Water Level Monitoring App

UCS 503 Software Engineering Project Report

END-Semester Evaluation

Submitted by:

Nandini Garg(102303383)

Ritvik Singla(102303204)

Sachleen Kaur(102303188)

Swastik Gupta(102483086)

BE Third Year, COE

Group No: 3C17

Submitted to:

Dr Deep Mann

Computer Science and Engineering Department

TIET, Patiala

November, 2025

TABLE OF CONTENTS

1. Project Selection Phase

1.1 Software Bid

Group:3C17

Team ID:21

Project Name: Water level monitoring app

Name	Roll no	Project Experience	Programming Languages
Ritvik Singla	102303204	NavSight	Kotlin,Firebase, PL/SQL
Swastik Gupta	102483086	Gait Analysis	Python, PL/SQL
Sachleen Kaur	102303188	Insurance Management System	Python, PL/SQL
Nandini Garg	102303383	Insurance Management System	Python, PL/SQL

1.2 Project Overview

The Water Level Monitoring System is designed to modernize and automate the process of collecting, validating, and reporting river water levels, a task currently performed manually by field workers. Traditional methods rely on physical visits to river sites where workers read gauge posts and record water levels on paper. This manual approach often results in delayed reporting, human error, fake entries, and a lack of proof that the field worker was actually present at the site. These limitations directly affect the accuracy of flood prediction models and the efficiency of disaster management authorities. To address these challenges, the proposed system introduces a digital, automated, and verifiable water level monitoring platform that enhances reliability, transparency, and real-time data availability.

The system consists of a mobile application for field workers, a backend server for data processing and storage, and an admin dashboard for monitoring and decision-making. The central idea is to capture a photo of the river gauge post and automatically detect the water level using a machine learning model. This eliminates the need for manual reading and reduces the chances of incorrect or manipulated data entry. The application also integrates GPS-based location verification to ensure that workers are physically present at the designated site when submitting the reading. This acts as a safeguard against fake or remote submissions.

Once the field worker captures an image, the ML model processes it to determine the exact water level. The application then bundles the image, extracted reading, GPS coordinates, timestamp, and site ID into a structured submission. In areas with poor internet connectivity—common in river-bank regions—the system supports offline mode through a local submission storage feature. Submissions are saved locally and synchronized automatically with the central database once internet connectivity is restored. This ensures uninterrupted operations even in remote environments.

On the backend, the Database Manager securely stores all submission records along with the corresponding metadata. Each submission is logged with proof elements such as the captured image and actual geolocation. The admin dashboard provides real-time visualization of water levels across different locations using graphs, color-coded alerts, and site-wise summaries. Admins can track historical readings, verify submissions, monitor field worker activity, and analyze trends for early flood alerts. The dashboard enhances transparency and operational efficiency by giving authorities a centralized and accurate view of river conditions.

The system's workflow is supported by additional components such as the Sync Manager for offline data handling, the Authentication module for user login through OTP verification, and the Location Service for geofencing validation. Together, these components ensure accurate data collection, seamless synchronization, and secure access.

Overall, the Water Level Monitoring System transforms the traditional manual data collection process into a robust digital ecosystem. It increases accuracy, reduces fraud, enhances accountability of field workers, and enables real-time access to actionable data. By leveraging machine learning, mobile technology, and automated verification, the project contributes significantly toward efficient flood management, improved safety measures, and a more responsive disaster monitoring framework.

