean perimeter
                float(data['feature_4']),  # mean area
                float(data['feature_5']),  # mean smoothness
                float(data['feature_6']),  # mean compactness
                float(data['feature_7']),  # mean concavity
                float(data['feature_8']),  # mean concave points
                float(data['feature_9']),  # mean symmetry
                float(data['feature_10'])  # mean fractal dimension
            ]

            input_data = pd.DataFrame([features], columns=[
                'mean radius', 'mean texture', 'mean perimeter', 'mean area',
                'mean smoothness', 'mean compactness', 'mean concavity',
                'mean concave points', 'mean symmetry', 'mean fractal dimension'
            ])

            prediction = model.predict(input_data)[0]
            proba = model.predict_proba(input_data)[0]
            confidence = round(max(proba) * 100, 2)

            return jsonify({
                "prediction": "Malignant" if prediction == 1 else "Benign",
                "confidence": confidence,
                "status": "success"
            })

        except Exception as e:
            return jsonify({
                "error": str(e),
                "message": "Invalid input data"
            }), 400

@app.route('/upload-csv', methods=['GET', 'POST'])
@login_required
def upload_csv():
    if request.method == 'POST':
        if 'csv_file' not in request.files:
            return jsonify({"error": "No file uploaded"}), 400

        file = request.files['csv_file']
        if file.filename == '':
            return jsonify({"error": "No file selected"}), 400

        try:
            data = pd.read_csv(file)
            required_columns = [
                'mean radius', 'mean texture', 'mean perimeter', 'mean area',
                'mean smoothness', 'mean compactness', 'mean concavity',
                'mean concave points', 'mean symmetry', 'mean fractal dimension'
            ]

            if not all(col in data.columns for col in required_columns):
                return jsonify({"error": "Missing required columns."}), 400

            predictions = model.predict(data[required_columns])
            benign_count = int((predictions == 0).sum())
            malignant_count = int((predictions == 1).sum())

            return jsonify({
                "benign_count": benign_count,
                "malignant_count": malignant_count,
                "status": "success"
            })
        except Exception as e:
            return jsonify({"error": str(e)}), 500

    return render_template('upload_csv.html')

@app.route('/symptom_checker', methods=['GET', 'POST'])
@login_required
def symptom_checker():
    if request.method == 'POST':
```

```
    print("✅ Form received:", request.form)
    answers = request.form
    score = 0

    factors = {
        'Lump': answers.get('lump') == 'yes',
        'Pain': answers.get('pain') == 'yes',
        'Shape Change': answers.get('shape_change') == 'yes',
        'Skin Change': answers.get('skin_change') == 'yes',
        'Nipple Irritation': answers.get('nipple_irritation') == 'yes',
        'Discharge': answers.get('discharge') == 'yes',
        'Family History': answers.get('family_history') == 'yes',
        'Smoking/Alcohol': answers.get('smoke_alcohol') == 'yes',
        'Self Exam': answers.get('self_exam') == 'no',
        'Mammogram': answers.get('mammogram') == 'no'
    }

    weights = {
        'Lump': 2, 'Pain': 1, 'Shape Change': 2, 'Skin Change': 2,
        'Nipple Irritation': 1, 'Discharge': 2,
        'Family History': 2, 'Smoking/Alcohol': 1,
        'Self Exam': 1, 'Mammogram': 1
    }

    for symptom, active in factors.items():
        if active:
            score += weights[symptom]

    if score <= 3:
        risk_level = 'Low Risk'
        message = 'Your answers suggest a low risk. Continue monitoring and consult a doctor if needed.'
    elif score <= 6:
        risk_level = 'Moderate Risk'
        message = 'Your answers suggest a moderate risk. Consider scheduling a screening or consult with a doctor.'
        else:
            risk_level = 'High Risk'
                message = 'Your answers suggest a high risk. Please consult a
healthcare provider immediately.'

