Final Year Project Software Design Specification

Segmentation Strategies for Enhancing System Automation

by

Khadija Tariq: BSEF21M003

Tayyaba Abbas: BSEF21M011

Laiba Shakoor: BSEF21M021

Areeba Qamer: BSEF21M027

Project Advisor:
Dr. Muhammad Farooq

Executive Summary:

"The **'Segmentation Strategies for Enhancing System Automation'** project aims to improve automation by using segmentation techniques in telecommunications, healthcare, and agriculture. This system will employ segmentation algorithms to optimize processes like network resource allocation, bed sore monitoring, and plant irrigation, improving operational efficiency, accuracy, and real-time adaptability across industries."

Collaboration with industry stakeholders is paramount to the success of this project, ensuring that the solutions developed meet real-world needs and challenges. By leveraging cutting edge technologies and innovative segmentation strategies, this project aspires to set a new benchmark for automation across these critical sectors, driving advancements that contribute to overall economic growth and enhanced quality of life.

Table of Contents

[1]

[1] Once the document is completed, Right click on table of contents, Choose “Update Field” then choose “Update entire table” to automatically update table of contents

List of Tables

System Design

1. Assumptions and Dependencies

Assumptions:

1. Reliable and consistent data collection methods for telecommunication tower signal strength, healthcare monitoring, and agricultural data are available.
2. The hardware infrastructure (e.g., sensors, servers, communication devices) required for data acquisition and processing is in place and operational.
3. The users (e.g., healthcare providers, telecom engineers, agricultural experts) are trained in using the system interface and interpreting the results.
4. The system will operate in environments with stable network connectivity, ensuring uninterrupted data transfer.
5. There will be sufficient storage capacity to handle historical data for patients, signal strength, and agricultural growth patterns.

Dependencies:

1. Dependence on real-time data acquisition hardware, such as sensors for tower signal strength, pressure mapping for healthcare, and soil/plant sensors for agriculture.
2. Integration with external APIs for predictive analytics, such as weather forecasts (for agriculture) or demographic data (for healthcare resource optimization).
3. Dependence on external libraries or frameworks for error handling, data encryption (AES-256), and performance optimization.
4. Dependency on third-party backup systems to ensure secure and timely data storage.

2. Limitations

Functionality:

1. The segmentation algorithms may have reduced accuracy in scenarios with incomplete or noisy data inputs (e.g., faulty sensors or missing healthcare parameters).
2. Resource optimization decisions are limited by the quality of the input data and the precision of the implemented algorithms.

Performance:

1. Real-time segmentation processes are designed to execute within 5 seconds but may face delays under heavy computational loads or high data volumes.
2. The system's response time may be affected by poor network bandwidth or latency in remote or rural areas.

Compatibility:

1. Compatibility issues may arise if the system is deployed on hardware or operating systems that do not meet the minimum technical specifications.
2. Limited scalability on older infrastructure may hinder horizontal and vertical growth.

3. Risks

Risk 1: Data Integrity and Accuracy

1. **Potential Impact**: Inaccurate segmentation or predictions may result in suboptimal resource allocation or incorrect risk assessments.
2. **Mitigation**: Implement robust data validation checks and integrate error-detection mechanisms at each processing stage.

Risk 2: System Downtime

1. **Potential Impact**: Service interruptions can affect real-time decision-making in critical domains like healthcare or telecommunication.
2. **Mitigation**: Implement high availability through load balancing, failover mechanisms, and regular backups.

Risk 3: Privacy and Security Breaches

1. **Potential Impact**: Unauthorized access to sensitive data (e.g., patient records, telecom usage data) may lead to legal and reputational consequences.
2. **Mitigation**: Enforce strict data encryption (AES-256), secure authentication methods, and compliance with privacy regulations such as GDPR or HIPAA.

Risk 4: Resource Limitations

1. **Potential Impact**: Insufficient computational resources during peak loads may degrade performance.
2. **Mitigation**: Design the system to support dynamic scaling of resources (both horizontal and vertical) based on demand.

Risk 5: User Interface Complexity

1. **Potential Impact**: A complex or unintuitive UI may reduce user adoption and effectiveness.
2. **Mitigation**: Conduct usability testing and adhere to WCAG 2.1 AA standards for accessible and user-friendly design.

Risk 6: Integration Challenges

1. **Potential Impact**: Dependencies on third-party APIs or hardware may lead to delays or incompatibility issues.
2. **Mitigation**: Test integrations thoroughly and provide fallback mechanisms for critical functionalities.