2. Analysis Phase

2.1 Use Cases

2.1.1 Use Case Diagrams

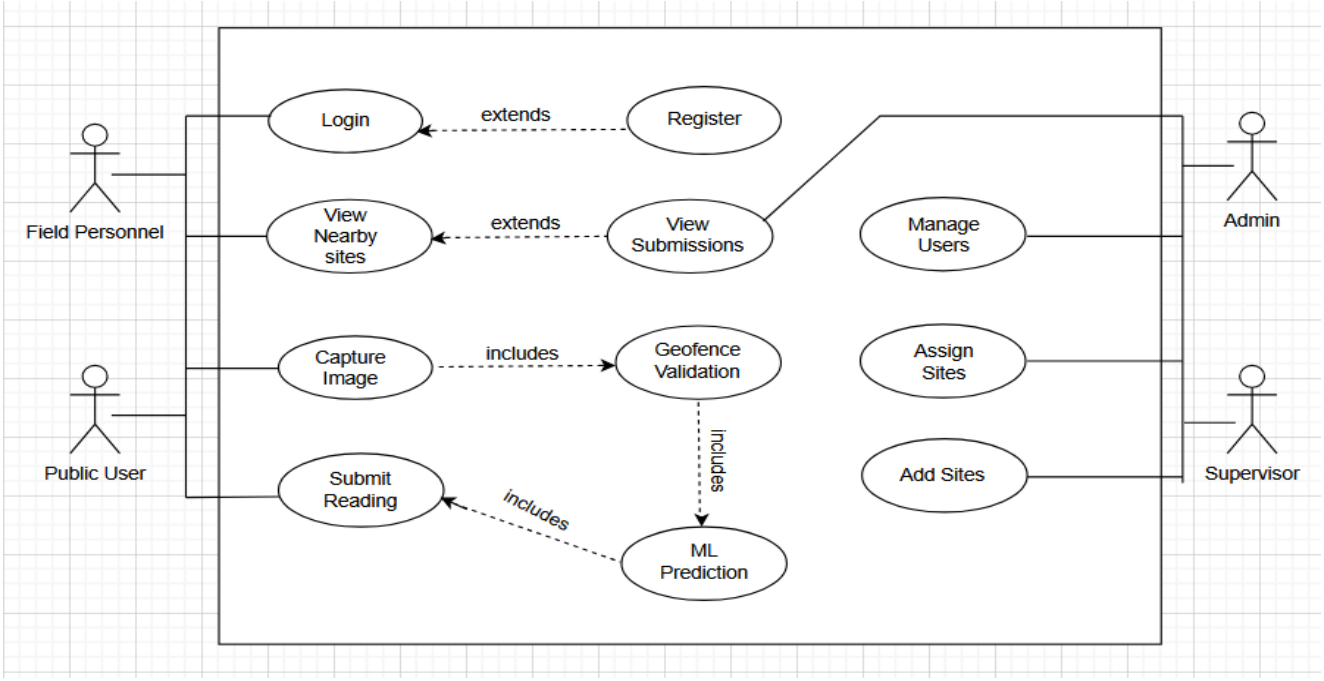


Fig 2.1.1: shows use case diagram

2.1.2 Use Case Templates

Use Case 1: User Registration

Abbreviated Title	Register
Use Case ID	UC01
Actors	Common Person, Field Person, Supervisor, Admin
Description	Enables users to register using their mobile number and select their role for accessing app features.
Pre-Conditions	1. User not previously registered 2. Device connected to Internet
Task Sequence	1. User opens the registration screen. 2. Enters mobile number and taps “Send OTP.” 3. System sends OTP using Firebase Authentication. 4. User enters OTP and verifies. 5. User selects role (Common Person / Field Person / Supervisor / Admin). 6. User details saved to Firebase database.

	7. System confirms successful registration.
Post Conditions	1. New user account created. 2. User role and profile stored securely.
Modification History	November 26, 2025
Authors	Team Hydrowatch

Use Case 2: User Login

Abbreviated Title	Login
Use Case ID	UC02
Actors	All Registered Users
Description	Allows users to log in using their registered mobile number and OTP to access their dashboard.
Pre-Conditions	1. User must be registered. 2. Internet connection active.
Task Sequence	1. User opens login screen. 2. Enters registered mobile number. 3. System sends OTP via Firebase. 4. User enters OTP to verify. 5. System authenticates and grants access to respective dashboard.
Post Conditions	1. User successfully logged in. 2. Session stored for auto-login.
Modification History	November 26, 2025
Authors	Team Hydrowatch

Use Case 3: Capture Water Level Reading

Abbreviated Title	Add Reading
Use Case ID	UC03

Actors	Field Person, Common Person
Description	Allows a user to capture a live photo of the gauge post, automatically detect location, and record water level reading using AI.
Pre-Conditions	1. User logged in. 2. GPS and Camera enabled. 3. Within allowed site radius.
Task Sequence	1. User opens “Add Reading.” 2. App fetches GPS location and displays nearby sites. 3. User selects a site and captures a live photo. 4. ML model predicts water level from photo. 5. User verifies or corrects predicted reading. 6. Data saved to Firebase (photo, GPS, timestamp, reading).
Post Conditions	1. Reading successfully stored. 2. Available for supervisors and admins in dashboard.
Modification History	November 26, 2025
Authors	Team Hydrowatch

Use Case 4: Offline Submission Sync

Abbreviated Title	Offline Sync
Use Case ID	UC04
Actors	Field Person,
Description	Allows data submissions to be stored locally and automatically synced to Firebase when internet is restored.
Pre-Conditions	1. User logged in. 2. Offline mode active (no Internet).
Task Sequence	1. User captures water reading offline. 2. App saves data locally in Room database. 3. App periodically checks network status. 4. Once connected, unsynced data is uploaded automatically.

	5. User notified of sync completion.
Post Conditions	1. Data synchronized to Firebase. 2. Local cache cleared after upload.
Modification History	November 26, 2025
Authors	Team Hydrowatch

Use Case 5: Dashboard Access and Analytics

Abbreviated Title	Dashboard
Use Case ID	UC05
Actors	Supervisor, Admin
Description	Enables supervisors and admins to view site data, analyze water level trends, and manage users.
Pre-Conditions	1. User logged in as Supervisor or Admin. 2. Internet access available.
Task Sequence	1. User opens web dashboard. 2. System loads map view with site data. 3. User selects site to view readings, photos, and graphs. 4. Can filter data by date, site, or user. 5. Admin can assign sites or deactivate users.
Post Conditions	1. Data displayed with analytics. 2. Admin actions saved in database.
Modification History	November 26, 2025
Authors	Team Hydrowatch

