**Chapter 3**

**METHODOLOGY**

**3.1 Introduction**

In this chapter, the methodology used for the project development is discussed briefly.

**Methodology**

The project was developed using a combination of technologies and tools, including:

* **Kaggle:** The dataset of plant images used in the project was obtained from Kaggle.
* **Python:** The plant disease detection and classification models were developed using Python, using libraries such as OpenCV, TensorFlow, and Keras.
* **Streamlit:** The Python models were integrated into a web application using Streamlit, allowing users to easily upload images and receive predictions.
* **MERN Stack:** The web application was built using the MERN stack (MongoDB, Express, React, Node.js), with the front-end user interface developed in React and the back-end server developed using Node.js and Express. The MongoDB database was used to store user queries and expert responses.
* **IFrame tag:** To link the Python and Streamlit app to the MERN app, we used an iframe tag in the React front-end to embed the Streamlit app within the MERN app.

This combination of technologies allowed us to create a robust and user-friendly plant disease detection, classification, and treatment web application. Users can easily upload images of plant leaves, receive accurate predictions of the plant disease, and access information about treatments. The MERN stack provides a solid foundation for the app, allowing for easy scalability and maintenance, while the Python and Streamlit components provide powerful and accurate disease detection and classification.

Furthermore, the addition of an iframe tag linking the Python and Streamlit app to the MERN app enables users to easily access the plant disease detection and classification models without leaving the main app interface. Expert responses to user queries are also easily accessible through the MERN app, making it a one-stop-shop for all plant-related needs.

***Login Authentication***

For this project, a login authentication system has been implemented to ensure that users have secure access to their queries and can view only their own queries. The bcrypt npm module has been used to encrypt user passwords, which ensures that even if the database is compromised, the users' passwords remain safe and secure. This ensures that the user's privacy and security is maintained throughout the application.

**3.3 SDLC**

**3.3.1 Introduction**

For the development of this website, Iterative model[[1]](#footnote-1) of SDLC is used. In this life cycle model, a Project Control List (PCL) on the basis of current known requirements is developed. A PCL is a list containing the series of tasks/functionalities that are to be present in the given system. If at certain phase of development, we come across any new requirement, we add it to our Project Control List.

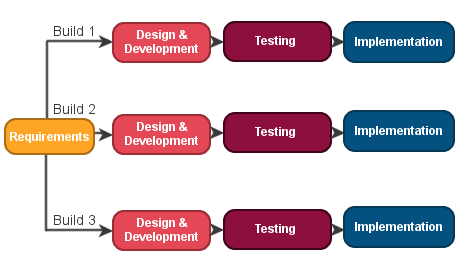


Figure 1: Iterative Model

For developing the website, a task from is chosen from the given PCL and Planning, Analysis, Designing, Testing and Evaluation is performed. When the specific functionality is added we remove it from the Project Control List. In similar way, one task at a time from PCL is chosen, implemented and then removed from PCL. This process iterates until the desired requirements of product are not met. The Website is made on the basis of two PCLs, the first was initiated at the time of Project Assignment and the second was created after certain requirements were added by the supervisor.

**3.3.2 Project Control List I**

* Once the user is logged in, they can access the query asking section.
* In this section, the user can register a new query by providing a title, description, and tag related to their problem.
* The registered query is then saved in the database for future reference.
* The user can also see their previously registered queries on the same page, displayed below the query asking section.
* This allows the user to easily track the status of their queries and view the solutions provided by experts.
* The query asking section allows users to submit their queries through a form that includes fields for title, description, and tags.
* After submitting a query, the user can view their previous queries on the same page below the query asking section.
* Users can search for previous queries based on tags or keywords using the search bar.
* The query asking section also includes a feature for experts to mark a query as resolved once they have provided a solution for the user.