2. Requirements Traceability Matrix

Table 1: Requirements Traceability Matrix

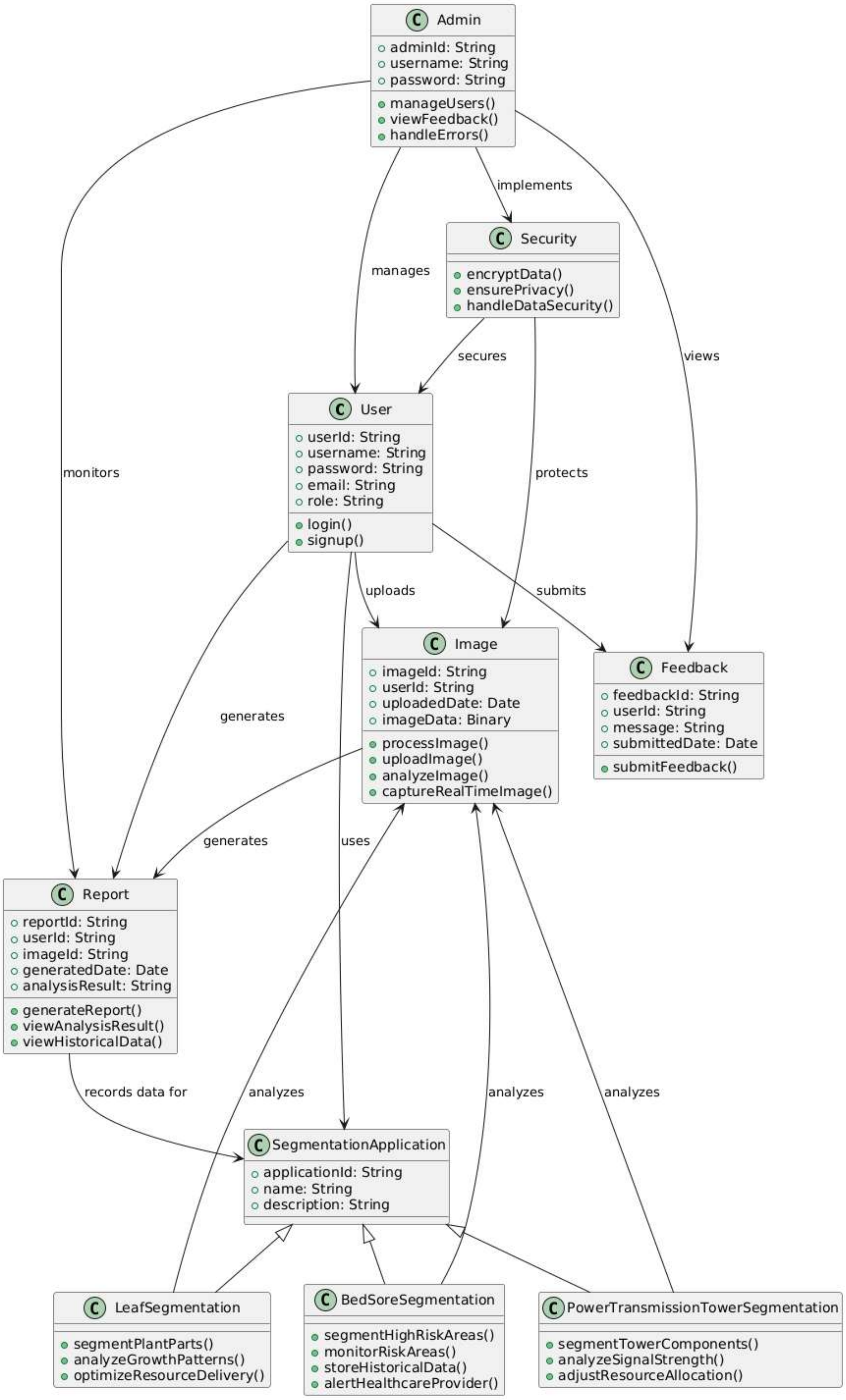
Requirement ID	Requirement Description	Design Specification
FR-1	The system shall segment tower components, such as peaks, cross arms, and insulators, based on varying user density and signal strength.	Component: "Telecommunication Tower Segmentation"
FR-2	The system shall analyze and display real-time signal strength data for each segmented component of the tower.	Component: "Signal Strength Analysis"
FR-3	The system shall adjust resource allocation to tower segments based on demand peaks in real-time.	Component: "Real-Time Resource Allocation"
FR-4	The system shall segment patient body areas at high risk of bed sores based on pressure points and anatomical locations.	Component: "Healthcare Risk Segmentation"
FR-5	The system shall provide continuous monitoring and alert healthcare providers when a high-risk area requires intervention.	Component: "Healthcare Monitoring & Alerts"
FR-6	The system shall store and analyze historical data on segmented risk areas for each patient to improve monitoring strategies.	Component: "Patient Historical Data Management"
FR-7	The system shall segment plants based on leaf size and root structure to tailor water and nutrient delivery.	Component: "Agriculture Segmentation"
FR-8	The system shall track and analyze the growth patterns of segmented plant areas over time to predict resource needs and optimize yield.	Component: "Growth Analysis and Prediction"
NFR-1	Achieve an MTBF (Mean Time Between Failures) exceeding 5,000 operational hours to ensure high reliability.	Quality Attribute: "System Reliability"
NFR-2	Set an MTTR (Mean Time to Recover) target under 30 minutes.	Quality Attribute: "Quick Recovery Mechanisms"
NFR-3	Schedule incremental backups at 2 a.m. daily to safeguard critical data.	Feature: "Automated Data Backup Scheduler"

NFR-4	Maintain data integrity with a target of 98%.	Feature: "Data Validation and Consistency"
NFR-5	Include real-time error detection, automatic retry logic for segmentation failures, and error logs.	Component: "Error Handling Framework"
NFR-6	Provide an interface enabling users to perform essential tasks within 10 minutes, meeting usability standards.	Component: "User-Friendly Interface Design"
NFR-7	Ensure compliance with WCAG 2.1 AA standards for accessible design.	Feature: "Accessible Design Features"
NFR-8	Real-time segmentation processes should return results within 5 seconds.	Component: "Performance Optimization"
NFR-9	System architecture shall support horizontal and vertical scaling for future growth.	Feature: "Scalable Infrastructure Design"
NFR-10	AES-256 encryption for all stored and transmitted data to protect against unauthorized access.	Component: "Data Security and Encryption"