2.2 Activity Diagram and Swimlane Diagrams

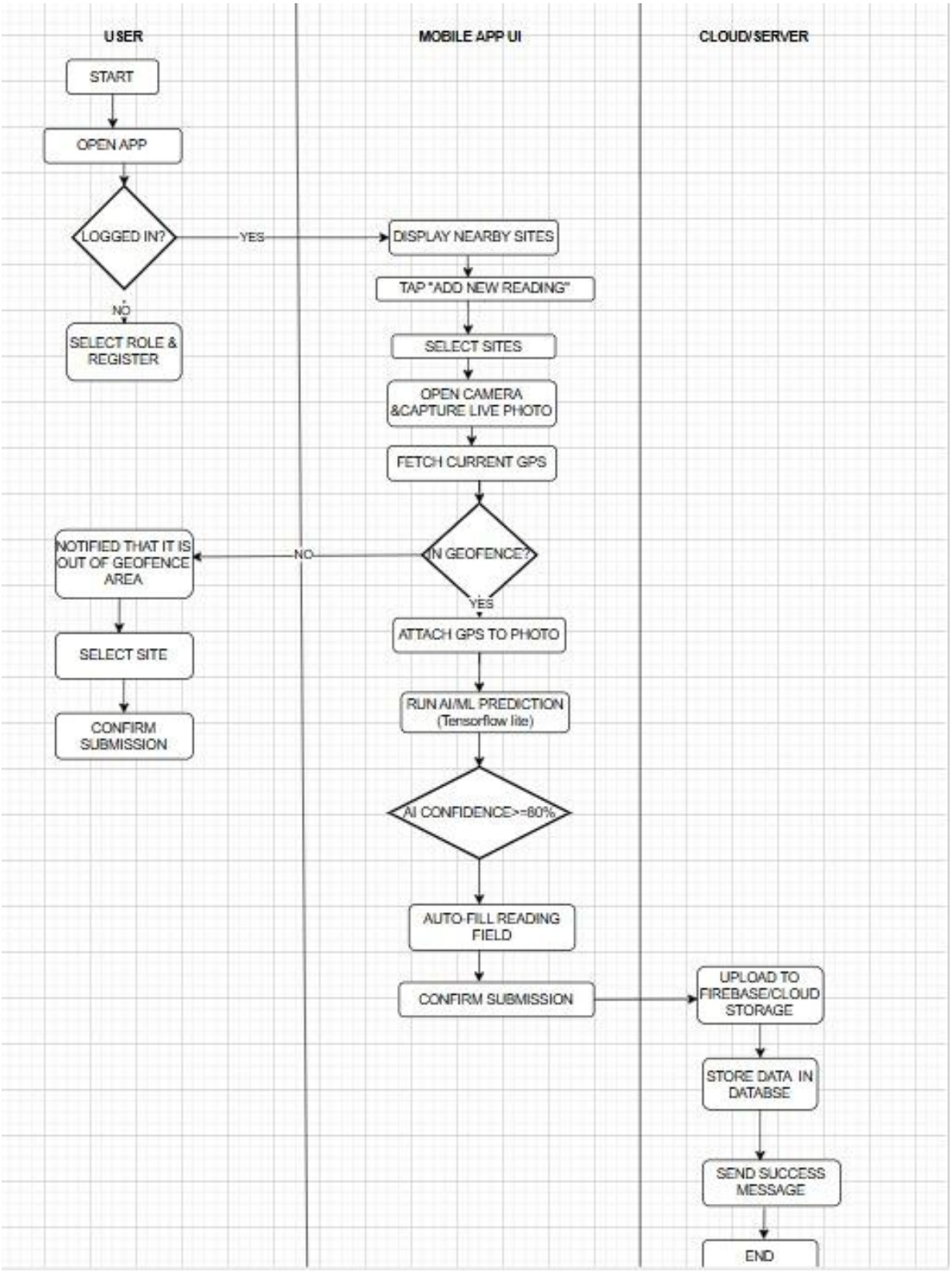


Fig 2.2: shows swimlane diagram

2.3 Data Flow Diagrams (DFDs)

2.3.1 DFD Level 0

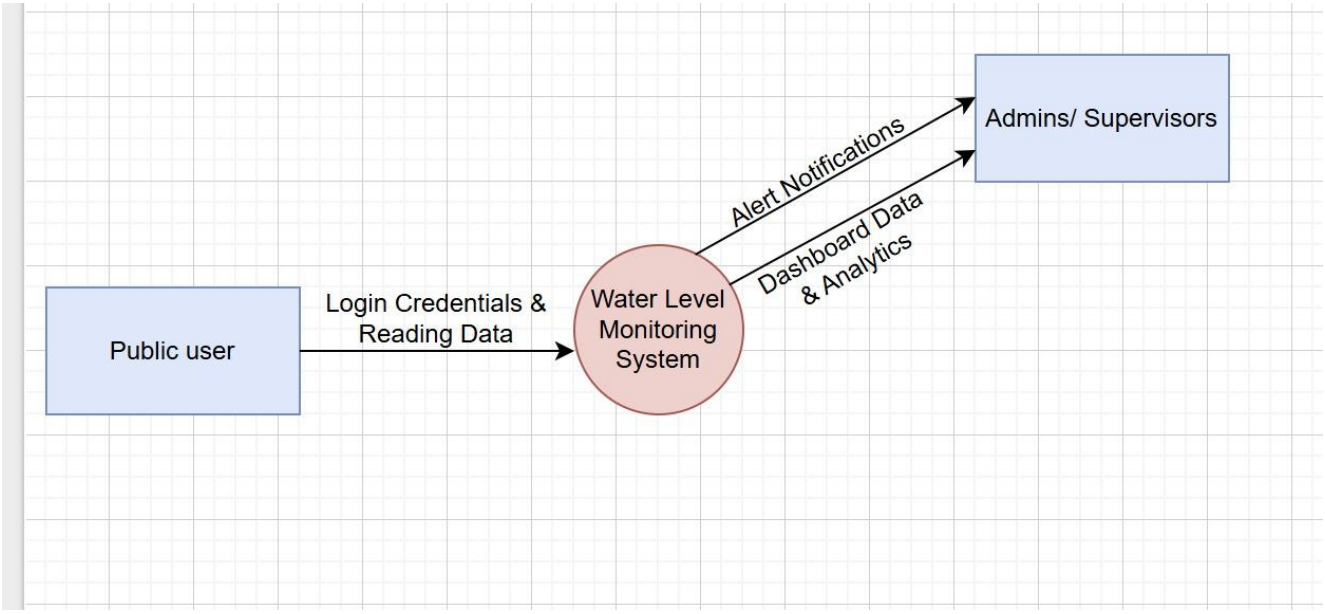


Fig 2.3.1: shows dfd level 0 diagram

2.3.2 DFD Level 1

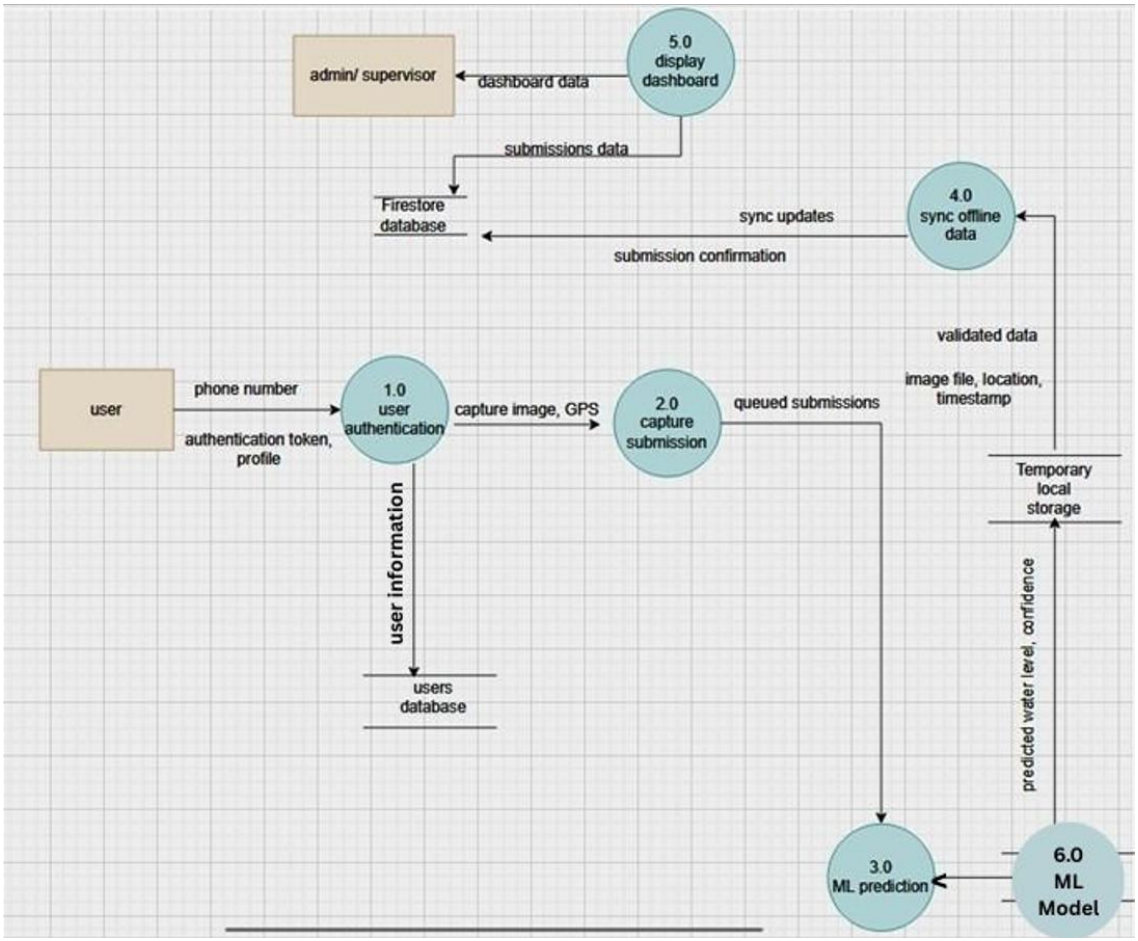


Fig 2.3.2: shows dfd level 1 diagram

2.3.2 DFD Level 2

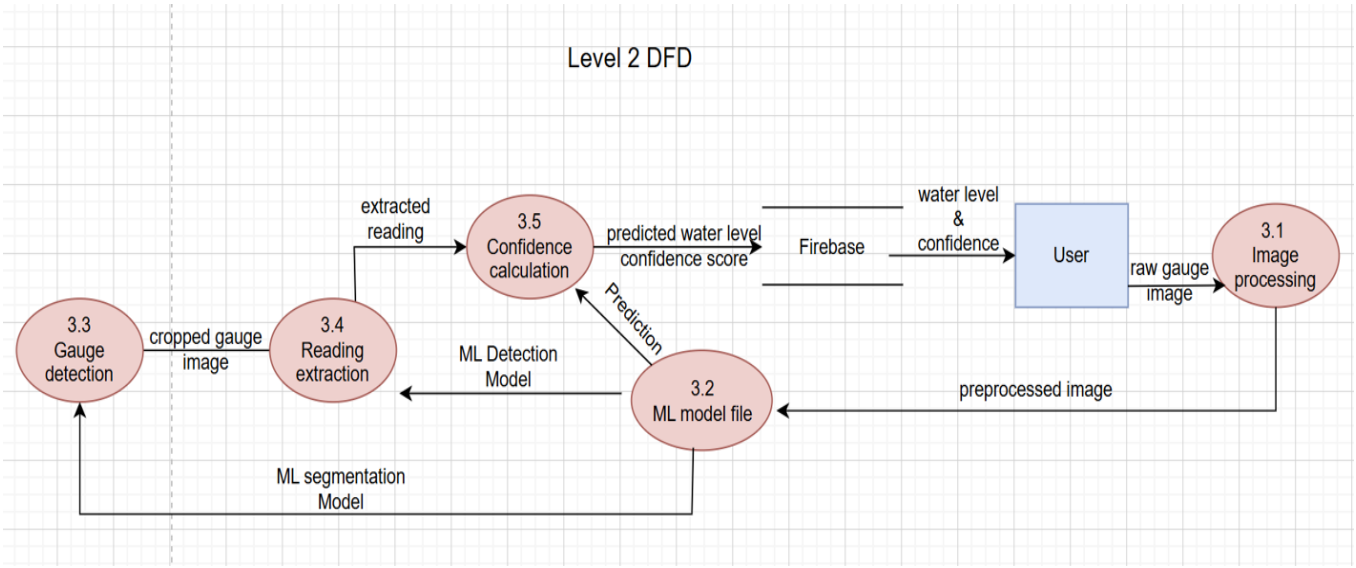


Fig 2.3.3: shows dfd level 2 diagram

2.4 Software Requirement Specification in IEEE Format

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to provide a comprehensive overview of the Water Level Monitoring App project, developed to automate and modernize the manual water level monitoring process used by India's Central Water Commission (CWC). It defines the problem, proposed solution, and complete system workflow—from user authentication and GPS-based site validation to AI-powered image analysis and data visualization. The document serves as a technical and functional guide for developers, testers, and stakeholders, detailing the architecture, modules, data flow, class and use case diagrams, and software requirements to ensure accurate, efficient, and transparent water level reporting.

1.2 Scope of the Development Project

The Water Level Monitoring App project aims to revolutionize the way river and reservoir water levels are measured, reported, and analyzed in India. Its scope encompasses the complete development, deployment, and maintenance of an intelligent, automated, and user-friendly system for both government agencies and the public. By integrating mobile technology, cloud computing, GPS, and AI-based image recognition, the project seeks to replace outdated manual processes with a reliable, transparent, and real-time monitoring solution.

1.2.1 Functional Scope

The system's primary function is to enable real-time submission and verification of water level readings from river gauge posts. The mobile application will serve as the main interface for field personnel and common users to capture live images of gauge posts, automatically detect GPS

