

**University Institute of Information Technology,**

**PMAS-Arid Agriculture University,**

**Rawalpindi Pakistan**

**­­­­­­­**

**Smart Testing of Waterborne Pathogens­­**

***By***

**Malik Abdullah Naazar 20-ARID-872**

**Hakim Nabi Sulehria 20-ARID-860**

***Supervisor*Dr. Muhammed Habib**

***Bachelor of Science in Software Engineering***

***(2020-2024)***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others**.

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software documentation and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Malik Abdullah Naazar Hakim Nabi Sulehria

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (SE) “Smart Testing of Waterborne Pathogens” was developed by “Malik Abdullah Naazar**, 20-Arid-872”** and “Hakim Nabi Sulehria**, 20-Arid-860”** under the supervision of “Dr. Muhammad Habib.” and that in their opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Software Engineering.

Dr. Muhammad Habib

**Supervisor**

---------------------------------------

**( )**

**External Examiner**

**Prof. Dr. Yasir Hafeez**

---------------------------------------

**Administrator UIIT**

**Executive Summary**

This report provides an overview of our University Final Year Project, which focuses on developing an advanced Water Pathogens Detection System using Artificial Intelligence. Our primary goal is to address the urgent need for rapid and precise identification of waterborne pathogens, merging cutting-edge AI techniques with microbiology. This initiative is poised to transform traditional pathogen detection methods, thereby enhancing public health and improving water quality management.

Access to clean water is a fundamental human necessity, with waterborne pathogens posing significant health risks. Current detection methods are often slow and labor-intensive, limiting their effectiveness in real-time monitoring, particularly in remote or resource-constrained regions. Our project aims to overcome these limitations by creating a system that leverages AI for the swift and accurate detection of pathogens.

The project employs the YOLOv8 deep learning model to identify harmful bacteria such as E. coli and yeast with an accuracy of 84.56%. This model is integrated into a user-friendly Flutter mobile application, allowing users to upload images of water samples and receive instant feedback on water safety. This solution significantly reduces testing time and makes water contamination detection accessible to individuals with limited resources.

The project's success demonstrates the powerful potential of combining AI with mobile technology to address real-world challenges. Looking ahead, future development will focus on creating a portable hardware device, such as a mini microscope, that can capture high-resolution images of water bacteria in real time. This device will seamlessly integrate with the existing mobile application, enabling on-the-spot testing without the need for external equipment. Further efforts will be directed towards enhancing the model's accuracy, expanding the range of detectable pathogens, and incorporating advanced features to improve user experience and system reliability.

In conclusion, our project not only offers a timely response to water safety concerns but also showcases the transformative impact of AI in public health initiatives. The ongoing development and future enhancements of this system promise to further revolutionize water pathogen detection and ensure safer water for all.

**Acknowledgement**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Dr. Muhammad Habib” for personal supervision, advice, valuable guidance and completion of this project. We are deeply indebted to him for encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Malik Abdullah Naazar Hakim Nabi Sulehria

**Abbreviations**

|  |  |
| --- | --- |
| **SRS** | Software Requirement Specification |
| **PC** | Personal Computer |
| **CNNs** | Convolutional Neural Networks |
| **DL** | Deep Learning |
| **FR** | Functional Requirements |
| **NFR** | Non-Functional Requirements |
| **TC** | Test Case |
| **UC** | Use Case |
| **ML** | Machine Learning |
| **AI** | Artificial Intelligence |

**Table of Contents**

**Introduction 2**

[1.1 Brief](#_Toc268523777) 2

[1.2 Relevance to Course Modules](#_Toc268523779) 2

[1.3 Project Background](#_Toc268523780) 3

[1.4 Literature Review](#_Toc268523780) 3

[1.5 Methodology and Software Life Cycle](#_Toc268523782) 4

**Problem Definition** 5

[2.1 Purpose](#_Toc268523795) 5

[2.2 Product Functions](#_Toc268523796) 5

[2.3 Proposed Architecture](#_Toc268523804) 6

[2.4 Project Deliverables](#_Toc268523806) 6

[2.5 Operating Environment](#_Toc268523807) 7

[2.6 Assumptions and Dependencies](#_Toc268523808) 7

**Requirement Analysis** 8

[3.1 Functional Requirments](#_Toc268523823) 13

[3.2 Non – Functional Requirments](#_Toc268523825) 14

[3.2.1 Usability](#_Toc268523787) 15

[3.2.2 Reliability](#_Toc268523787) 15

[3.2.3 Performance](#_Toc268523787) 15

[3.2.4 Supportability](#_Toc268523787) 15

[3.2.5 Design Constraints](#_Toc268523787) 15

[3.2.6 Licensing Requirements](#_Toc268523787) 15

[3.3 Use case Model](#_Toc268523823) 13

[3.3.1 Use Case Diagarm](#_Toc268523787) 13

[3.3.2 Actors Discription](#_Toc268523787) 13

[3.3.3 Use Case Discription](#_Toc268523787) 14

**The Design** 16

[4.1 UML Structural Diagrams](#_Toc268523830) 16

[4.1.1 Component Diagram](#_Toc268523787) 18

[4.1.2 System Component Diagram](#_Toc268523787) 19

[4.1.3 Package Diagram](#_Toc268523787) 19

[4.1.4 Deployment Diagram](#_Toc268523787) 20

[4.2 UML Behavioral Diagrams](#_Toc268523830) 21

[4.2.1 Activity Diagrams](#_Toc268523787) 21

[4.2.2 State Machine Diagrams](#_Toc268523787) 22

[4.3 UML Interaction Diagrams](#_Toc268523830) 23

[4.3.1 Sequence Diagrams](#_Toc268523787) 23

[4.4 Node Structure](#_Toc268523830) 23

[4.5 Communication Design Protocol](#_Toc268523830) 24

**Implementation** 24

[5.1 Communication Protocol Implementation](#_Toc268523830) 24

**Testing and Evaluation** 35

[6.1 Verification](#_Toc268523830) 35

[6.1.1 Functional Testing](#_Toc268523787) 35

[6.1.2 Static Testing](#_Toc268523787) 35

[6.2 Validation](#_Toc268523830) 36

[6.3 Usability Testing](#_Toc268523830) 38

[6.4 Unit Testing](#_Toc268523830) 38

* 1. [Integration Testing](#_Toc268523830) 38

6.6 [System Testing](#_Toc268523830) 38

6.7 [Model Results](#_Toc268523830) 38

6.12 [Test Case Tables](#_Toc268523830) 39

**GUI** 26

**Future Work** 40

**References** 50

**List of Figures**

Fig 1.1 Methodlogy and Software Lifecycle 4

Fig 2.1 Sample Picture 5

Fig 3.1 Use Case Diagram 8

Fig 4.1 System Architecture Picture 16

Fig 4.2 Class Diagram 17

Fig 4.3 Component Diagram 18

Fig 4.4 System Component 19

Fig 4.5 Package Diagram 19

Fig 4.6 Deployment Diagram 20

Fig 4.7 Activity Diagram 21

Fig 4.8 State Machine Diagram 22

Fig 4.9 Sequence Diagram 23

Fig 4.10 Node Structure 23

Fig 4.11 Confusion Matrix 39

Fig 4.12 Confusion Matrix Normalized 39

Fig 4.13 Cofidence Curve 40

Fig 4.14 Precision Confidence Curve 40

Fig 4.15 Preciosion Recall Curve 41

Fig 4.16 Recall Confusion Curve 42

Fig 4.16 Recall Confidence Curve 42

Fig 4.16 Results 43

**List of Tables**

Table 3.1 Sign Up Use Case 8

Table 3.2 Login Use Case 8

Table 3.3 Upload Image Use Case 9

Table 3.4 Detect Waterborne Pathogen Use Case 9

Table 3.5 Upload a Blog Use Case 9

Table 3.6 Read a Blog Use Case 9

Table 3.7 Like a Blog Use Case 10

Table 3.8 Extended Use Case for Upload Image 10

Table 3.9 Extended Use Case for Pathogens Search 11

Table 3.10 Functional Requirements 12

Table 3.11 Non Functional Requirements 13

# Chapter 1: Introduction

This chapter offers a comprehensive overview of our University Final Year Project, which is dedicated to the development of a state-of-the-art Water Pathogens Detection System empowered by Artificial Intelligence. With an unwavering commitment to addressing the pressing need for rapid and precise waterborne pathogen identification, our project merges advanced AI techniques with the field of microbiology. By doing so, we aim to revolutionize the conventional methods of pathogen detection, thus enhancing public health efforts and improving water quality management. Furthermore, the chapter discusses the relevance of this project to course modules related to AI and technology integration, elucidates the critical background that underscores the necessity of efficient pathogen detection in water sources, and delves into the insights gained from an extensive Literature Review. While a detailed analysis of the literature review findings and the methodology for this project is provided in subsequent sections, this chapter serves as a comprehensive introduction to our endeavor.