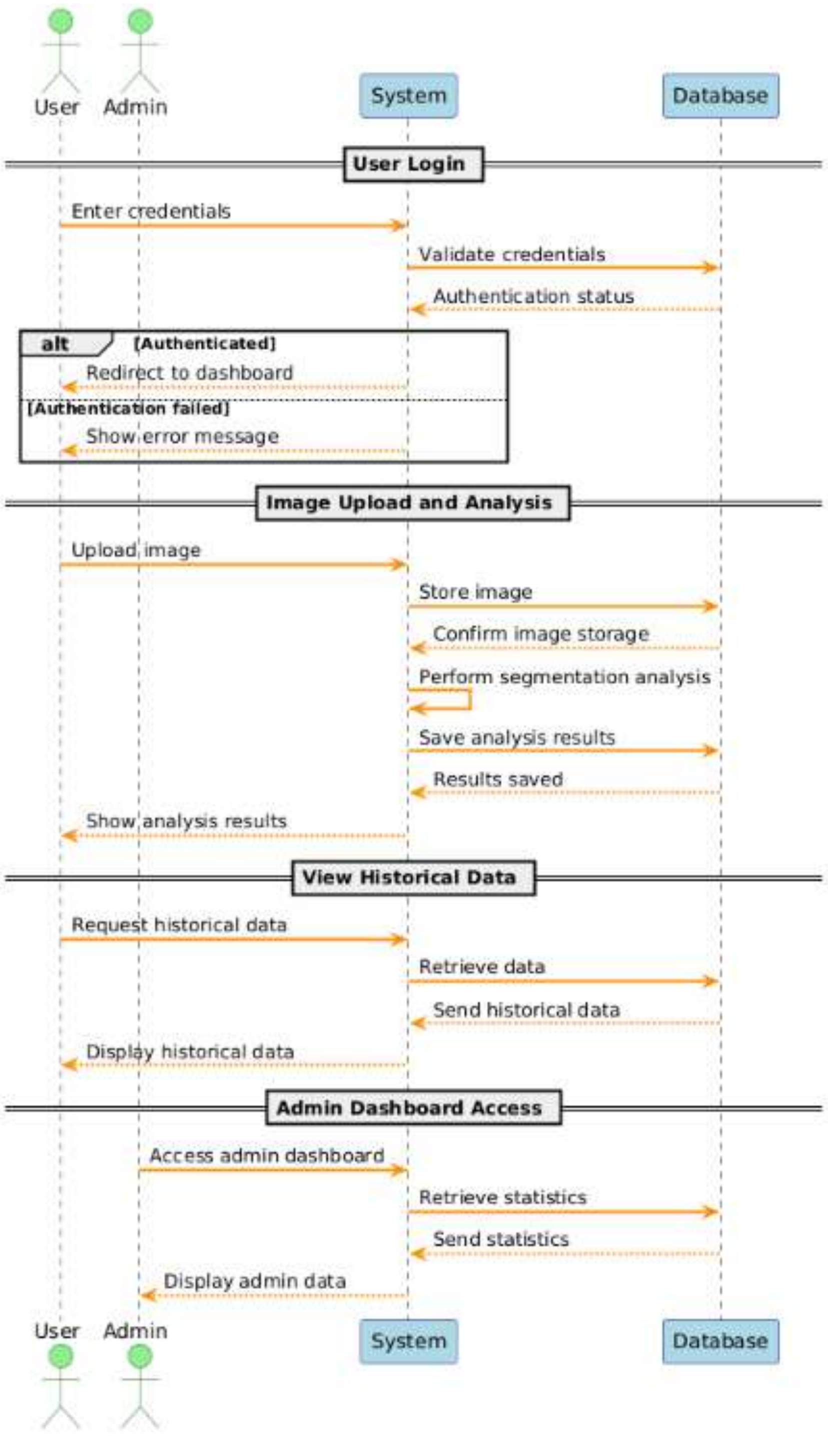
3 Design Models

3.1 Architecture Designs

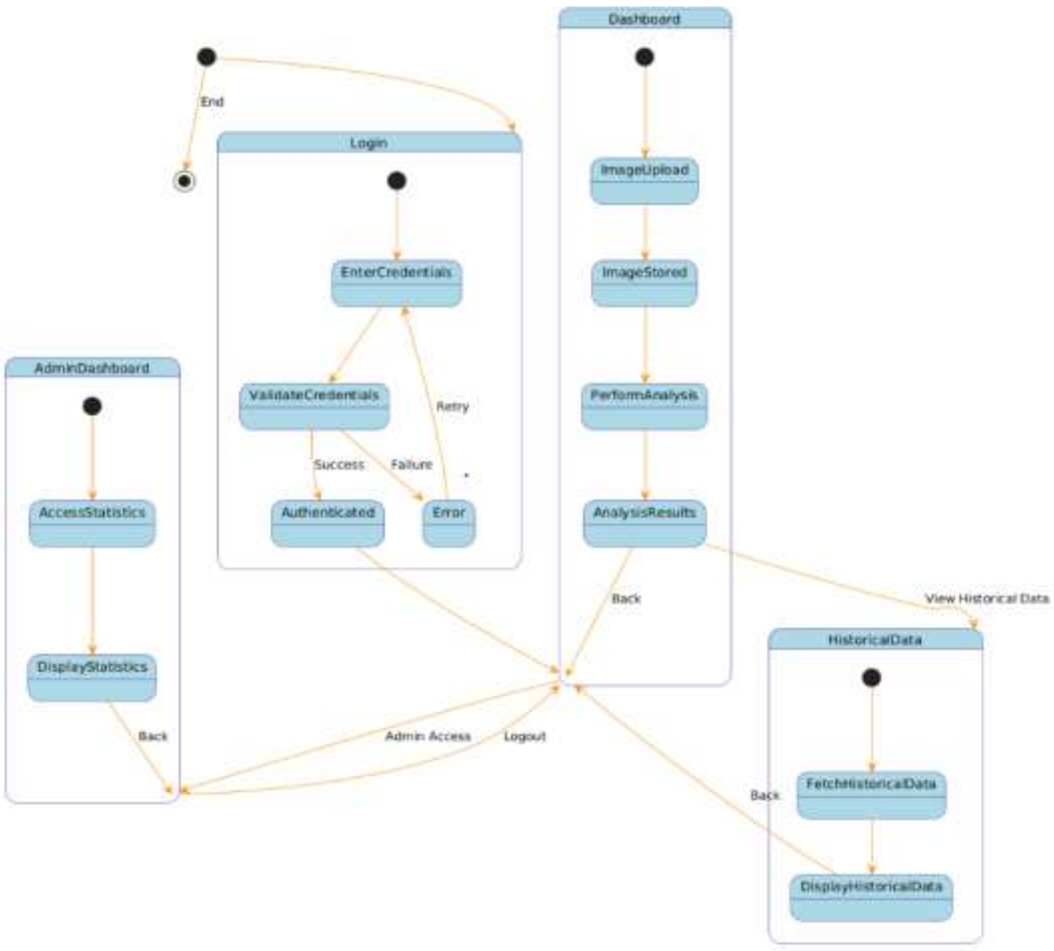
1. **Class Diagram**


Admin
+ adminId: String
+ username: String
+ password: String
+ manageUsers()
+ viewFeedback()
+ handleErrors()
implements
Security
+ encryptData()
+ ensurePrivacy()
+ handleDataSecurity()
manages
secures
views
monitors
protects
User
+ userId: String
+ username: String
+ password: String
+ email: String
+ role: String
+ login()
+ signup()
uploads
submits
Image
+ imageId: String
+ userId: String
+ uploadedDate: Date
+ imageData: Binary
+ processImage()
+ uploadImage()
+ analyzeImage()
+ captureRealTimeImage()
Feedback
+ feedbackId: String
+ userId: String
+ message: String
+ submittedDate: Date
+ submitFeedback()
generates
generates
uses
Report
+ reportId: String
+ userId: String
+ imageId: String
+ generatedDate: Date
+ analysisResult: String
+ generateReport()
+ viewAnalysisResult()
+ viewHistoricalData()
records data for
analyzes
analyzes
analyzes
SegmentationApplication
+ applicationId: String
+ name: String
+ description: String
LeafSegmentation
+ segmentPlantParts()
+ analyzeGrowthPatterns()
+ optimizeResourceDelivery()
BedSoreSegmentation
+ segmentHighRiskAreas()
+ monitorRiskAreas()
+ storeHistoricalData()
+ alertHealthcareProvider()
PowerTransmissionTowerSegmentation
+ segmentTowerComponents()
+ analyzeSignalStrength()
+ adjustResourceAllocation()

2.Interaction Diagram (Either sequence or collaboration)