location, and record water level readings using an integrated TensorFlow Lite (TFLite) AI model. The model will read water levels directly from photos, minimizing human error and manipulation.

Each submission will include essential metadata such as photo proof, GPS coordinates, timestamp, and user details, ensuring data authenticity. In case of poor connectivity, the app will store submissions offline and automatically sync them once internet access is restored. The backend, powered by Firebase Firestore (Cloud Storage), will securely manage all data, ensuring scalability and real-time availability.

A web-based dashboard will be developed for supervisors and administrators to visualize submissions, verify readings, and analyze historical data. This dashboard will include maps for geographic visualization, charts for trend analysis, and management panels for user control and alert configuration.

1.2.2. User Scope

The system will serve multiple user categories:

Common People: Contribute crowd-sourced readings from nearby rivers or lakes.

Field Personnel (CWC workers): Submit official water level readings at designated sites.

Supervisors: Oversee multiple sites and validate field submissions.

Admins: Manage all users, sites, and system configurations while analyzing national-level data trends.

Each user role will have defined permissions and functionalities, controlled by Firebase Authentication and role-based access rules.

1.2.3. Technical Scope

The development involves:

Android Mobile App: Built with Kotlin or Flutter, integrating Firebase, GPS, Camera API, and TensorFlow Lite.

AI Module: CNN-based model trained to recognize water levels from gauge images and integrated into the mobile app for offline inference.

Cloud Backend: Firebase Firestore for NoSQL data storage, Cloud Storage for image hosting, and Firebase Functions for automation.

Web Dashboard: Developed using React.js or Flutter Web, featuring real-time analytics, site management, and alert systems.

Offline Functionality: Implemented via Room Database and WorkManager to ensure reliability in remote areas.

1.2.4. Implementation Scope