**Brief**

|  |  |  |
| --- | --- | --- |
| **Tools**  **And**  **Technologies** | **Tools** | **Version** |
| MS Visual Studio Code | 2023 |
| Jupyter Notebook | 2023 |
| MS Word | 2019 |
| MS Power Point | 2019 |
| Google Colab |  |
| **Technology** | **Version** |
| Python |  |
| Flutter | 2023 |
| Yolo | V8 |
| TensorFlow | 2021 |
| Firebase |  |

# Relevance to Course Modules

In the following table we have listed down number of subjects which helped us in different modules of our project.

##### Table 1.2 Relevant Course Modules

|  |  |
| --- | --- |
| **SUBJECTS** | **RELEVENT COURSE MODULES** |
| MAD | Mobile Application (Dart) |
| Software Engineering | UML Diagrams, Software Methodologies |
| Database | Firebase |
| AI | Machine learning, Image Detection |
| OOAD | Class Diagram, Use cases |

# Project Background

Access to clean water is a fundamental requirement for human health, and waterborne pathogens pose a significant threat to this vital resource. Contaminated water can lead to the spread of diseases, affecting both human and environmental well-being. Traditional methods often involve time-consuming laboratory procedures, which may delay the detection of pathogens. Moreover, these methods might not be suitable for real-time monitoring, especially in remote or resource-constrained areas.

**1.2.1 Literature Review**

The literature review for the project on "Smart Testing of Waterborne Pathogens" encompasses a thorough exploration of contemporary water quality monitoring methodologies, emphasizing the integration of artificial intelligence (AI) for image-based pathogen detection. Traditional methods like microbial culturing [[1]](#_References) and PCR are evaluated alongside their limitations, emphasizing the challenges of time delays and limited accessibility. The review focuses on the transformative shift towards image-based analysis facilitated by AI, particularly through the use of small microscopes for image capturing. Advancements in sensor technologies are discussed, emphasizing AI-driven image analysis for automating and expediting pathogen detection. The integration of microfluidics is explored for its potential in portable, AI-enhanced image analysis. Literature is surveyed for studies leveraging the Internet of Things (IoT) to connect image-capturing devices for real-time data collection. The role of AI in data analytics is scrutinized, emphasizing machine learning algorithms for pattern recognition in water quality images. Successful implementations and case studies highlight the efficacy of AI-powered image analysis in improving pathogen detection and response times. Consideration is given to the alignment of AI-based smart testing systems with regulatory frameworks and standards. The review concludes with a critical analysis, identifying gaps in the literature that the proposed project, centered on AI image analysis for waterborne pathogen detection using small microscopes, aims to address, ultimately advancing the field in the context of emerging technologies.

**Analysis from Literature Review**

The analysis from the literature review unveils a transformative landscape in water quality monitoring, particularly within the context of the proposed project on "Smart Testing of Waterborne Pathogens" using AI-based image analysis with small microscopes. Traditional methods like microbial culturing and PCR have long been cornerstones in pathogen detection, but their limitations, including time delays and accessibility challenges, underscore the need for innovation. The integration of AI for image-based analysis, especially with the use of small microscopes, emerges as a promising solution to overcome these challenges. This synthesis of AI and microscopy offers the potential for real-time, automated pathogen detection, revolutionizing conventional practices.

# Methodology and Software Lifecycle for This Project

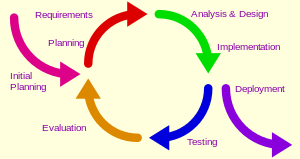
The chosen methodology for this project is an amalgamation of both structural and Object-Oriented methodologies. [[2]](#_References)

1.3.1 Rationale behind Selected Methodology

This hybrid approach is motivated by the project's complex nature, necessitating a systematic structuring of components (structural) while embracing the flexibility and adaptability offered by Object-Oriented principles. By combining these methodologies, we aim to achieve a robust and scalable system architecture that aligns with the intricacies of waterborne pathogen detection.

* + - 1. *Rationale behind Selected Methodology*

Additionally, the Software Development Life Cycle (SDLC) model selected for this project is the Iterative and Incremental model. This model aligns with the iterative nature of AI model training and refinement, allowing for successive cycles of development, testing, and improvement. The incremental aspect ensures that new functionalities are added in stages, promoting continuous enhancements based on feedback and evolving requirements. This iterative and incremental approach is well-suited for the dynamic landscape of AI-based projects, fostering agility and responsiveness throughout the development lifecycle.



**Figure 1.1: Design Methodology**

**Chapter 2: Problem Definition**

# Problem Statement

The existing methods for water pathogen detection suffer from limitations such as time-intensive manual processes and the potential for false negatives or positives. This project addresses the urgent need for a rapid and precise system that can automatically identify waterborne pathogens using AI techniques, ensuring timely response to potential health hazards and enhancing overall water safety as shown in the diagram below:[[4](#_References)]

****

**Figure 2: Current Water Pathogen Detection System [**[**4**](#_References)**]**

# Deliverables and Development Requirements

* Our project aims to deliver a robust Water Pathogens Detection System, integrating cutting-edge Artificial Intelligence techniques with microbiology. The key deliverables include:
* AI-Enabled Water Pathogens Detection Algorithm: Develop an advanced algorithm capable of accurately identifying waterborne pathogens through the analysis of water samples. This algorithm will leverage machine learning and deep learning techniques to enhance precision and efficiency.
* User-Friendly Interface: Create an intuitive and user-friendly interface for easy interaction with the system. This interface will allow users, including water quality management professionals and microbiologists, to input samples and interpret the results generated by the AI system.
* Integration with Laboratory Equipment: Ensure seamless integration with existing laboratory equipment for sample processing and data input. This involves compatibility with standard microbiology tools and protocols to facilitate the adoption of the system within research and testing environments.
* Real-Time Detection Capability: Implement real-time pathogen detection to provide timely results. This feature is crucial for responding promptly to potential water contamination events, thereby aiding in the swift implementation of necessary interventions.
* Scalability: Design the system to be scalable, accommodating varying sample sizes and types. This ensures adaptability to different water sources and facilitates broader application in diverse settings.
* Documentation and Training Materials: Develop comprehensive documentation and training materials to assist users in understanding the system's functionality, operation, and maintenance. This includes user manuals, troubleshooting guides, and training modules.

# Current System

Currently, waterborne pathogen detection relies on traditional methods that often involve time-consuming culturing processes and manual microscopic examination. Figure 2.1 provides a visual representation of the typical workflow in the existing system. The limitations of the current system include prolonged detection times, dependency on skilled personnel, and potential human errors. Our project aims to overcome these challenges by introducing a state-of-the-art AI-powered system, revolutionizing the way water pathogens are identified and ensuring a faster and more accurate response to potential public health threats. The subsequent sections of this report will delve into the literature review, methodology, and in-depth analysis of the proposed Water Pathogens Detection System.



**Figure 2.1: Current Water Pathogen Detection System [**[**4**](#_References)**]**

**Figure Description**

* In the figure above the traditional way of testing water samples is visible.
* Samples of water are brought and observed under the supervision of an expert.
* The testing process consists of expensive hardware devices such as microscopes.
* Different chemicals are used and reactions are studied to discover the results.
* After this long process the results are handed over to the user.
* With our proposed solution this complete process can be avoided and user will be able to get results rapidly.