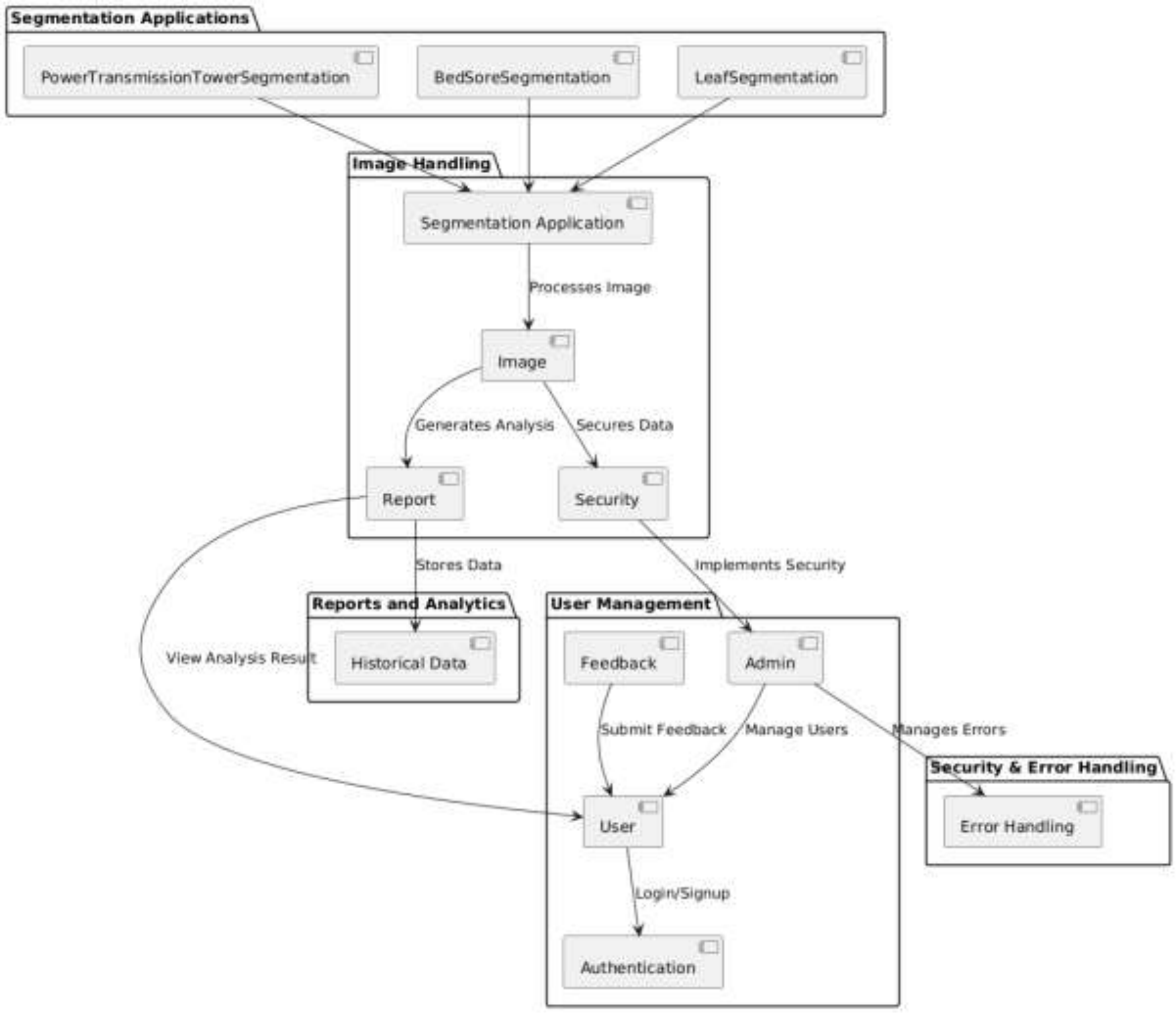


2.State Transition Diagram (for the projects which include event handling and backend processes)



3.1 Architectural Design

UML Class Diagram:



Mapping of components:

1. Presentation Layer (UI Tier)

- **Modules/Components**:
    - User Management
        - User
        - Authentication
    - Feedback
    - View Analysis Results (from Reports and Analytics)

Reason: This layer handles the user interface, input, and display functionalities like login, signup, feedback submission, and result views.

2. Application Layer (Business Logic Tier)

- **Modules/Components**:
    - Segmentation Applications
        - PowerTransmissionTowerSegmentation
        - BedSoreSegmentation
        - LeafSegmentation
        - Segmentation Application
    - Image Handling
        - Image (Processes Images)
        - Report (Generates Analysis)

Reason: This layer contains the business logic for image processing, segmentation, and analysis.

3. Data Access Layer

- **Modules/Components**:
    - Reports and Analytics
        - Historical Data
        - Report (Stores Data)
    - Image Handling
        - Image (Data Storage/Upload)

Reason: This layer manages data storage, retrieval, and communication with databases.

4. Infrastructure Layer (Security & System Management Tier)

- **Modules/Components**:
    - Security & Error Handling
        - Security (Implements Encryption/Privacy)
        - Error Handling
    - Admin (Manages Users, Handles Errors, and Ensures System Integrity)

Reason: This layer supports system-level operations like security enforcement, error handling, and system administration.

Components	Mapped To	Function
User	Presentation Layer	Provides user interface for login/signup operations.
Authentication	Presentation Layer	Manages user authentication for secure access.
Admin	Infrastructure Layer	Manages users, errors, and ensures system integrity.
Feedback	Presentation Layer	Allows users to submit feedback to the system.
PowerTransmissionTowerSegmentation	Application Layer	Segments tower components for analysis.
BedSoreSegmentation	Application Layer	Segments high-risk body areas for bed sores.
LeafSegmentation	Application Layer	Segments plant areas based on leaf size and roots.
Segmentation Application	Application Layer	Provides logic for handling different segmentation apps.
Image	Application Layer	Processes and uploads images for analysis.
Report	Application Layer	Generates and provides analysis reports.
Historical Data	Data Access Layer	Stores and retrieves historical segmentation data.
Security	Infrastructure Layer	Implements data security, encryption, and privacy.
Error Handling	Infrastructure Layer	Manages system errors and ensures reliable operation.
Report (View Analysis Results)	Presentation Layer	Displays analysis results to the user.

3.2Data Design

3.2.1 Data Dictionary

 Alphabetical List of System Entities or Major Data with Types and Descriptions:

Term/Entity	Description
BedSoreAnalysisResults	Object that holds segmented bed sore data, aiding in analysis and progression tracking.
ElectricTowerParts	Data structure for segmentation of electric tower components for further calculations.
FertilitySegmentData	Stores soil region segmentation data, facilitating fertility analysis.
HeightMeasurement	Float type; represents computed height of segmented electric tower parts.
ImageInput	Input file type for images, supporting various formats for analysis.
SegmentationAlgorithm	Identifier for AI models like "UNet" used to process segmentation tasks.
SoilFertilityScores	Array of float values indicating fertility across different soil regions.
TowerWidthCalculation	Float type; denotes calculated width of segmented electric tower components.