The project includes all phases: requirement analysis, design, development, testing, deployment, and maintenance. The system will be piloted across selected CWC sites before nationwide scaling. Future expansions may include integrating IoT water sensors, flood prediction models, and multi-language support for wider accessibility.

1.2.5. Out-of-Scope

The project does not include hardware sensor development, server hosting outside Firebase infrastructure, or integration with third-party government systems during the initial phase.

1.2.6. Expected Outcome

Upon completion, the project will deliver a scalable, secure, and AI-powered monitoring system that ensures accurate data collection, timely reporting, and data-driven decision-making for flood management and water resource planning across India.

1.3 Definitions, abbreviations and acronyms

Definitions

Table 1 gives explanation of the most commonly used terms in this SRS document.

Table 1: Definitions for most commonly used terms

S.No	Term	Definition
1	Convolutional Neural Network	A deep learning algorithm used for image-based recognition of gauge post readings.
2	Firebase Firestore Database	A NoSQL cloud database used to store user data, submissions, and site information.
3	Application Programming Interface	A set of functions that enable communication between software components (e.g., Google Maps API).
4	TensorFlow Lite	A lightweight version of TensorFlow that allows ML models to run efficiently on mobile devices.
5	Central Water Commission	The Indian government authority responsible for river monitoring and flood management.
6	Global Positioning System	Satellite-based navigation system used to determine the user's precise location

7	User Interface	The visual part of the app through which users interact (buttons, forms, maps, etc.).
8	User Experience	Overall experience and satisfaction users get while using the app.
9	Cloud Storage	Firebase component used to securely store uploaded images in the cloud.
10	React.js	JavaScript library used to develop the web dashboard for admins and supervisors
11	Not Only SQL	A database model that stores data in key-value or document form instead of traditional tables.