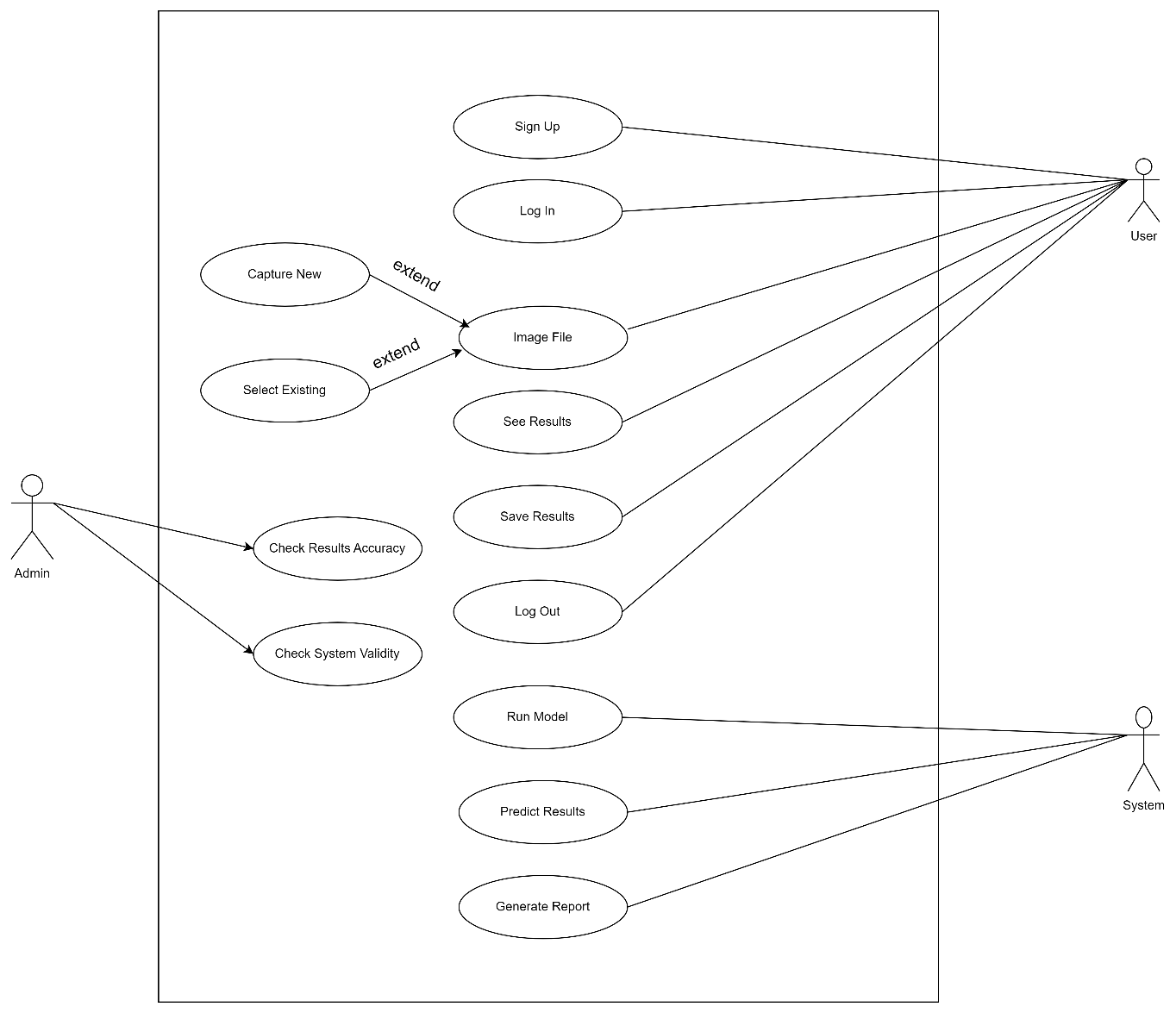
.

# Chapter 3: Requirement Analysis

This chapter is all about the software requirement specifications of our project. In the requirement analysis phases, we gather requirements from our stakeholders and write down those requirements in the form of functional and non-functional requirements to get more understanding into software functionality. Use Case diagram is created and description actors of use cases from this diagram are provided.

# Use Cases

### **3.3.1 Use Case Diagram**



**Figure 3.1: Use Case**

### **3.3.2 Actors Description**

The use case of this system has three actors which are:

**User:**

* This is the primary user of the use case that will directly interact with the system.
* The functionality provided to the user is that, they will be able to sign up or login to the system, upload image, see any pathogens are detected or not.

**Admin:**

* This is also primary user of the use case, that will be able to check the accuracy of the result and validate if the system is working as expected or not.

**Detection System:**

* This is the secondary actor in the use case, which is used by the system to obtain a result or required information which is detecting the waterborne pathogens but it does not interact with the system on its own.
* When data is provided by the user then it starts to function.

### **3.3.3 Brief Use Cases**

**Table 3.1: Signup Use Case**

|  |  |
| --- | --- |
| **Use Case ID**: | UC-1 |
| **Name:** | Sign Up |
| **Actor Type:** | Primary (User) |
| **Description:** | The user sign up in to the system by making an account or adding username and password respectively. |

**Table 3.2: Login Use Case**

|  |  |
| --- | --- |
| **Use Case ID**: | UC-2 |
| **Name:** | Log In |
| **Actor Type:** | Primary (User) |
| **Description:** | The user login to the system by making an account or giving username and password respectively. |

**Table 3.3: Upload image Use Case**

|  |  |
| --- | --- |
| **Use Case ID:** | UC-3 |
| **Name:** | Upload Image |
| **Actor Type:** | Primary (User) |
| **Description:** | The user uploads an image of area to be detected for Waterborne Pathogens on the app in order to detect the Pathogens. |

**Table 3.4: Detect Waterborne Pathogen Use Case**

|  |  |
| --- | --- |
| **Use Case ID:** | UC-5 |
| **Name:** | Detect Waterborne Pathogens |
| **Actor Type:** | Secondary (System) |
| **Description:** | The system processes the image provided by the users and detect pathogens using large Waterborne Pathogens dataset. |

**Table 3.5: Upload blog Use Case**

|  |  |
| --- | --- |
| **Use Case ID:** | UC-6 |
| **Name:** | Upload BLOG |
| **Actor Type:** | System User |
| **Description:** | The Super User (Admin) can Upload BLOG. |

**Table 3.6: View blog Use Case**

|  |  |
| --- | --- |
| **Use Case ID:** | UC-7 |
| **Name:** | View BLOG |
| **Actor Type:** | User & System User |
| **Description:** | The user upload an image from device gallery to be detected for dengue larvae on the app in order to detect the dengue larvae. |

**Table 3.7: Like a blog Use Case**

|  |  |
| --- | --- |
| **Use Case ID:** | UC-8 |
| **Name:** | Like BLOG |
| **Actor Type:** | User & System User |
| **Description:** | The user like the BLOG. |

### **3.3.4 Extended Use Cases**

**Table3.8: Extended Use Case for upload image**

|  |  |
| --- | --- |
| **Use Case ID:** | UC-3 |
| **Use Case Name:** | Upload Image for Waterborne Pathogens Testing |
| **Actors:** | **Primary:** User |
| **Description:** | The user uploads an image of the target that is used to detect the pathogens. |
| **Trigger:** | User wants to know the presence of waterborne pathogens and identify it. |
| **Preconditions:** | 1. User must have a smart phone or camera to click the image of area of water. 2. Internet connection is available on the user’s device before uploading image. 3. User has signed up into the app(optional). |
| **Post conditions:** | 1. The image has been uploaded on the app. 2. The detected Waterborne Pathogens are shown to the user. |
| **Normal Flow:** | 1. User takes the image of water in which he/she desires to detect the Waterborne Pathogens. 2. User login into the app(optional). 3. The water image is uploaded by the user on the app. 4. System receives the image and send it to deep learning model for further processing. 5. User waits for the system to provide results for the input image. 6. System detect the pathogens. 7. The output including detected pathogens is shown to the user. |
| **Alternative Flows:**  **Alternative Flow 1** | 2a. In step 2 of the normal flow, if the image uploaded by user is of the water where pathogens can be found.   1. System will tell the user that the input image is invalid if user uploads any other file. |
| **Exceptions:** | 1. The user has a slow or no internet connection. 2. System tells the user that uploading failed due to connection error. |
| **Includes:** | None |
| **Special Requirements:** | Smart Microscope must be attached with the mobile camera before taking or capturing the image. |
| **Assumptions:** | 1. The user has a camera or other device to click and upload image. |
| **Notes and Issues:** | Make sure to upload a clear image of the water sample. |

**Table 3.9: Extended Use Case for Waterborne Pathogens search**

|  |  |
| --- | --- |
| **Use Case ID:** | UC-4 |
| **Use Case Name:** | Search Waterborne Pathogens |
| **Actors:** | **Primary:** User |
| **Description:** | The user search the Pathogens by uploading the image. |
| **Trigger:** | The user wants to know the pathogens by image. |
| **Preconditions:** | 1. User has signed up into the app (optional).  2. User should search the related pathogens.  3. Internet connection should be stable. |
| **Post conditions:** | 1. The related information shows to user.  2. User has signed up into the app (optional). |
| **Normal Flow:** | 1. User has signed up into the app (optional).  2. User search the waterborne pathogens. And wait for system reaction. |
| **Alternative Flows:**  **Alternative Flow 1** | In step 2 of the normal flow, if search image are not related with water then System will tell the user that the input file is invalid, Select/Upload the correct one. |
| **Exceptions:** | 1. The user has a slow or no internet connection.  2. System tells the user that searched word does not match with anything. |

**3.1 Functional Requirements**

The success of the Water Pathogens Detection System relies on well-defined functional requirements that govern its capabilities and features. The following outlines the key functional requirements for our system:

* **Data Acquisition and Preprocessing:**

The system will be capable of acquiring water samples for analysis.

Raw data collected must undergo preprocessing to extract relevant features.

* **Pathogen Identification:**

Implement AI algorithms to accurately identify waterborne pathogens.

* **Real-time Monitoring:**

Enable real-time monitoring of water samples for timely detection of pathogens.

Provide instant alerts in case of pathogen detection.

* **User Interface:**

We will develop an intuitive user interface for users to interact with the system.

Include features for data visualization, result interpretation, and system control.

**Table 3.10: Functional Requirements**

|  |  |
| --- | --- |
| **Requirement ID** | **Description** |
| **FR-01** | The system should be able to detect the waterborne pathogens when the user inputs an image. |
| **FR-02** | The users should be able to register into the system by signing up. |
| **FR-03** | The users should be able to see the results after detection. |
| **FR-04** | The users should not be able to upload image even without login. |

**3.2 Non-Functional Requirements**

Non-functional requirements are essential for ensuring the overall effectiveness and performance of the Water Pathogens Detection System. These requirements encompass various aspects such as usability, reliability, performance, supportability, design constraints, and licensing requirements.