        chart_data = {
            'Low Risk': 1 if risk_level == 'Low Risk' else 0,
            'Moderate Risk': 1 if risk_level == 'Moderate Risk' else 0,
            'High Risk': 1 if risk_level == 'High Risk' else 0
        }

        return render_template(
            'symptom_checker.html',
            score=score,
            risk_level=risk_level,
            message=message,
            chart_data=chart_data
        )

    return render_template('symptom_checker.html', score=None)

# Create database tables
with app.app_context():
    db.create_all()

if __name__ == '__main__':
    app.run(debug=True)
```

Train Model :

```
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

```
import pickle
import os

# Create model directory if it doesn't exist
os.makedirs('model', exist_ok=True)

def train_and_save_model():
    # Load the dataset
    data = load_breast_cancer()
    X = pd.DataFrame(data.data, columns=data.feature_names)
    y = pd.Series(data.target)

    # Define feature order explicitly (must match form inputs)
    FEATURE_ORDER = [
        'mean radius',         # feature_1
        'mean texture',        # feature_2
        'mean perimeter',      # feature_3
        'mean area',           # feature_4
        'mean smoothness',     # feature_5
        'mean compactness',    # feature_6
        'mean concavity',      # feature_7
        'mean concave points',# feature_8
        'mean symmetry',       # feature_9
        'mean fractal dimension' # feature_10
    ]

    # Filter and order features
    X_selected = X[FEATURE_ORDER]

    # Split the data
    X_train, X_test, y_train, y_test = train_test_split(
        X_selected, y, test_size=0.2, random_state=42
    )

    # Train model with optimized parameters
    model = SVC(
        kernel='linear',
        probability=True,  # Required for confidence scores
        random_state=42,
        C=0.1  # Regularization parameter
    )
    model.fit(X_train, y_train)
        # Evaluate model
    train_acc = accuracy_score(y_train, model.predict(X_train))
    test_acc = accuracy_score(y_test, model.predict(X_test))

    # Save the model and metadata
    model_data = {
        'model': model,
        'feature_order': FEATURE_ORDER,
        'accuracy': {
            'train': train_acc,
            'test': test_acc
        }
    }
    with open('model/breast_cancer_model.pkl', 'wb') as f:
        pickle.dump(model_data, f)
        print("✅ Model trained and saved successfully")
    print(f"  - Training accuracy: {train_acc:.2%}")
    print(f"  - Test accuracy: {test_acc:.2%}")
    print("  - Feature order:", FEATURE_ORDER)
if __name__== '__main__':
    train_and_save_model()
```

CHAPTER 6

TESTING

6.1 INTRODUCTION

6.1.1 Testing Objectives

The following are the testing objectives:

- -Testing is a process of executing a program with the intent of finding an error.
- -A good test case is one that has a high probability of finding an as-yet- undiscovered error successful test is one that uncovers an as yet undiscovered error.

6.1.2 Testing Principles

The basic principles that guide software testing are as follows:

- -All tests should be traceable to customer requirements.
- -Tests should be planned long before testing begins.
- -The pirate principle applies to software testing.

Pareto principle states that 80 percent of all errors uncovered during testing will likely be traceable to 20 percent of all program components. Testing should begin “in the small “and progress toward testing “in the large.” Exhaustive testing is not possible.