Abbreviations

Table 2 gives the full form of most commonly used mnemonics in this SRS document.

Table 2: Full form for most commonly used mnemonics

S.No	Mnemonic	Full Form
1	API	Application Programming Interface
2	NoSQL	Not Only SQL
3	GPS	Global Positioning System
4	CNN	Convolutional Neural Network
5	CWC	Central Water Commission
6	TFLite	TensorFlow Lite
7	UI	User Interface
8	UX	User Experience

1.4 References

1. Google Firebase Documentation. Firebase Authentication, Firestore, and Cloud Storage Guides. https://firebase.google.com/docs
2. TensorFlow Lite Documentation. TensorFlow Lite Guide. https://www.tensorflow.org/lite
3. Google Maps Platform Documentation. Maps SDK for Android and JavaScript API. Available at: https://developers.google.com/maps
4. Android Developers. CameraX and Location Services APIs. Available at: https://developer.android.com
5. Central Water Commission (CWC). Water Level and Flood Forecasting Reports. Government of India, Ministry of Jal Shakti. https://cwc.gov.in

6. Jain, A., et al. (2023). AI-Based Flood Monitoring and Water Level Detection Using Image Processing Techniques. International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE).
7. Patil, S., & Deshmukh, R. (2022). IoT-Based River Water Level Monitoring System. IEEE International Conference on Smart Technologies for Power, Energy, and Control (STPEC).
8. Kaur, G., & Sharma, D. (2021). Real-Time Flood Detection Using Machine Learning and Cloud Computing. International Journal of Innovative Research in Computer and Communication Engineering.
9. Google Developers. WorkManager and Room Persistence Library Documentation. https://developer.android.com/topic/libraries/architecture
10. Chart.js Documentation. Data Visualization for Web Dashboards. https://www.chartjs.org/docs/latest/
11. Singh, R., & Gupta, M. (2020). Design and Implementation of Mobile Applications for Environmental Data Collection. Springer Lecture Notes in Networks and Systems.
12. World Meteorological Organization (WMO). Manual on Stream Gauging (Vol. I & II). Geneva: WMO Publication.

1.5 Overview

The Water Level Monitoring App is an intelligent mobile and web-based system designed to automate river water level monitoring for India's Central Water Commission (CWC). It replaces the manual process of field data collection with a digital solution that uses GPS, camera input, and AI-based image recognition to detect water levels from gauge post photos. Integrated with Firebase for secure data storage and real-time synchronization, it enables accurate, verifiable, and timely reporting. The system includes role-based access for users, field staff, supervisors, and admins, offering data visualization dashboards and alert systems to support effective flood management and decision-making.

2. Overall Description

2.1 Product Perspective

The Water Level Monitoring App is a comprehensive digital system developed to modernize and automate the process of river and reservoir water level monitoring in India. It is conceptualized to directly support the operations of the Central Water Commission (CWC) by replacing outdated manual methods with a real-time, AI-driven, and cloud-based solution. This product is not a standalone tool but a connected ecosystem combining mobile technology, artificial intelligence, GPS services, and cloud databases to create an intelligent, transparent, and scalable monitoring infrastructure.

At its core, the product consists of two major components: the mobile application and the web-based dashboard. The mobile app serves as the primary data collection interface for various users, including field personnel, supervisors, and even common citizens. It allows users to log in securely through Firebase Authentication using OTP-based mobile verification. Once authenticated, users can capture live images of water gauge posts, which are automatically tagged with GPS coordinates and timestamps. The application uses Google Maps API and geofencing to ensure that submissions

are made from within an approved distance (typically 100–200 meters) of the monitoring site, preventing false entries.

One of the most innovative aspects of the product is the integration of Artificial Intelligence (AI) through a TensorFlow Lite (TFLite) model embedded directly in the mobile app. This model analyzes the captured image to identify gauge markings and automatically determine the water level. If the AI's confidence level exceeds a certain threshold (e.g., 80%), the reading is auto-filled; otherwise, the user is prompted to manually verify or correct it. This ensures a blend of automation and human oversight, improving both accuracy and reliability.

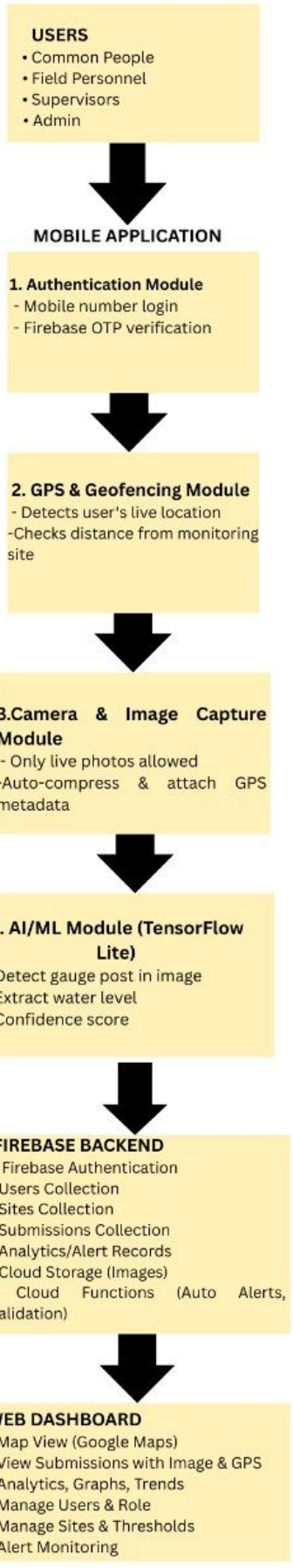
The system's backend infrastructure is powered by Firebase Firestore (NoSQL database) and Firebase Cloud Storage, which together manage all data in real time. Firestore stores user details, site information, and submission metadata, while Cloud Storage holds image files securely. Each submission record includes key attributes such as user ID, role, site ID, timestamp, coordinates, image URL, predicted water level, and manual correction (if any). Firebase Cloud Functions are used to automate processes like sending alerts, synchronizing offline submissions, and generating notifications for supervisors when water levels exceed defined thresholds.