**Table 3.11: Non-Functional Requirement**

|  |  |
| --- | --- |
| **Requirement ID** | **Description** |
| **NFR-01** | The system should be easily understandable by the new user. |
| **NFR-02** | The image detection should have the accuracy of 99%. |
| **NFR-03** | The app should have a response rate of 0.5 to 2 second. |
| **NFR-04** | The app should not reload in order to improve responsiveness. |
| **NFR-05** | The detected pathogens information should be correct and reliable. |

**3.2.1 Usability**

Our system will be designed to ensure the best user experience. All the age groups will be considered while opting for the interface.

* **Intuitive Interface:**

The system will feature an intuitive and user-friendly interface. Users, including non-technical personnel, will also be able to navigate and utilize the system effectively.

**3.2.2 Reliability**

Reliability ensures the consistent and accurate performance of the system. The AI model is trained with large amounts of data to ensure the accuracy of results and make our system reliable.

* **Accuracy:**

The AI algorithms must consistently provide accurate pathogen identification results. The system should have a low rate of false positives and false negatives. In this case more data for training brings more accuracy.

**3.2.3 Performance**

With reliability and accuracy achieved the next step is efficiency. The system response will be rapid along with alerts. It is ensured that accuracy is not compromised to achieve efficiency.

* **Processing Time:**

The system should analyze water samples promptly to meet real-time monitoring requirements. User shall see the results of uploaded data in moments.

**3.2.4 Supportability**

It will be ensured that the application is supportable on all mobile devices. We will be developing using Flutter which makes it possible to run on cross platforms.

**3.2.4 Maintainability**

* **Maintenance Procedures**

Regular data analysis from the application will be done to ensure that if there are any flaws , they are fixed on time and the best way is user reviews.

# Chapter 4: Design and Architecture

# System Architecture

Database. Users interact with the User Interface to upload Dengue larvae images, initiating a process managed by the Web Server. The uploaded images undergo preprocessing before being processed by the YOLOv8 Object Detection Component, which employs advanced machine learning techniques for the identification and classification of Dengue mosquito larvae. The Detection API facilitates seamless communication between the Web Server and the YOLOv8 component. Results, including detected Dengue larvae and associated information, are displayed to users through the User Interface. The Database stores relevant data for future reference or analysis, contributing to the effective monitoring and management of mosquito breeding sites. This architecture aligns with the project's goal of employing smart technology to create a rapid and accurate system, akin to providing a superpower in the fight against Dengue, ultimately enhancing public health outcomes.

A diagram of a software application

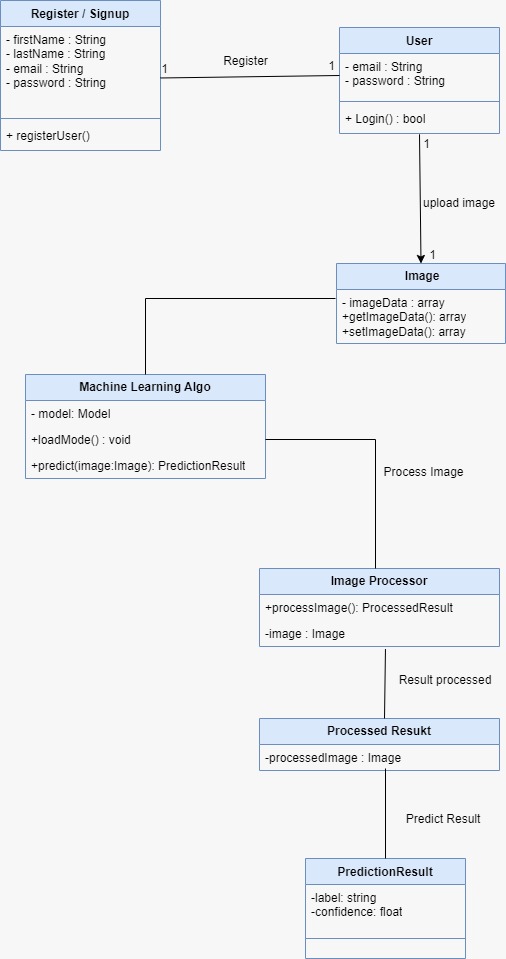
Description automatically generated with medium confidence

**Figure 4.1: System Architecture**

# UML Structural Diagrams

Structural diagrams describe the static behavior of the system. These diagrams are very useful in the design and analysis phase of software development lifecycle, for software design [[5](#_References)].

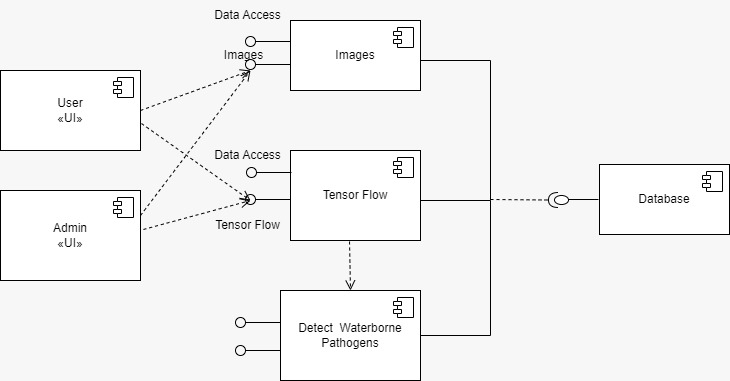
**4.2.1 Class Diagram**



**Figure 4.2: Class Diagram**

### **4.2.2 Component Diagram**

Component diagrams shows the logical and physical units or modules in our system and how they relate with each other. It is used to model the static implementation of the system.



**Figure 4.3: Component diagram**

### **4.2.3 System Component Diagram**

System Component diagram is structural diagram shows the logical and physical units or modules in our system and how they relate with each other through system.

A diagram of a computer component

Description automatically generated

**Figure 4.4: System Component diagram**

### **4.2.4 Package Diagram**

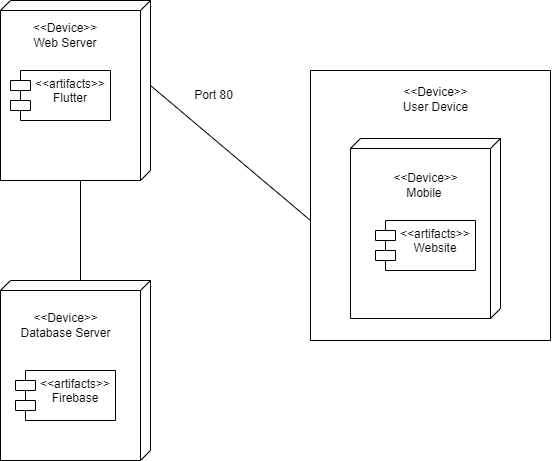
A package diagram is structural diagram which is used to reflect the organization of packages and their elements in our system.

A diagram of a computer

Description automatically generated

**Figure 4.5: Package diagram**

### **4.2.5 Deployment Diagram**



**Figure 4.6: Deployment diagram**

## **4.3 UML Behavioral Diagrams**

Behavioral diagrams are used to show the dynamic behavior of the system, and how the system will work.