6.2 LEVEL OF TESTING

There are different levels of testing

->Unit Testing

->Integration Testing

->System Testing

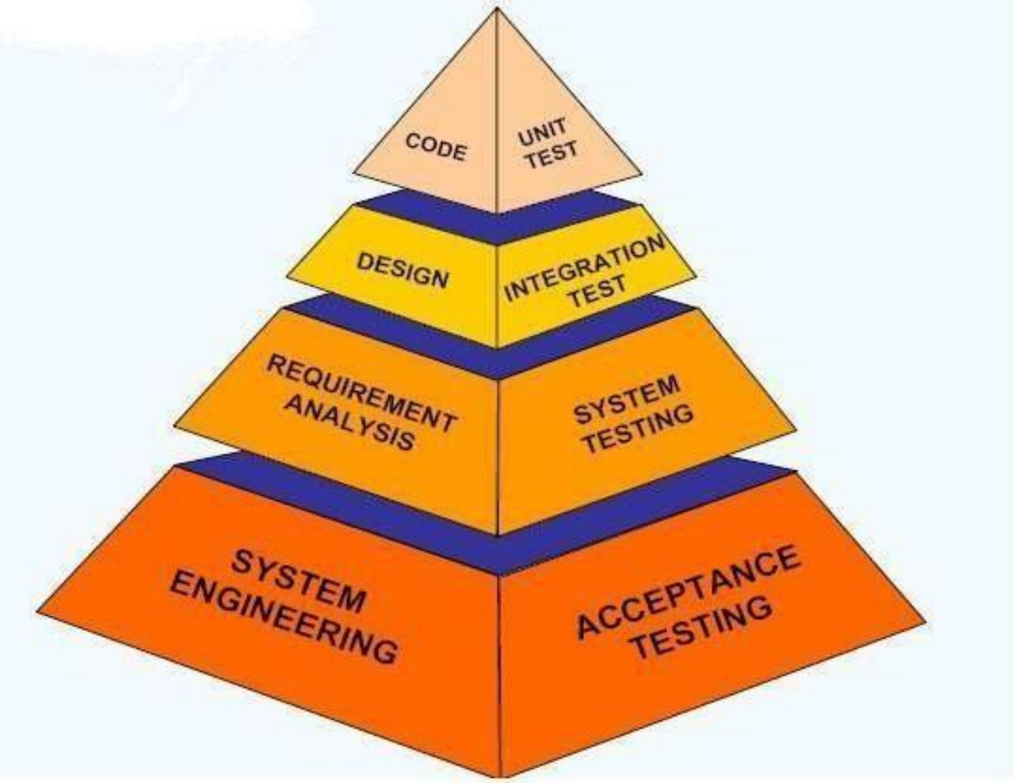


Figure 5.1: Testing pyramid

6.2.1 Unit testing

Unit testing is the initial level of software testing where individual components or modules of a software application are independently tested in isolation. The primary objective of unit testing is to validate that each software unit performs as expected and fulfills its intended functionality. A "unit" in this context typically refers to the smallest

testable part of the application, such as a function, method, procedure, or class.

This phase of testing is crucial because it helps detect and fix bugs early in the development cycle, reducing the cost and effort required to resolve them later. Developers usually perform unit testing during the coding phase to ensure that each module functions correctly before it is integrated with other modules.

Key Focus Areas of Unit Testing:

- **Verification of Individual Units:**

  Each software module is tested independently to confirm that it executes the expected functionality. This includes checking business logic, internal algorithms, calculations, and data handling within that module.

- **Testing Control Paths:**

  Critical control paths within a module are tested to verify logical correctness and ensure that all branches and conditions function as intended. This includes testing loops, decision-making structures, and conditional statements.

- **Interface Testing:**

  The module’s interface is tested to verify that it correctly accepts input and produces the expected output. Proper data flow into and out of the module is essential to ensure correct interaction with other modules or external components during integration.

- **Boundary Condition Testing:**

  Special attention is given to boundary conditions such as minimum and maximum input values, buffer sizes, and data limits. Testing these conditions ensures that the module operates correctly at the edges of its input domain and gracefully handles any limitations or constraints imposed on its processing logic.

- **Input Data Simulation:**

  Test data is provided through custom testing scripts or dedicated input screens. These are designed to simulate realistic and edge-case scenarios that the module may encounter in a live environment. This helps ensure robustness and reliability of the unit.

Benefits of Unit Testing:

- Detects bugs early in the development process, minimizing future debugging efforts.
- Simplifies code debugging by isolating problems within specific modules.
- Encourages modular, maintainable, and testable code development.
- Facilitates refactoring by ensuring existing functionality remains intact.
- Enhances code documentation by providing insight into what the unit is supposed to do.