 Structured Approach: Functions and Function Parameter

Function Name	Description	Parameters (Data Type)
segmentImage	Performs image segmentation based on input data.	image (ImageInput), model (String)
calculateTowerDimensions	Computes the height and width of the tower parts from segmented data.	segmentationResults (ElectricTowerParts)
analyzeSoilFertility	Analyzes soil fertility based on segmented regions.	fertilityData (FertilitySegmentData)
monitorBedSoreProgression	Tracks the progression of bed sores using segmentation data.	bedSoreData (BedSoreAnalysisResults)
analyzeWebResults	Analyzes uploaded website results for patterns or insights.	resultData (File), criteria (String)
downloadReport	Generates and downloads a report based on analysis.	reportFormat (String), analysisData (File)
viewHistory	Retrieves and displays the history of previous analyses.	userId (String), dateRange (String)

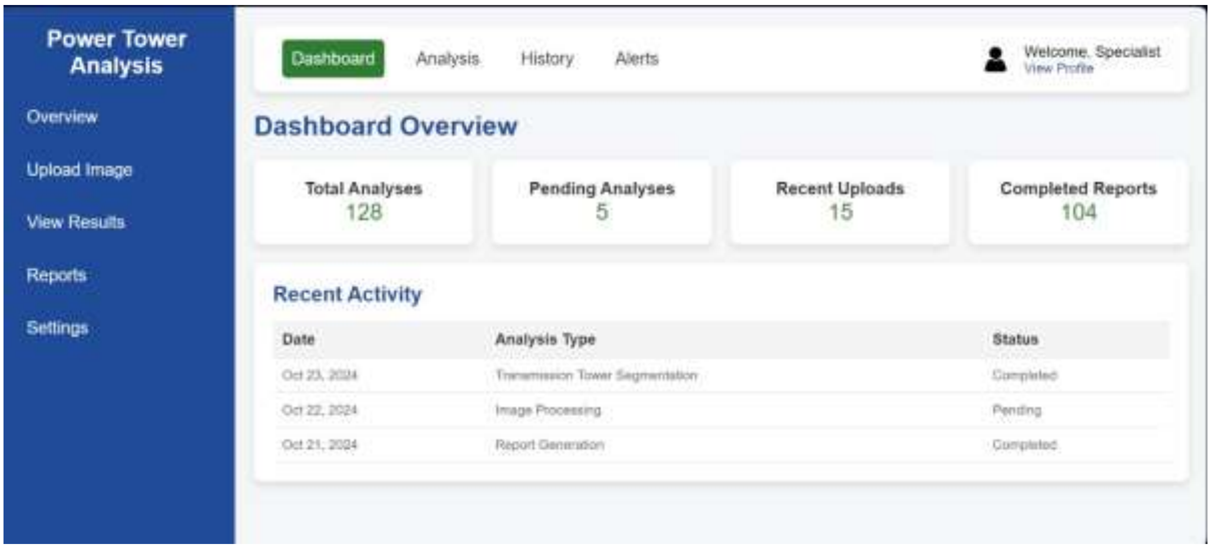
 Object-Oriented (OO) Approach: Objects, Attributes, Methods, & Method Parameters:

1. **Object: ElectricTowerParts**
   - **Attributes**:
     - partsList (List of Strings): Names of segmented parts.
     - height (Float): Calculated height of tower.
     - width (Float): Calculated width of tower.
   - **Methods**:
     - calculateHeight(): Computes height from segmentation data.
     - calculateWidth(): Computes width from segmentation data.
2. **Object: BedSoreAnalysisResults**
   - **Attributes**:
     - soreRegions (List of Strings): Names or labels of identified sore regions.
     - severityScores (List of Floats): Severity levels of each sore region.
   - **Methods**:
     - trackProgress(): Analyzes the changes in bed sore severity over time.
3. **Object: SoilFertilitySegmentData**
   - **Attributes**:
     - fertilityScores (List of Floats): Fertility levels for each region.
     - regionLabels (List of Strings): Names or identifiers for soil regions.
   - **Methods**:
     - evaluateFertility(): Computes an overall score or classification based on segmented data.