The web dashboard forms the administrative and analytical component of the system. It provides supervisors and administrators with a real-time overview of all monitoring sites via an interactive Google Map interface. Markers on the map display recent readings, color-coded by status — for example, green for normal, yellow for outdated, and red for critical levels. The dashboard also includes powerful data analytics features, enabling users to view trends, generate charts, and compare site-wise or user-wise statistics using tools like Chart.js. Admins can manage users, assign field sites, set alert levels, and export reports directly from this portal.

The app is also designed for offline functionality. In regions with poor connectivity, data is temporarily stored in a Room Database on the device and automatically synced once an internet connection becomes available, managed by Work Manager in the background.

In summary, the Water Level Monitoring App represents a technologically advanced and socially impactful product. It integrates multiple modern technologies—mobile computing, AI, cloud infrastructure, and geospatial intelligence—to address real-world challenges in environmental monitoring. It provides an end-to-end solution that enhances data accuracy, accelerates reporting, and strengthens the nation's capacity for flood prediction, disaster preparedness, and water resource management.

Block diagram:-



2.2 Product Functions

The Water Level Monitoring App offers an intelligent, automated system to track river and reservoir water levels, ensuring accurate, real-time data for the Central Water Commission (CWC) and other stakeholders through mobile, backend, and web platforms.

1. User Authentication & Roles:
   Secure login via Firebase OTP. Users have defined roles — Common Users submit readings, Field Staff verify data, Supervisors approve submissions, and Admins manage users, sites, and reports.

2. GPS Verification & Geofencing:
   GPS ensures submissions occur at the correct site. Geofencing (100–200 m radius) prevents false or remote uploads.

3. AI-Based Detection:
   An on-device TFLite model analyzes water gauge images to auto-read water levels. Low-confidence readings can be manually corrected for accuracy.

4. Cloud Data Storage:
   Validated readings, images, and GPS data are stored in Firebase Firestore and Cloud Storage, with offline sync using WorkManager.

5. Real-Time Monitoring:
   A web dashboard provides live maps, color-coded water level indicators, and trend charts for continuous monitoring.

6. Alerts & Notifications:
   Threshold-based alerts via Firebase Cloud Messaging notify supervisors and admins during critical water level events.

7. Reports & Analytics:
   Admins generate CSV/PDF reports with analytics like trends, averages, and historical comparisons for data-driven decisions.

8. System Administration:
   Admins configure user roles, thresholds, geofences, and update AI models to keep the system accurate and efficient.

In essence, the app unites AI, GPS, cloud, and analytics to provide a smart, reliable, and responsive solution for flood prevention and water management.

2.3 User Characteristics

The Water Level Monitoring App serves multiple user groups in water resource management, each with defined roles, permissions, and expertise levels to ensure a secure and efficient system.

2.3.1. Common Users (Public Contributors):

Citizens who submit nearby water gauge images to support community-based monitoring.

- Skills: Basic smartphone use.
- Access: Upload images, add remarks, view submission status.
- Needs: Simple, guided UI with visual feedback.

2.3.2. Field Staff (CWC Ground Team):

Official CWC employees collecting site data.

- Skills: Familiar with GPS and data entry.
- Access: Capture/upload images, verify AI readings, manage assigned sites.
- Needs: Offline support, GPS validation, and error handling.

2.3.3. Supervisors (Regional Officers):

Oversee multiple sites and field staff.

- Skills: Comfortable with dashboards and analytics.
- Access: Review, approve/reject data, receive alerts.
- Needs: Real-time dashboard with maps, charts, and filtering tools.

2.3.4. Administrators (CWC Authorities):

Top-level users managing the system and data insights.

- Skills: Proficient in analytics and system management.
- Access: Full control over users, sites, thresholds, and reports.
- Needs: Web dashboard with analytics, report generation, and configuration options**.**

2.3.5. Developers & System Maintainers (Technical Team):

IT professionals handling updates, AI models, and backend maintenance.

- Skills: Expertise in Flutter, Firebase, and AI.
- Access: Backend privileges for debugging, deployment, and optimization.
- Needs: Secure system access for updates and monitoring.

2.4 General Constraints, Assumptions and Dependencies

The Water Level Monitoring App operates within a set of technical, environmental, and organizational constraints that influence its performance and functionality. Understanding these constraints, assumptions, and dependencies ensures the system is realistic, reliable, and adaptable to field conditions.

<u>General Constraints:</u>

Network Availability:

Continuous internet connectivity may not be available in remote riverine regions. The app must, therefore, support offline data collection with background synchronization once connectivity is restored.