Tools Commonly Used for Unit Testing:

- **JUnit** (Java)
- **NUnit** (.NET)
- **PyTest / unittest** (Python)
- **Jest / Mocha** (JavaScript)
- **Google Test** (C++)

6.2.2 Integration testing

Integration testing is a systematic and structured phase in the software testing life cycle that focuses on verifying the interactions and data flow between integrated modules. After individual modules have been successfully tested during unit testing, they are gradually combined and tested as a group to ensure that they function cohesively as a larger system.

The primary goal of integration testing is to detect and resolve issues related to the interfaces between modules. These issues may include mismatched data formats, incorrect function calls, broken communication links, or improper handling of data passed from one module to another. Integration testing ensures that the software components interact correctly and that the overall structure of the application aligns with the intended design.

Purpose of Integration Testing:

- To ensure that combined modules communicate and interact correctly.
- To identify interface-related defects that may arise when units are linked together.
- To validate the flow of data and control across modules.
- To build and verify the program structure as specified in the software design documents.
- To detect integration-level issues such as incorrect assumptions between modules, unhandled exceptions, and configuration problems.

Integration Testing Process:

Integration testing is typically performed in a step-by-step manner, guided by the software architecture or design. Various strategies are used to conduct integration testing based on the complexity of the system and the dependencies between modules. Some common strategies include:

- **Top-Down Integration Testing**

  Begins testing from the top-level modules and progressively integrates lower-level modules. Stubs are used to simulate lower modules if they are not yet developed.

- **Bottom-Up Integration Testing**

  Begins with testing of lower-level modules first, followed by integration of higher-level modules. Drivers are used to simulate higher modules.

- **Big Bang Integration Testing**

  All modules are integrated simultaneously, and the entire system is tested in one go. While this approach is simple, it can be difficult to isolate the cause of errors.

- **Incremental Integration Testing**

  Modules are integrated and tested one by one in incremental steps. This approach allows easier identification and resolution of defects.

Key Focus Areas in Integration Testing:

- **Interface Verification:**

  Ensuring that data passed between modules is accurate and consistent with expected formats and parameters.

- **Error Detection in Interaction:**

  Identifying logical errors that may occur when modules interact, such as incorrect logic flow, mismatched data types, or unhandled exceptions.

- **System Design Conformance:**

  Verifying that the constructed program structure adheres to the architecture and design specifications outlined during the design phase.

Benefits of Integration Testing:

- Detects errors that occur when modules are combined.
- Reduces the risk of failure in the later stages of testing.
- Ensures consistency between integrated units.
- Validates the overall software structure and module interaction.
- Provides a strong foundation for system testing.

Tools Commonly Used for Integration Testing:

- **JUnit**, **TestNG** – for Java-based applications
- **PyTest**, **unittest. mock** – for Python
- **Postman**, **SoapUI** – for API-level integration testing
- **Selenium**, **Cypress** – for web UI integration testing
- **Jenkins**, **Maven** – for continuous integration support

6.2.3 System testing

- System testing is a critical phase in the software testing life cycle where the complete and integrated software is tested as a whole. It validates the end-to-end functionality of the application in an environment similar to the actual production setup.

Objectives of System Testing:

- To ensure the software application works as a fully integrated system.
- To validate that the software meets all specified functional and non-functional requirements.
- To identify defects related to system interactions and workflows.
- To confirm that the system behaves correctly under expected and extreme conditions.
- To verify the implementation of customer requirements.

- **Key Questions Addressed in System Testing:**
- Does the entire application function as expected?
- Are integrated modules interacting correctly?
- Are performance standards such as speed and response time met?
- Are all business requirements and use cases covered?