3.3 User Interface Design

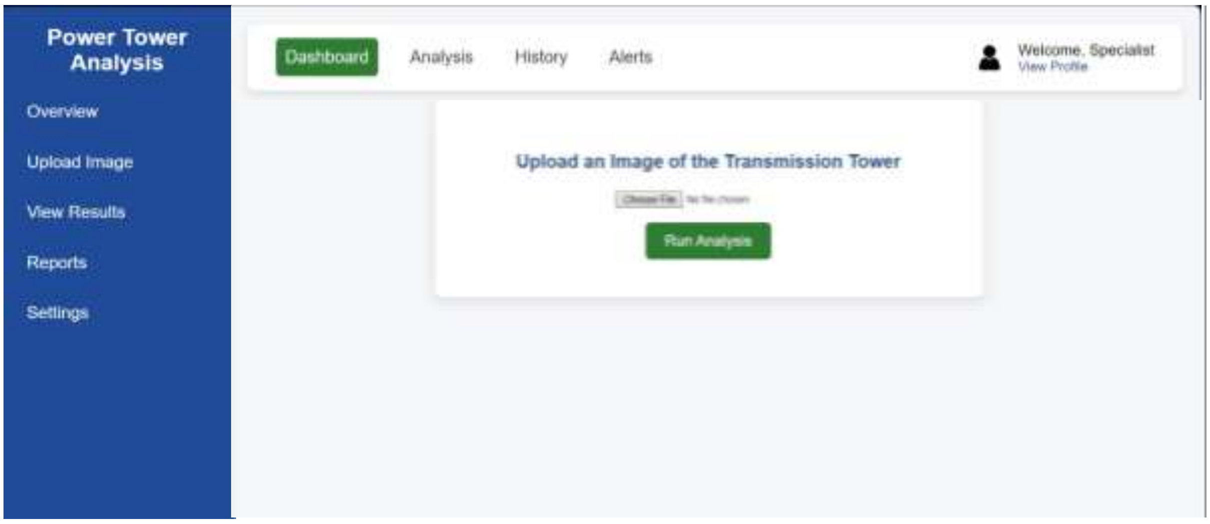
3.3.1 Screen Images

1. User Dashboard (Home Screen)

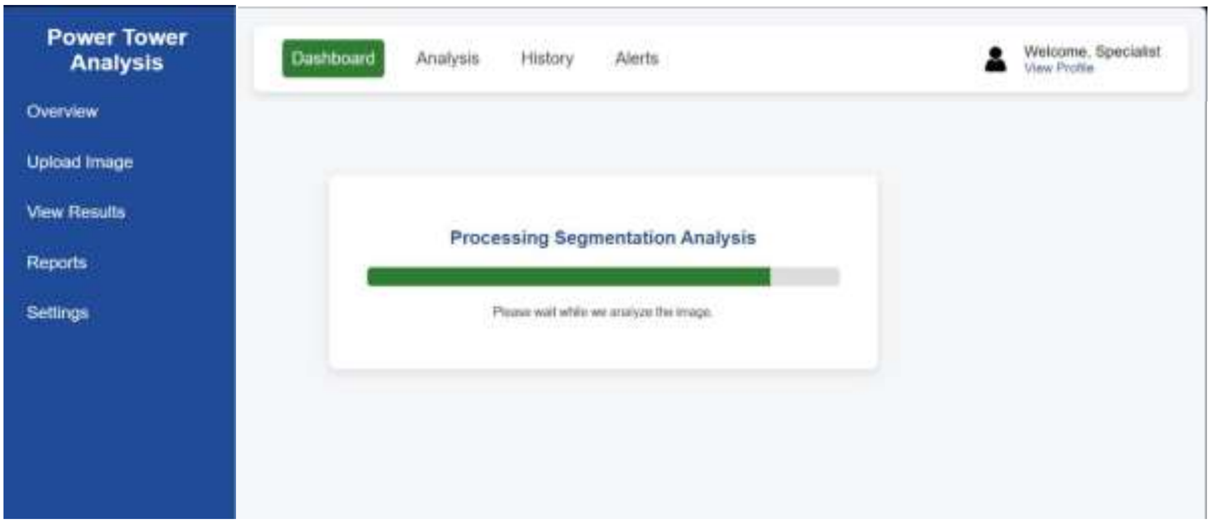


Date	Analysis Type	Status
Oct 23, 2024	Transmission Tower Segmentation	Completed
Oct 22, 2024	Image Processing	Pending
Oct 21, 2024	Report Generation	Completed

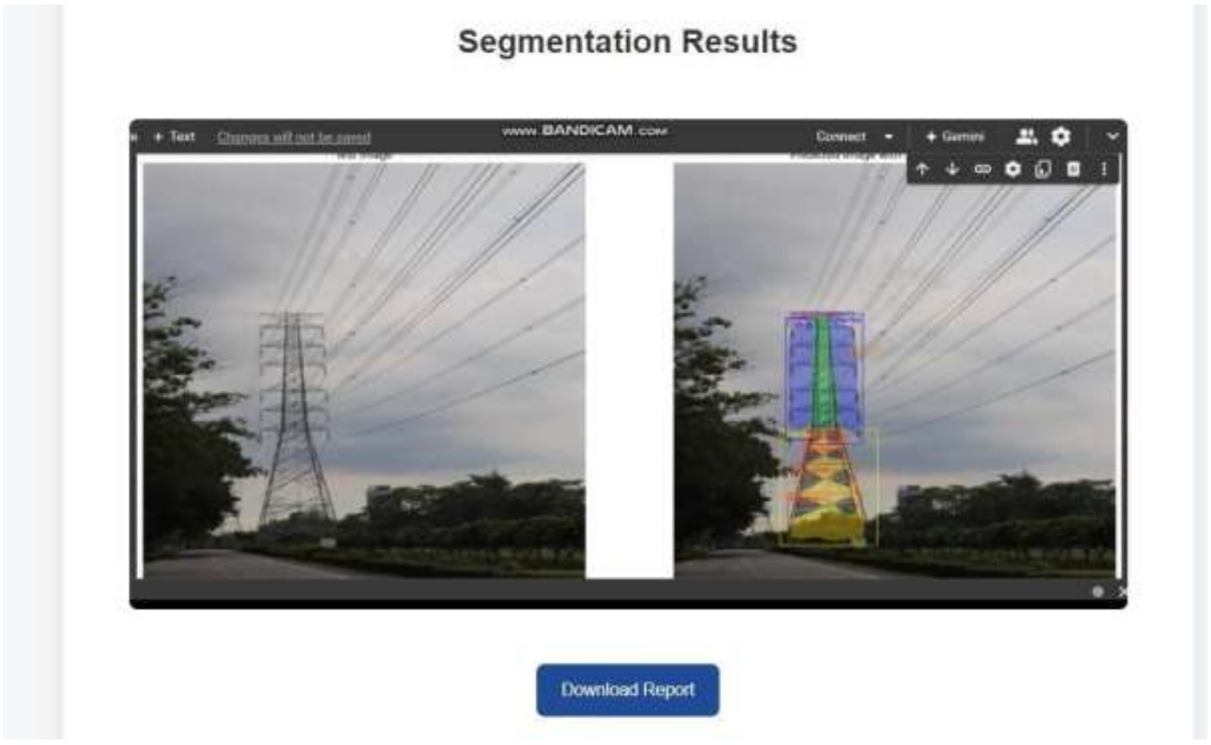
2. Uploading an Image



3. Running Segmentation Analysis



4. Viewing and Adjusting Segmentation Results



3.3.2. Screen Objects and Actions

Use Case 1: Accessing the Overview Page

- **Screen Object:**
  - **Overview Link**: A clickable link on the sidebar that brings the user to the main dashboard page.
- **Action:**
  - The user clicks on the **Overview Link** to view a summary of key activities, including the total analyses, pending analyses, recent uploads, and completed reports.
- **Feedback Information:**
  - After clicking the link, the user is directed to the **Dashboard Overview**, where the system updates and displays the latest activity status, including analysis progress and report status.

Use Case 2: Uploading an Image and Running the Analysis

- **Screen Objects:**
  - **Upload Image Button**: Allows the user to select a file (image of the transmission tower) from their computer.
  - **Run Analysis Button**: Initiates the analysis after the image is uploaded.
- **Actions:**
  - The user clicks the **Upload Image Button** to choose an image file.
  - After selecting the file, the user presses the **Run Analysis Button** to start the analysis process.

  Feedback Information:

  - After the user clicks "Run Analysis," the system will show a progress indicator, and once the analysis is complete, the system will display the results or prompt the user to view the report.