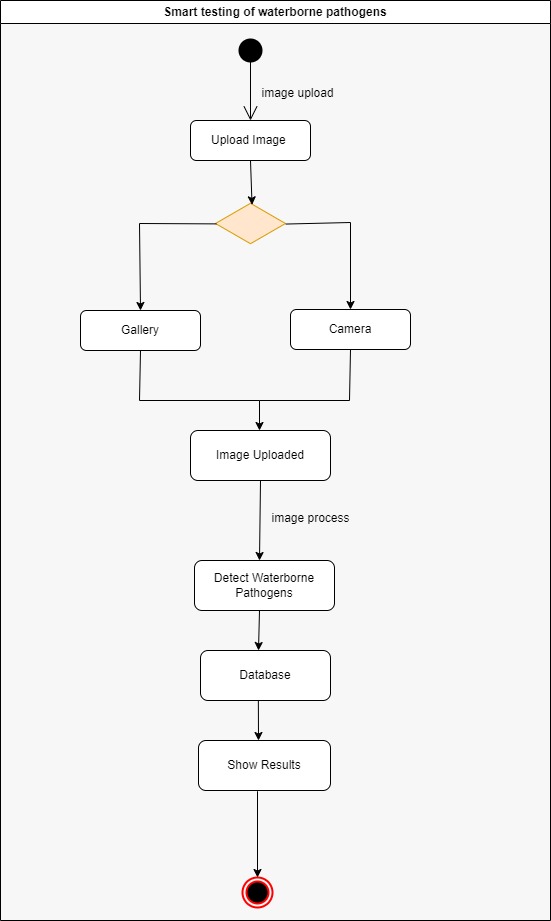
**4.3.1 Activity Diagram**

A diagram of a software company

Description automatically generated

**Figure 4.7: Activity diagram**

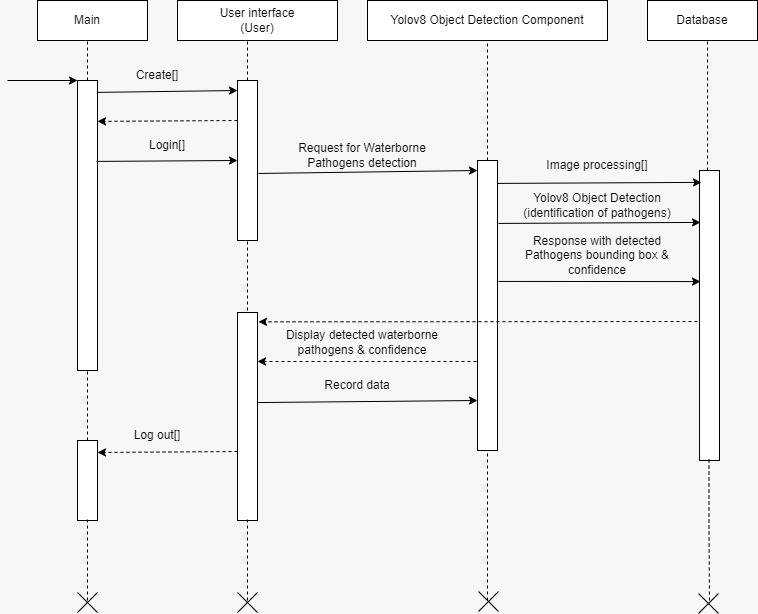
### **4.3.2 State Machine Diagram**



**Figure 4.8: State Machine diagram**

**4.3 UML Interaction Diagrams**

### **4.3.1 Sequence Diagram**



**Figure 4.9: Sequence diagram**

* 1. **Node Structure**

A black background with white text

Description automatically generated

1. **Figure 4.10: Node Structure diagram**

**4.5 Communication Design Protocol**

A blue lines on a black background

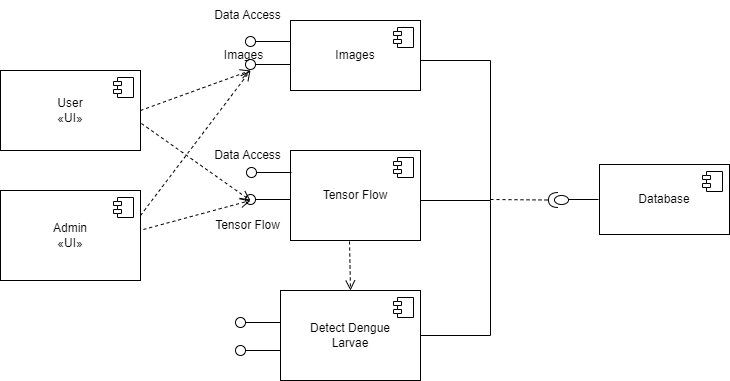
Description automatically generated

**Fig 4.11 Communication Design Protocol [**[**3**](#_References)**]**

**Chapter 5: Implementation**

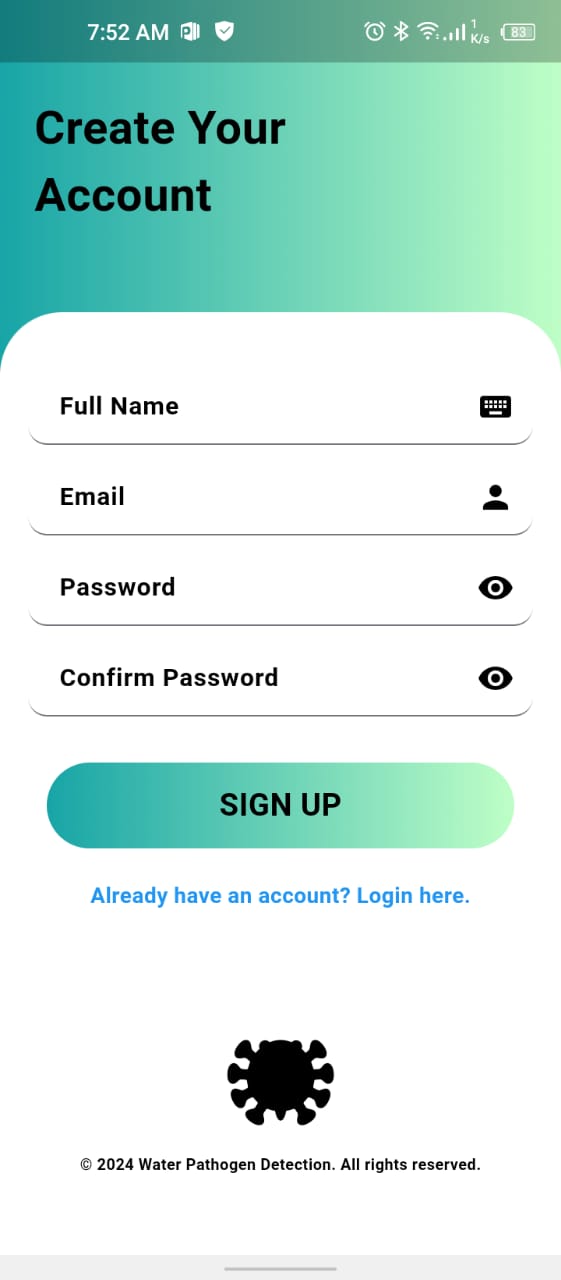
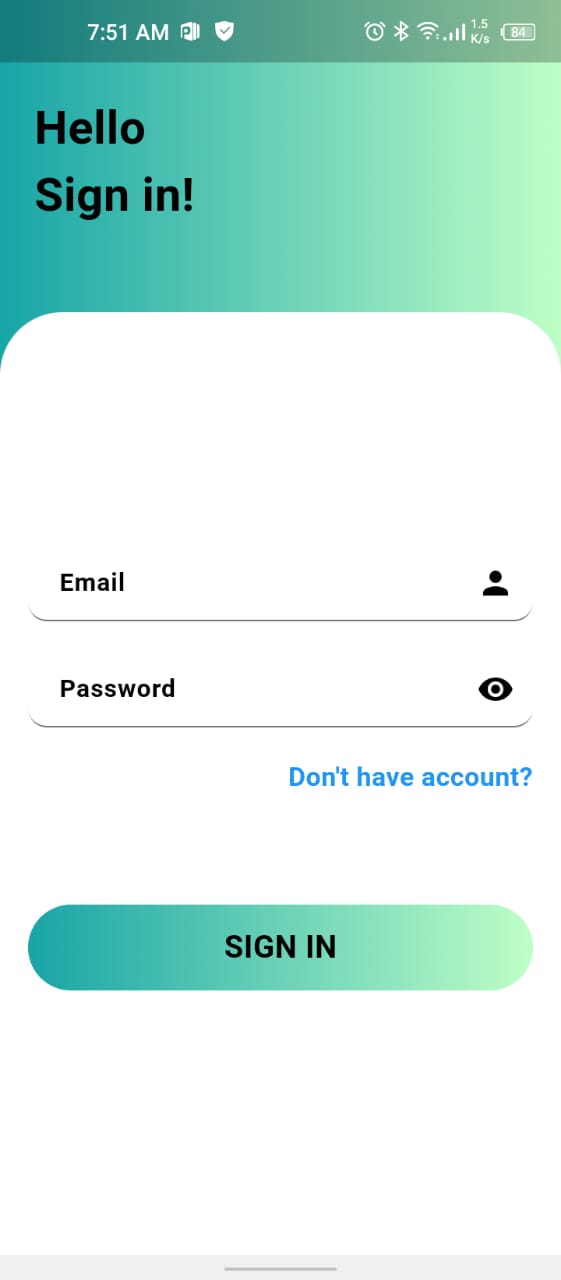
This chapter will discuss implementation details supported by UML diagrams (if applicable). You will not put your source code here. Any of the following sections may be included based on your project.