Types of Testing Included in System Testing:

1. Functional Testing

- Ensures that the software functions as specified in the requirements.
- Validates the implementation of business logic and workflows.
- Tests user inputs, data processing, and output generation.
- Checks data validation, navigation, and overall functionality.
- **Common Techniques:**
- Smoke Testing
- Sanity Testing
- Regression Testing
- Boundary Value Analysis
- Equivalence Partitioning

2. Performance Testing

- Evaluates the responsiveness and stability of the system.
- Assesses the system's behavior under various loads.
- **Types of Performance Testing:**
- **Load Testing** – Tests the system under expected user load.
- **Stress Testing** – Evaluates behavior under extreme conditions.
- **Scalability Testing** – Measures the ability to scale with increasing demand.
- **Stability Testing** – Verifies consistent performance over time.
- Response Time
- Throughput
- Memory Usage
- CPU Utilization

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 CONCLUSION

The **Breast Cancer Detection System** presents an innovative and user-friendly approach to early diagnosis using machine learning techniques integrated into a web-based platform. The primary goal of the project was to create a system that could assess symptoms and input data to predict the likelihood of breast cancer, empowering users to seek medical guidance at an early stage.

By incorporating features such as manual data entry, CSV uploads, symptom checker quizzes, and a clear diagnosis result interface, the system ensures both accessibility and reliability for users with varied technical backgrounds. The integration of Flask, HTML/CSS, and ML libraries also ensures that the application remains lightweight, responsive, and effective in processing and predicting health outcomes.

Key Features of the Breast Cancer Detection Project:

- **Symptom-Based Diagnosis:** Users can input symptoms through interactive forms and quizzes to get a personalized risk score.
- **Dual Entry Modes:** Manual entry and CSV upload options enable both individual users and healthcare providers to assess multiple records.
- **Clear Output:** The platform provides intuitive prediction results, enabling users to make informed decisions.
- **User-Centric Design:** A clean dashboard, informative support section, and easy account creation streamline user experience.

Benefits of the Project Hub:

- **Promotes Early Detection:** Helps users identify risks at an early stage, potentially leading to better treatment outcomes.
- **Accessibility:** Enables anyone with an internet connection to assess symptoms without needing technical knowledge.
- **Data Privacy:** Users can securely manage their data in a personal account space.

7.2 FUTURE SCOPE

- **Integration with Health Databases:** The system can be linked with hospital databases or EHR systems to pull real-time patient history and provide more accurate predictions.
- **Mobile Application Development:** Creating a mobile app version of the platform would provide ease of access, enabling users to manage their profiles and check symptoms on the go.
- **Incorporating Advanced Algorithms:** Future versions can utilize deep learning models and ensemble techniques to enhance accuracy and provide multi-stage diagnosis suggestions.
- **Multi-language Support:** To reach a wider demographic, the platform can include multiple language options for both regional and international users.
- **Real-time Doctor Consultation:** Integration with telehealth services to connect users directly with certified oncologists based on their risk assessment.
- **Improved Visualizations:** Adding more advanced result visualizations and charts can help users understand their health status better.
- **Health Tracker Integration:** The system can be expanded to sync with fitness trackers and wearables to monitor ongoing health parameters and suggest risk alerts.
- **Enhanced Security:** Implementing two-factor authentication and secure encryption protocols to protect sensitive health data.

BIBLIOGRAPHY

- **UCI Machine Learning Repository - Breast Cancer Wisconsin (Diagnostic) Data Set**
  https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)
  (Used as the primary dataset for training and evaluating the machine learning model.)
- **Flask - Web Development Framework**
  https://flask.palletsprojects.com/
  (Used for developing the backend of the web application.)
- **Scikit-learn - Machine Learning in Python**
  https://scikit-learn.org/stable/
  (Used for model creation, training, and evaluation of predictive algorithms.)
- **Pandas - Python Data Analysis Library**
  https://pandas.pydata.org/
  (Used for data preprocessing and manipulation.)
- **HTML, CSS, and JavaScript Web Technologies**
  https://developer.mozilla.org/en-US/docs/Web
  (Used for designing and styling the frontend of the application.)
- **SQLAlchemy - Python SQL Toolkit and ORM**
  https://www.sqlalchemy.org/
  (Used for database interaction and user account management.)