Use Case 3: Opening Settings

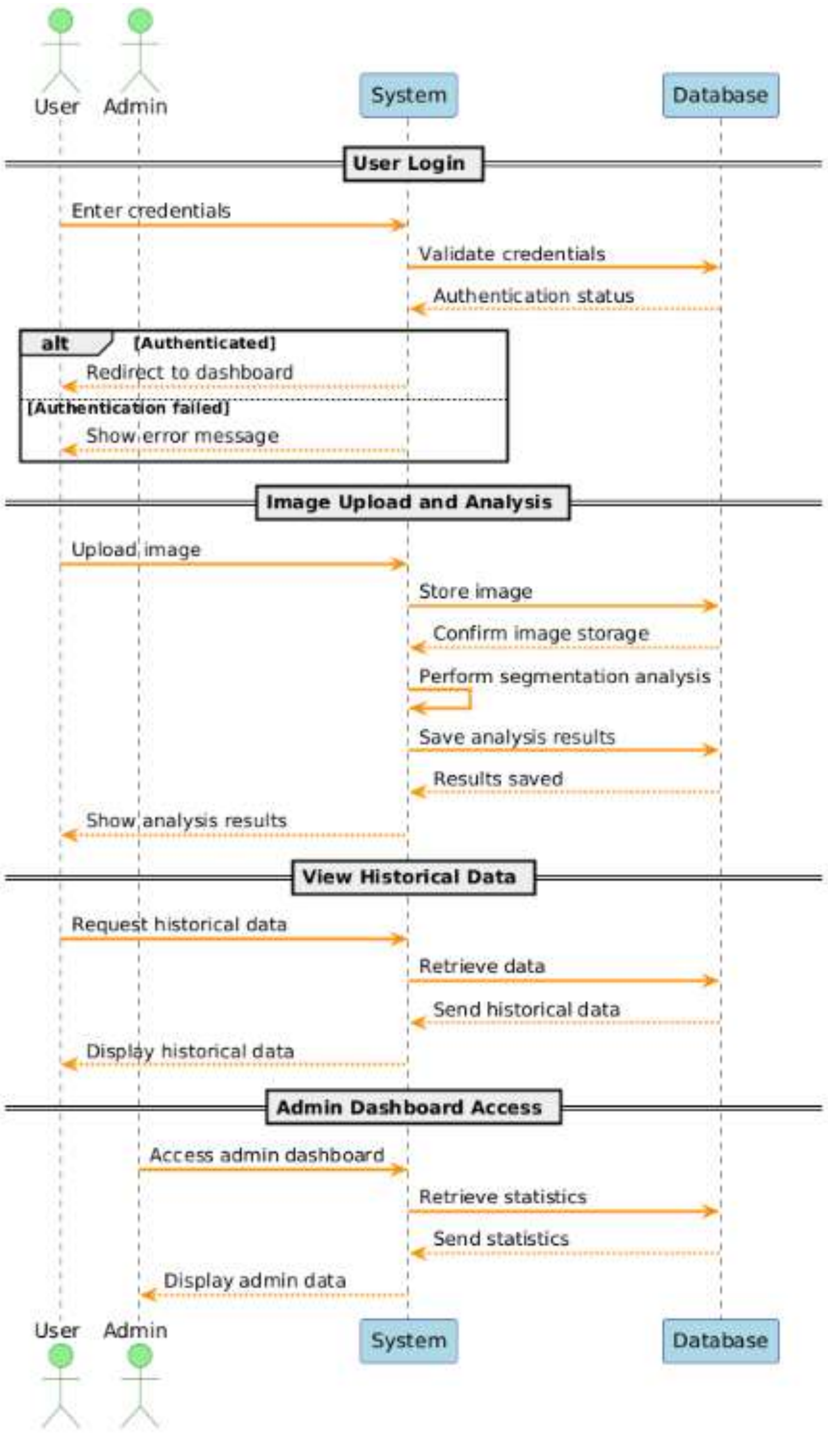
- **Screen Object:**
  - **Settings Link**: A clickable link that opens the settings page.
- **Action:**
  - The user clicks on the **Settings Link** to adjust their preferences or configurations.
- **Feedback Information:**
  - After clicking, the system will display the settings page where the user can update their profile, change settings, or customize system preferences.

Use Case 4: Accessing Profile

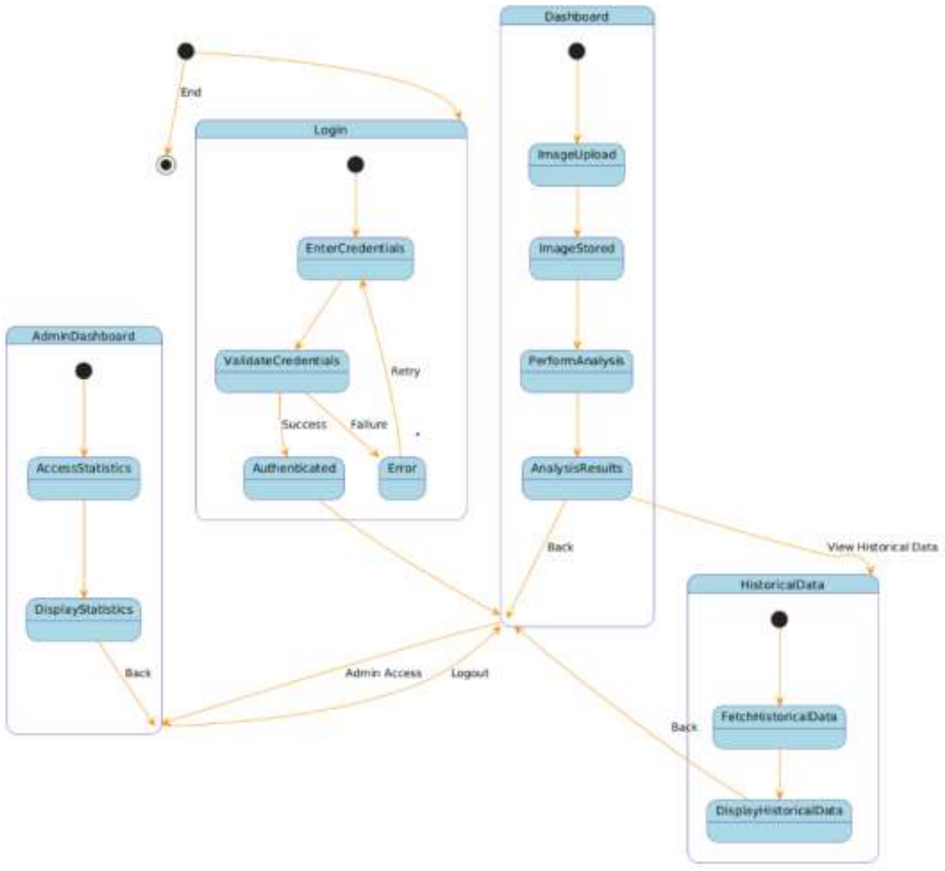
- **Screen Object:**
  - **Profile Section**: Located at the top of the sidebar, this section shows the user's name and provides access to their profile details.
- **Action:**
  - The user clicks on the **Profile Section** to view or edit personal information.
- **Feedback Information:**
  - Clicking on the profile section will direct the user to their personal profile page, where they can view or update their profile information, such as name, email, and preferences.