# Component Diagram



# User Interfaces

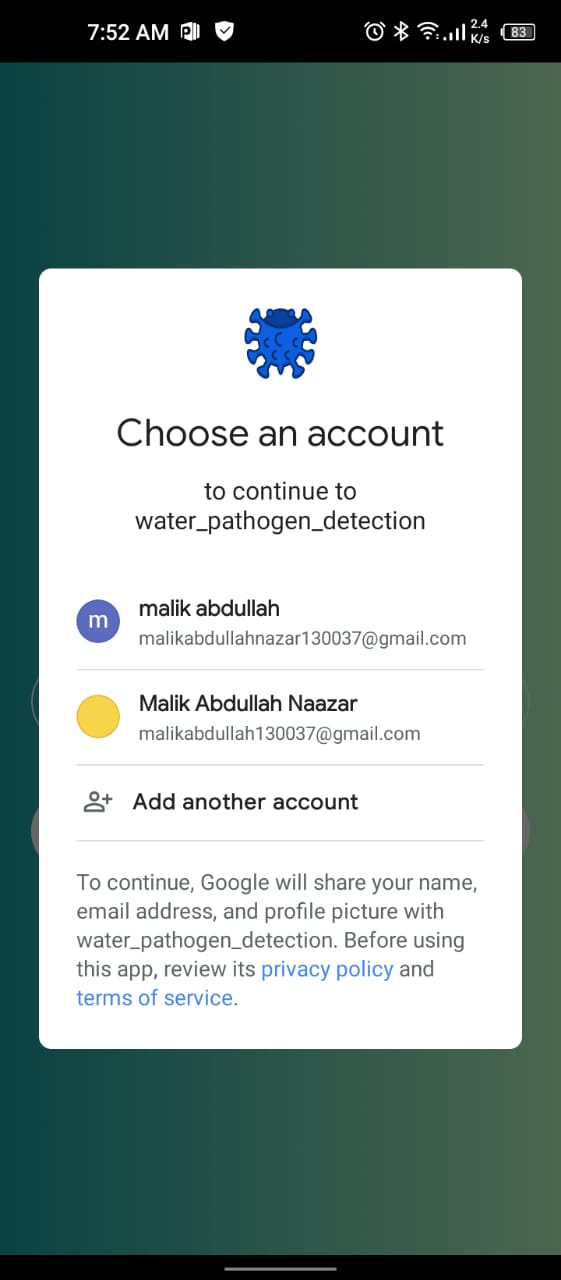
**Sign Up & Sign In**

** **

* The user can easily register by easily creating an account.
* If the user is already registered he/she can easily sign in to applcation.
* To create an account user must provide all relevant information correctly.
* User can also register by using its Google Account.

**Welcome Back & Google Sign In**

**A screenshot of a sign up

Description automatically generated **

* The user sign in screen is displayed.
* If the user is already registered he/she can easily sign in to applcation.
* To create an account user can press the sign Up button

**Home & Blogs**

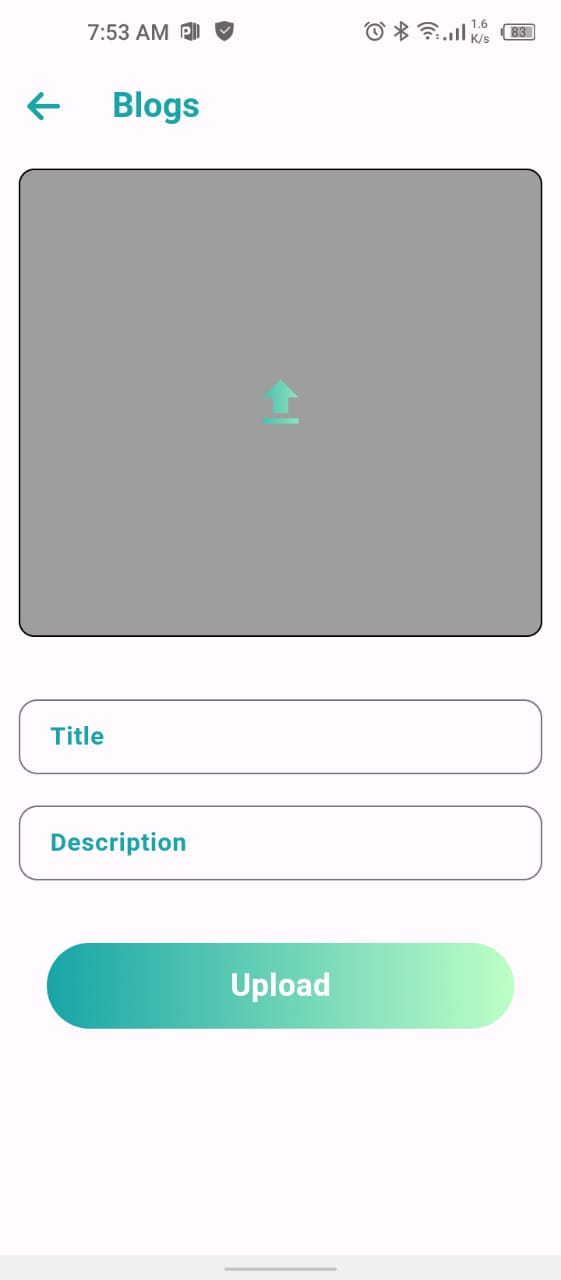
**A screenshot of a cell phone

Description automatically generated A white rectangular object with black text

Description automatically generated**

* On the left side the Home screen is displayed
* After Signing In successfully user can see home screen.
* Different Options are available for user to perform actions.
* User can also see the recent results or results history.

**Upload Blog & View Blog**

** A close up of a bug

Description automatically generated**

* The user (Admin) can upload blogs.
* The users can view the blogs uploaded on the application.
* The blogs list is updated with latest blogs at the top.
* User can read information about pathogens from here.

**My Profile & Contact Us**

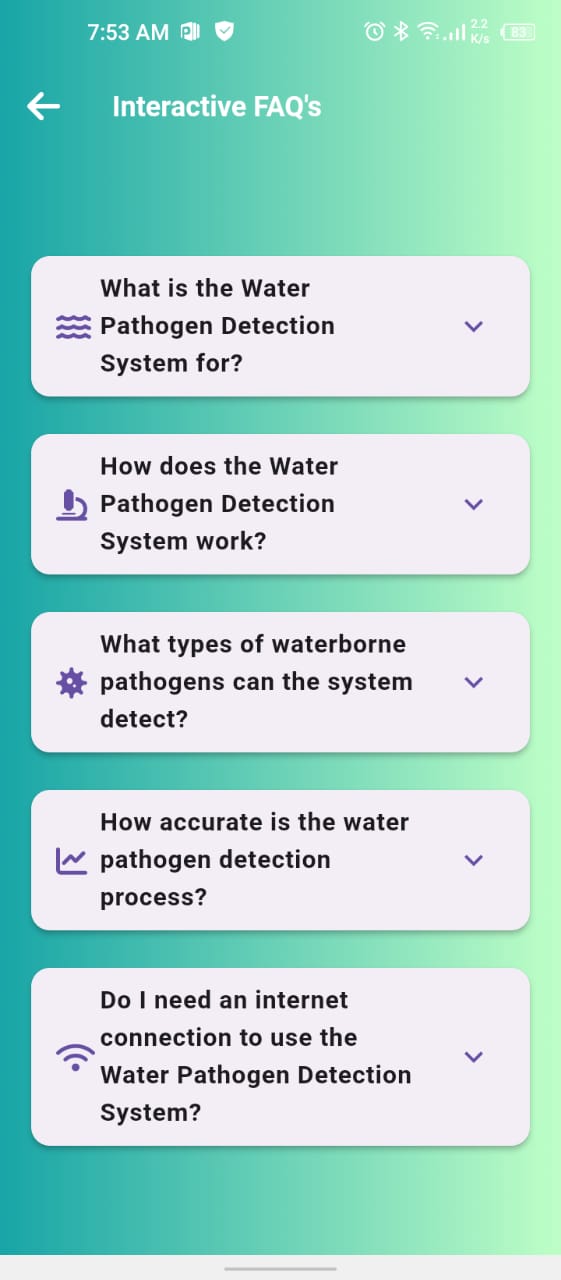
**A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated**

* The user profile and contact us screens are shown.
* User can see some important stats.
* User can also view the favourite blogs by clicking my saves.
* Through contact us page we are able to get useful feedback from our users.

**FAQ’s & Help Center**

** A screenshot of a cell phone

Description automatically generated**

* The Frequently asked questions are displayed for user facilitation.
* If user is stuck somewhere he/she can get their answers from here.
* If the application is not working correctly we can be informed through help center.
* Also critical user feedback is gathered from this section.

**Save Results & Saved History**

**A screenshot of a calendar

Description automatically generated **

* The user save results
* Also the user can view the history.
* User can also delete the results if not interested.