**3.3.3 Project Control List II**

1. The first step is to collect the dataset of images of diseased and healthy plants. This dataset can be obtained from various sources, such as Kaggle or through web scraping.
2. Once the dataset is collected, the images are preprocessed to enhance the quality of the images and to remove any unwanted noise or distortion.
3. The preprocessed images are then passed through a pre-trained deep learning model such as VGG16 or ResNet to detect the presence of disease in the plant. The model is trained on the dataset of diseased and healthy plants to learn the features of the images.
4. The output of the model is the predicted class of the image, which can be either healthy or diseased. If the image is predicted to be diseased, then the specific disease is identified by analyzing the pattern of the symptoms on the plant.
5. Once the disease is identified, the appropriate treatment is recommended to the user. This can be done by providing a description of the disease and the recommended treatment or by providing a link to a reliable source of information.
6. To improve the accuracy of the model, data augmentation techniques such as rotation, scaling, and flipping are used to generate additional training data.
7. Transfer learning is another technique that can be used to improve the accuracy of the model. In this approach, the pre-trained model is fine-tuned on the dataset of diseased and healthy plants to adapt it to the specific task of disease detection and classification.
8. The final model can be deployed as a web application using a web framework such as Flask or Django. The user can upload an image of the plant and get the prediction of its health status along with the recommended treatment.
9. To make the system more user-friendly, a graphical user interface can be created using libraries such as PyQt or Tkinter.
10. The system can be further improved by incorporating feedback from users to improve the accuracy of the model and to update the database of diseases and treatments.

**3.4 Schema Design**

**3.4.1 Introduction**

Software is added for each of the functionality, and the schema design is involved in each of it. Each Schema of the website is briefly discussed here.

**3.4.2 Schema and Tables**

… add more ss and data here …

**3.4.3 Entity Relationship Diagram**

The ER or (Entity Relational Model) is a high-level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them.

ER modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

… add ER diagram here …

**3.5 Project Architecture**

The steps involved in the whole projects are as follows:

* User can see latest agriculture news and twitter handle of national agriculture and agriculture of up government
* To ask query user must have to login
* It will show error if any wrong credentials is entered
* To see plant disease detection, classification and treatment no login is required.

**3.6 Technologies Used**

**3.6.1 Introduction**

Various front-end and back-end technologies are available in this era of digitalization. The technologies used in this project are discussed briefly in the following sections.

**3.6.2 Front-End Technologies**

React:

* React is a popular JavaScript library for building user interfaces.
* It allows for building complex UIs using reusable components.
* React uses a virtual DOM to efficiently update the UI based on changes in state.
* React can be used to create single-page applications, mobile apps, and desktop apps.
* React is widely used in industry and has a large and active community.

Python - Streamlit:

* Streamlit is a Python library for creating interactive web applications.
* It allows for building data-driven apps with just a few lines of Python code.
* Streamlit comes with built-in widgets for visualizations, user inputs, and outputs.
* Streamlit is ideal for prototyping, sharing, and deploying data science projects.
* Streamlit can be integrated with other Python libraries like TensorFlow, PyTorch, and Scikit-learn.

Bootstrap:

* Open-source CSS framework for building responsive and mobile-first web pages
* Provides pre-built UI components and styles for consistent design and layout
* Supports various customization options, including themes and colors
* Used by many popular websites and web applications, including Twitter, LinkedIn, and Netflix

**3.6.3 Back-End Technologies**

MERN:

* Full-stack JavaScript framework for building web applications
* Includes MongoDB, Express.js, React, and Node.js
* Provides a robust and scalable solution for building complex web applications
* Supports real-time updates using WebSockets and other technologies

MERN is a web development stack consisting of four technologies - MongoDB, ExpressJS, ReactJS, and Node.js. Here are some more details about each component:

1. MongoDB: MongoDB is a NoSQL document-oriented database that stores data in a flexible, JSON-like format. It is highly scalable and provides high performance.
2. ExpressJS: ExpressJS is a lightweight and flexible Node.js web application framework that provides a set of features for web and mobile applications.
3. ReactJS: ReactJS is a JavaScript library for building user interfaces. It allows developers to build complex UIs using a component-based architecture.
4. Node.js: Node.js is a server-side JavaScript runtime that allows developers to build scalable, high-performance web applications.

1. Pankaj Jalote, (2003), ’An Integrated Approach towards Software Engineering’, Narosa Publishing House. [↑](#footnote-ref-1)