3.4 **Behavioural Model**

Interaction Diagrams:



State Diagrams:



4. Design Decisions

Highlight the design choices while designing your system (e.g. choosing the Object oriented design pattern, choosing between the algorithmic approaches, normalization level of the database etc). Clearly specify which approach you have used and why you have made a specific design choice.

1. Architecture and Design Pattern

- **Decision:** Adopted an Object-Oriented Design (OOD) with a multi-tier architecture.
- **Reason:** Object-Oriented Design promotes modularity, encapsulation, and reusability. A multi-tier architecture (e.g., Presentation, Business Logic, and Data Access Layers) ensures separation of concerns, making the system easier to manage, debug, and scale.
- **Implementation:**
    - Presentation Layer: React.js was used to create a user-friendly and responsive interface.

    - Business Logic Layer: Node.js handled server-side logic, processing user requests and interacting with other services.
    - Data Layer: MongoDB was chosen for its flexible schema and ability to handle unstructured data efficiently.

2. Algorithmic Approach

- **Decision:** Computer vision algorithm, machine learning algorithms etc
- **Reason:** The algorithm was selected based on its performance, accuracy, and computational efficiency.
- **Implementation:**
    - Alternative algorithms were benchmarked to compare their time complexity and output accuracy.
    - For example, in a real-time signal analysis module, algorithms were chosen to ensure low-latency processing (e.g., Fast Fourier Transform for signal processing).

3. Database Design and Normalization

- **Decision:** Used a NoSQL database (MongoDB) and applied partial normalization up to the third normal form (3NF).
- **Reason:**
    - NoSQL databases provide schema flexibility, which is essential for handling varying data types such as segmentated data, patient records, and plant information.
    - Normalization reduces redundancy and improves data consistency, especially for transactional components like healthcare data or resource allocation.
- **Implementation:**
    - Embedded documents were used to minimize joins, such as embedding plant segmentation data within plant records.
    - Indexing was implemented on critical fields (e.g., IDs, timestamps) to optimize query performance.

4. Choice of Frameworks and Tools

- **Frontend:** React.js

  Reason: React offers a component-based approach, enabling reusable UI components and efficient state management.
- **Backend:** Node.js

  Reason: Node.js provides non-blocking I/O, making it suitable for real-time data processing and communication.
- **Database:** MongoDB

  Reason: MongoDB's document-based storage is ideal for storing complex, hierarchical data, such as segmentation or monitoring records.

5. Scalability and Performance

- **Decision:** Used a microservices-based architecture to handle independent system components (e.g., telecommunications service, healthcare service, agriculture service).
- **Reason:** Microservices ensure modularity and scalability. Each service can be scaled or updated independently without affecting others.
- **Implementation:**

    - Communication between services was established using RESTful APIs.
    - Deployed services on a cloud platform (e.g., AWS, Azure) for scalability and high availability.

6. User Experience (UX)

- **Decision:** Designed a responsive and intuitive UI using React and a CSS framework like Tailwind or Bootstrap.
- **Reason:** A clean and responsive UI ensures that the system is user-friendly across various devices, enhancing user adoption.
- **Implementation:**
    - Included features like real-time feedback (e.g., alerts) and dynamic resource allocation visualizations.
    - Conducted usability testing to refine the interface based on user feedback.

7. Security Measures

- **Decision:** Implemented robust authentication and authorization mechanisms.
- **Reason:** To safeguard sensitive data (e.g., healthcare records, user credentials) and comply with security standards (e.g., HIPAA, GDPR).
- **Implementation:**
    - JWT (JSON Web Tokens) was used for user authentication.
    - Role-based access control (RBAC) ensured that users only had access to data relevant to their roles.

8. Real-Time Features

- **Decision:** Integrated WebSocket for real-time communication in modules like Continuous Monitoring and Alerts.
- **Reason:** WebSocket provides low-latency, bi-directional communication, making it suitable for real-time applications like signal analysis and live monitoring.
- **Implementation:**
    - Socket.IO library was used to handle WebSocket connections efficiently.
    - Failover mechanisms ensured smooth fallback to polling if WebSocket connectivity failed.

9. Data Processing and Storage

- **Decision:** Used a hybrid approach for data processing—real-time processing for active monitoring and batch processing for historical data analysis.
- **Reason:** Real-time data ensures immediate feedback, while batch processing optimizes resource usage for non-critical tasks.
- **Implementation:**
    - Data pipelines were established to handle large-scale data from healthcare, agriculture, and telecommunications services.
    - Aggregation pipelines in MongoDB facilitated efficient historical data analysis.

10. Logging and Monitoring

- **Decision:** Integrated logging and monitoring tools like ELK Stack or Prometheus.
- **Reason:** These tools provide real-time insights into system performance and help diagnose issues proactively.
- **Implementation:**

- Logs captured user activity, system errors, and performance metrics.
- Alerts were set up to notify the development team of potential failures or bottlenecks.

5. Summary

The development of an artificial intelligence-based platform for tower analysis involves a comprehensive approach to address real-life challenges. This project includes a dataset of towers that have been meticulously trained, comprising five distinct classes: peak, tower body, cage, cross arm, and base pattern. The images have been annotated for labeling, and segmentation techniques have been employed for accurate body part identification. Additionally, the platform is designed to identify the count of base patterns, determine the dimensions of each part, and measure the overall height and width of the towers. A user-friendly interface will be created for users to log in and sign up, allowing them to upload images of towers for analysis. The system will then display the results based on the pre-trained AI models. Moreover, a similar approach will be applied to analyze, our second real life problem, leaf images taken from soil, providing insights into growth metrics, healing, and non-healing conditions in our third real life problem of bedsores.
This initiative aims to provide a reliable and effective solution for users seeking to understand and assess tower structures and plant health through advanced AI techniques.

References

- **IEEE 830-1998** – *Guidelines for Software Requirements Specifications* Use for: Functional/Non-functional requirements structure and best practices.
  Functional and non functional requirements
- **Use Case Modeling by Kurt Bittner and Ian Spence** Use for: Developing and documenting use cases.
  Use Case Modeling