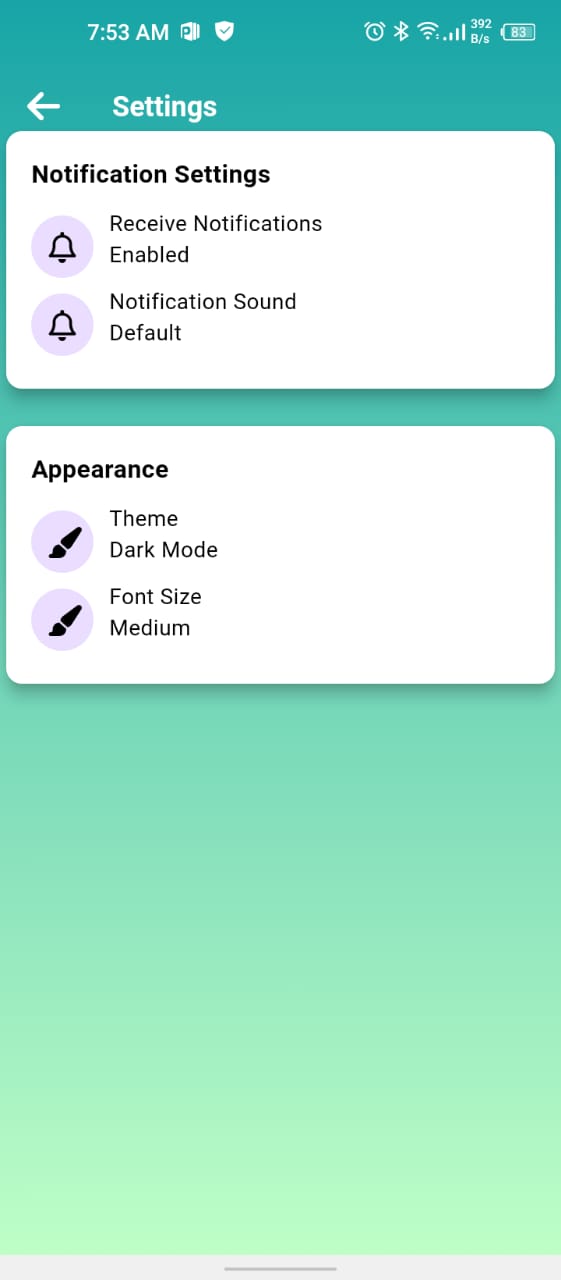
**Detect & Detection results**

** A screenshot of a phone

Description automatically generated**

* The user must select an image from gallery and click detect button
* The system takes it time for processing and after that success message in displayed.
* The results can be viewed in the Detection results tab.

**Settings**

****

* The user can explore diferent options at settings tab.
* Font size can be changed and also user can switch theme.
* Notifications can a,so be contriolled from this tab.

# Chapter 6: Testing and Evaluation

# Verification

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Use Case Name** | **Verification** | **Status** |
| UC-1 | Sign Up | This use case involves allowing users to sign up for the system. It typically includes capturing user information such as username, email, password, etc., and creating a new account. | Verified |
| UC-2 | Log In | This use case involves allowing registered users to log in to the system using their credentials. | Verified |
| UC-3 | Take Image from Camera | This use case involves capturing an image using the device's camera functionality. | Verified |
| UC-4 | Upload Image from Gallery | This use case involves allowing users to upload an image from their device's gallery or file system. | Verified |
| UC-5 | Detect Waterborne Pathogens | This use case involves using image processing techniques to detect waterborne pathogens in an uploaded image. | Verified |
| UC-6 | Upload Image for Blog | This use case involves allowing users to upload images for use in blog posts. | Verified |
| UC-7 | Upload Blog | This use case involves allowing users to upload blog posts. | Verified |
| UC-8 | View Blog | This use case involves allowing users to view blog posts. | Verified |

# Validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Use Case Name** | **Validation** | **Status** |
| UC-1 | Sign Up | The system should provide a registration form where users can enter their username, email, password, etc. Upon submission, the system should create a new account for the user. | Validated |
| UC-2 | Log In | The system should provide a login form where registered users can enter their credentials (username/email and password). Upon submission, the system should authenticate the user and grant access to the system. | Validated |
| UC-3 | Take Image from Camera | The system should provide functionality to access the device's camera and capture an image. Upon capturing, the system should store the image for further processing or use. | Validated |
| UC-4 | Upload Image from Gallery | The system should allow users to browse and select an image from their device's gallery or file system. Upon selection, the system should upload the image to the server for further processing or use. | Validated |
| UC-5 | Detect Waterborne Pathogens | The system should implement image processing algorithms to analyze uploaded images and detect the presence of waterborne pathogens. Upon detection, the system should provide relevant feedback or actions. | Validated |
| UC-6 | Upload Image for Blog | The system should provide functionality for users to upload images to be used in blog posts. Upon upload, the system should store the image and associate it with the corresponding blog post for display. | Validated |
| UC-7 | Delete Blog | The system should allow authorized users to delete blog posts. Upon deletion, the system should remove the blog post and associated content (e.g., images) from the database and display appropriate confirmation messages.  Blogs can Only be deleted by System User only. | Validated |
| UC-8 | View Blog | The system should allow authorized users to view the list of uploaded blogs. | Validated |

.

**6.3. Usability Testing**

Usability testing ensures that the mobile application is user-friendly and meets the needs of its intended users. In this phase, we conducted tests with a group of target users, including individuals from regions with scarce access to clean water. These users were asked to perform various tasks within the app, such as uploading images of water samples and interpreting the results provided by the system. Feedback was collected on the ease of navigation, clarity of information, and overall user satisfaction. Iterative improvements were made based on this feedback to enhance the user experience[[6](#_References)].

**6.4. Module / Unit Testing**

Unit testing involves testing individual components or modules of the YOLOv8 model and the mobile application to ensure they function correctly in isolation. Each function, from image upload to pathogen detection and result display, was tested for accuracy and reliability. Mock data was used to simulate various scenarios, ensuring that each module performed as expected under different conditions. This testing phase helped identify and fix bugs early in the development process, ensuring a robust and reliable system[[6](#_References)].

**6.5. Integration Testing**

Integration testing focuses on verifying that the modules and components work together seamlessly. This phase involved testing the interaction between the YOLOv8 model, the Flutter mobile application, and the backend server. The primary goal was to ensure data flows correctly from the image upload through pathogen detection to result display. Various integration scenarios were tested, including handling different image formats, network connectivity issues, and concurrent user access, to ensure smooth and reliable system performance[[6](#_References)].

**6.6. System Testing**

System testing is performed to validate the entire system's functionality and performance against the requirements. This phase involved end-to-end testing of the mobile application, including real-world scenarios where users upload images of water samples. The accuracy of pathogen detection, the speed of processing, and the clarity of the results were evaluated. Additionally, the system's behavior under typical and peak loads was assessed to ensure stability and reliability[[6](#_References)].

**6.7. Acceptance Testing**

Acceptance testing involves verifying that the system meets the specified requirements and is ready for deployment. This phase included user acceptance testing (UAT) with stakeholders, including potential users from target regions. The system was evaluated against acceptance criteria such as detection accuracy, user interface usability, and result reliability. Feedback from this phase was used to make final adjustments, ensuring the system is fully functional and ready for real-world use[[6](#_References)].

**6.8. Stress Testing**

Stress testing evaluates the system's performance under extreme conditions to ensure stability and reliability. This phase involved simulating high user traffic, large image uploads, and prolonged usage periods to identify potential performance bottlenecks and system failures. The system's response time, resource utilization, and error handling were monitored and optimized to ensure it can handle high-stress situations without compromising performance or accuracy[[6](#_References)].

**6.9. Evaluation**

The evaluation phase involves assessing the overall performance, usability, and effectiveness of the system. Metrics such as detection accuracy, user satisfaction, processing time, and system reliability were measured and analyzed. Feedback from usability, integration, system, and acceptance testing phases was compiled to provide a comprehensive evaluation. This assessment helped identify strengths, areas for improvement, and the overall readiness of the system for deployment[[6](#_References)].

**6.10. Deployment**

The deployment phase involves the process of releasing the mobile application to the intended users. This included preparing the application for distribution on relevant app stores, configuring backend servers for optimal performance, and ensuring necessary support resources are in place. A detailed deployment plan was created, including user training materials, support documentation, and a rollout strategy to ensure a smooth transition from development to real-world use[[6](#_References)].

**6.11. Maintenance**

Maintenance involves ongoing support and updates to ensure the system remains functional and up to date. This phase includes monitoring system performance, addressing user-reported issues, and releasing updates to improve functionality and security. Regular maintenance tasks such as database backups, server monitoring, and application updates are planned to ensure long-term reliability. User feedback continues to play a crucial role in identifying areas for improvement and guiding future development efforts[[6](#_References)].

**6.12. Test Case Tables**

**1. Login Test Case:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Pre-Conditions** | **Test Steps** | **Expected Result** |
| TC01 | Verify login with valid credentials | User has registered and has credentials | 1. Open the app  2. Navigate to the login screen.  3. Enter valid email and password  4. Click on the login button | User is successfully logged |
| TC02 | Verify login with invalid credentials | None | 1. Open the app  2. Navigate to the login screen.  3. Enter invalid email and/or password .  4. Click on the login button | Error message is displayed |
| TC03 | Verify login with empty fields | None. | 1. Open the app 2. Navigate to the login screen.  3. Leave email and password fields empty.  4. Click on the login button | Error message is displayed |
| TC04 | Verify password visibility toggle | None | 1. Open the app.  2. Navigate to the login screen.  3. Enter password.  4. Toggle the password visibility icon | Password is shown/hidden accordingly |

**2. Signup Test Case:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Pre-Conditions** | **Test Steps** | **Expected Result** |
| TC05 | Verify signup with valid details | None | 1. Open the app  2. Navigate to the signup screen.  3. Enter valid details (name, email, password).  4. Click on the signup button | User is successfully registered and logged in |
| TC06 | Verify signup with existing email | User already registered | 1. Open the app.  2. Navigate to the signup screen.  3. Enter existing email.  4. Click on the signup button | Error message is displayed |
| TC07 | Verify signup with invalid email | None. | 1. Open the app.  2. Navigate to the signup screen.  3. Enter invalid email.  4. Click on the signup button | Error message is displayed |
| TC08 | Verify signup with weak password | None | 1. Open the app.  2. Navigate to the signup screen.  3. Enter weak password  4. Click on the signup button | Error message is displayed |

**3. Upload Image Test Case:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Pre-Conditions** | **Test Steps** | **Expected Result** |
| TC09 | Verify image upload with valid image | User is logged in | 1. Open the app.  2. Navigate to the upload screen.  3. Click on "Upload Image". 4. Select valid image | Image is uploaded successfully and analysis starts |
| TC10 | Verify image upload with invalid format | User is logged in | 1. Open the app.  2. Navigate to the upload screen.  3. Click on "Upload Image". 4. Select invalid format | Error message is displayed |
| TC11 | Verify image upload with no image | User is logged in. | 1. Open the app.  2. Navigate to the upload screen.  3. Click on "Upload Image". 4. Do not select an image | Error message is displayed |
| TC12 | Verify real-time capture functionality | User is logged in | 1. Open the app.  2. Navigate to the upload screen.  3. Click on "Capture Image". 4. Take a photo | Image is captured and uploaded for analysis |

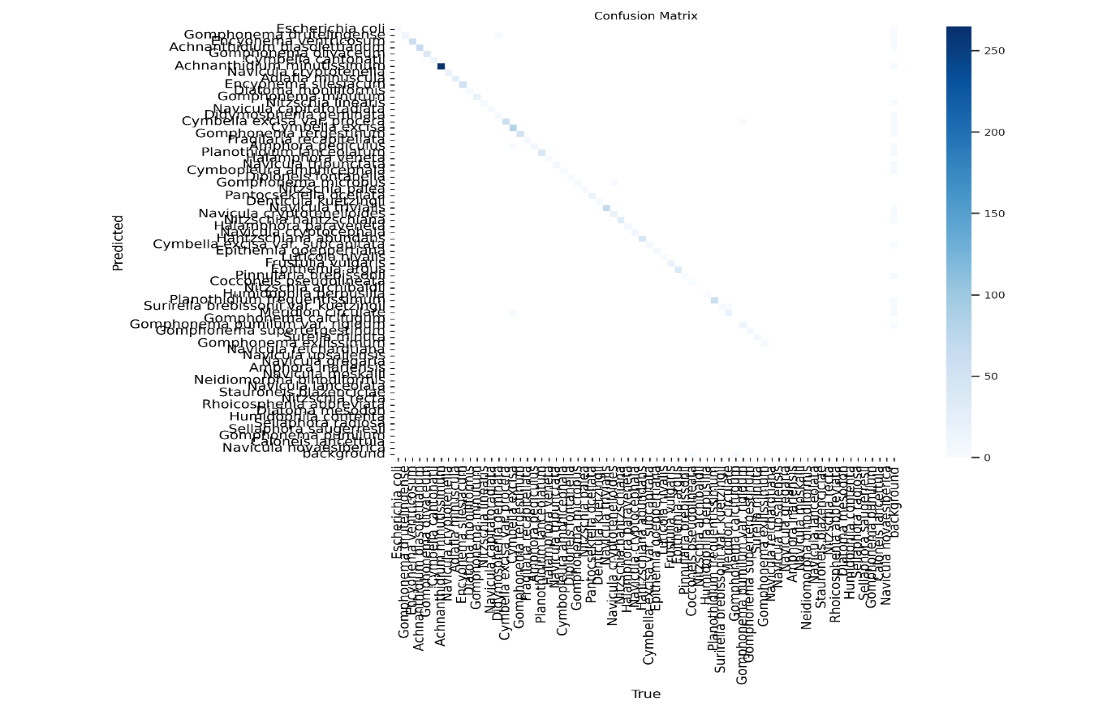
**4. View Results Test Case:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Pre-Conditions** | **Test Steps** | **Expected Result** |
| TC13 | Verify viewing past results | User is logged in, has previous results | 1. Open the app.  2. Navigate to the results history screen  3. Select a previous result entry | Detailed past result is displayed |
| TC14 | Verify results for a newly uploaded image | User is logged in, image uploaded | 1. Upload a valid image  2. Wait for processing  3. Navigate to results screen | Latest result is displayed accurately |
| TC15 | Verify results filtering functionality | User is logged in, has multiple results | 1. Open the app  2. Navigate to results history 3. Use filters (date, type of pathogen, etc.) | Results are filtered and displayed |
| TC16 | Verify results export functionality | User is logged in, has previous results | 1. Open the app.  2. Navigate to results history 3. Select result entry  4. Click on export | Results are exported successfully |

**5. User Profile Test Case:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Pre-Conditions** | **Test Steps** | **Expected Result** |
| TC17 | Verify updating profile information | User is logged in | 1. Open the app  2. Navigate to the profile screen  3. Update profile information 4. Save | Profile information is updated successfully |
| TC18 | Verify profile picture update | User is logged in | 1. Open the app  2. Navigate to the profile screen  3. Update profile picture  4. Save | Profile picture is updated successfully |
| TC19 | Verify viewing profile information | User is logged in. | 1. Open the app  2. Navigate to the profile screen | Profile information is displayed accurately |
| TC20 | Verify logout functionality | User is logged in | 1. Open the app  2. Navigate to the profile screen  3. Click on logout button | User is logged out and redirected to login screen |

**6.13. Model Results**

****

**Figure 4.11: Confusion Matrix**

**A graph of a graph

Description automatically generated with medium confidence**

**Figure 4.12: Confusion Matrix Normalized**

**A graph of a graph showing the difference between confidence and confidence

Description automatically generated with medium confidence**

**Figure 4.13: Confidence Curve**

**A collage of different colored graphs

Description automatically generated with medium confidence**

**A group of blue graphs

Description automatically generated with medium confidence**

**A graph of a graph showing the difference between confidence and confidence

Description automatically generated with medium confidence**

**Figure 4.14: Precision Confidence Curve**

**A screen shot of a graph

Description automatically generated**

**Figure 4.15: Precision Recall Curve**

**A graph of a graph of a graph

Description automatically generated with medium confidence**

**Figure 4.16: Recall Confidence Curve**

**A graph of a graph

Description automatically generated with medium confidence**

**Figure 4.17: Results**

# Chapter 7: Conclusion and Future Work

* 1. **Conclusion**

This project aimed to address the critical issue of water safety by developing a system that rapidly detects waterborne pathogens using a deep learning-based model integrated into a mobile application. By leveraging the YOLOv8 model, the system successfully identifies harmful bacteria such as E. coli, yeast, and other particles with an accuracy of 84.56%. The integration with a Flutter mobile application enables users to easily upload images of water samples and receive immediate feedback on water safety. This solution not only reduces the time required for water contamination testing but also provides a user-friendly interface accessible to individuals in regions with limited access to clean water. The success of the project demonstrates the potential of combining advanced AI techniques with mobile technology to solve real-world problems.

**7.2. Future Work**

Looking forward, the next phase of development will focus on creating a portable hardware device, such as a mini microscope, capable of capturing high-resolution images of water bacteria in real time. This device will be designed to work seamlessly with the existing mobile application, enabling users to perform on-the-spot water testing without the need for an external microscope or camera. The development of this portable device will involve collaboration with hardware engineers and further refinement of the image processing capabilities to ensure high accuracy and reliability. Additionally, future work will explore enhancing the deep learning model's accuracy, expanding the range of detectable pathogens, and integrating more advanced features to improve user experience and system robustness.

# References

1 Yoong-Ling Oon,# 1 , 2 , † Yoong-Sin Oon,# 1 , 2 , † Muhammad Ayaz, 2 , 3 , 4 Min Deng, 2 Lu Li, 2 , 4 and Kang Songcorresponding author 2 , 4 , Waterborne pathogens detection technologies: advances, challenges, and future perspectives

2 Object-Oriented Design Methodologies for Software Systems by Luiz Fernando Capretz

<https://core.ac.uk/download/pdf/153777917.pdf>

3 Communication Design Protocol

<https://www.geeksforgeeks.org/communication-protocols-in-system-design/>

4 Girl looking at water sample using Microscope, <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.alamy.com%2Fstock-photo%2Fwomen-looking-through-microscope.html&psig=AOvVaw0rmKiUcT7vyAQKW872Z7lC&ust=1717482458498000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCNDv9IznvoYDFQAAAAAdAAAAABAE>

5 UML Structural Diagrams.

<https://www.geeksforgeeks.org/structural-diagrams-unified-modeling-languageuml/>

6 Types of Testing in Software Engineering.

<https://www.geeksforgeeks.org/types-software-testing/>