Device Compatibility:
The mobile app is designed primarily for Android devices (Android 8.0 and above). Performance may vary across devices with different camera qualities, screen resolutions, or hardware specifications.

Data Accuracy:
The AI model's accuracy in reading gauge levels depends on image clarity, lighting, and environmental conditions such as rain, fog, or reflection on the water surface.

Storage Limitations:
Firebase's free tier imposes limits on data storage and request rates; exceeding them may require paid plans for large-scale deployments.

Regulatory Compliance:
The system must comply with government data privacy and security standards, ensuring no unauthorized access or misuse of environmental data.

Assumptions:

Users (especially field staff) are trained in capturing clear and correctly framed gauge images.

All registered sites have properly maintained gauge posts with visible markings.

Supervisors and administrators have regular access to the web dashboard for timely data validation and review.

Devices used for data capture have functional GPS, camera, and sufficient storage.

Dependencies:

External APIs: The system depends on Google Maps API for geolocation and mapping services.

Cloud Services: Firebase Authentication, Firestore, and Cloud Storage are essential for real-time data handling.

AI Model Updates: Continued accuracy depends on periodic retraining and updates of the TensorFlow Lite model.

Government Coordination: Effective functioning depends on cooperation between the CWC, regional offices, and field staff for smooth operational workflow.

2.5 Apportioning of requirements

The Water Level Monitoring App project involves multiple functional and non-functional requirements that can be apportioned based on priority, complexity, and implementation phase. Given the system's critical role in flood monitoring and water management, certain requirements are prioritized for the initial release, while others may be phased in later.

High-Priority Requirements: These are essential for basic system functionality:

User authentication via OTP and role-based access.

GPS-based site validation and geofencing for accurate location verification.

Live image capture of gauge posts and AI-based water level detection.

Cloud storage of submissions and real-time synchronization with Firebase.

Web dashboard for supervisors and admins to view submissions and alerts.

Medium-Priority Requirements: Features that enhance usability and reliability but can be implemented incrementally:

Offline submission support with local database storage and background synchronization.

Push notifications for threshold alerts and reminders to field personnel.

Data analytics features like trend graphs, historical reports, and statistics.

Low-Priority / Future Enhancements: These can be phased in after deployment to improve scalability and system intelligence:

Multi-language support for mobile and web interfaces.

Integration with IoT sensors for automated water level measurements.

Advanced AI improvements such as improved recognition under poor lighting conditions.

This apportioning ensures that the system delivers core functionality first, while providing a roadmap for future enhancements without disrupting ongoing operations.

3. Specific Requirements

The Water Level Monitoring App has clearly defined requirements to ensure accurate, reliable, and secure water level monitoring. These include external interface requirements, functional requirements, and system constraints to guide development and implementation.

3.1 External Interface Requirements

The system interacts with several external interfaces, which are crucial for its operation and usability:

3.1.1. User Interface:

Mobile App: Android-based app providing registration, login, GPS location tracking, camera capture, AI prediction display, and submission features.

3.1.2. Hardware Interfaces:

Mobile Device: Requires camera, GPS module, and internet connectivity. Minimum 2GB RAM and Android 8.0+.

3.1.3. Software Interfaces:

Firebase: For authentication, Firestore database storage, Cloud Storage, and backend synchronization.

Google Maps API: Provides geolocation, site mapping, and geofencing functionality.

TensorFlow Lite: Runs AI models locally on mobile devices for gauge reading extraction.

3.1.4. Communication Interfaces:

HTTPS protocols for secure data transfer.

Push notifications via Firebase Cloud Messaging for alerts and reminders.

These external interfaces ensure the system integrates seamlessly with devices, APIs, and users while maintaining security, real-time performance, and reliability

3.2 Detailed Description of Functional Requirements

3.2.1 Functional Requirement 1: User Registration and Authentication

Purpose	To securely register users, verify identity, and assign appropriate roles (Common User, Field Staff, Supervisor, Admin) to control access.
Inputs	User mobile number; OTP sent via Firebase; role selection; optional personal details (name, email).
Processing	OTP is verified against Firebase; checks for duplicate registrations; role assignment is validated; session is created for authenticated users. Handles invalid OTP entries, duplicate numbers, or unapproved role requests.
Outputs	Successful account creation confirmation; session token; error messages for invalid OTP, duplicate registration, or unapproved role; access rights set according to role.

1.1.1 Functional Requirement 2: GPS Location and Geofencing

Purpose	To ensure water level readings are submitted from the correct physical location
Inputs	Device GPS coordinates; selected monitoring site coordinates from database.
Processing	Distance between user and site calculated; submission allowed if within 100–200 meters. If outside, warning displayed but limited submission may be allowed. Handles GPS unavailability or inaccurate readings
Outputs	Geofence validation status; warning messages for out-of-range submissions; logs of location metadata for each submission.

1.1.2 Functional Requirement 3: Image Capture and Processing

Purpose	To capture live gauge images and prepare them for AI-based water level detection.
Inputs	Live image from device camera; device GPS coordinates; timestamp
Processing	Camera interface opened; image compressed (<500KB); EXIF metadata extracted; image processed for AI(resizing, filtering, cropping). Handles low-quality images or camera errors.
Outputs	Preprocessed image ready for AI prediction; error messages if camera fails or image is invalid; confirmation of successful capture.

1.1.3 Functional Requirement 4: AI-Based Water Level Detection

Purpose	To automatically read water level from gauge images to reduce manual errors.
Inputs	Preprocessed gauge image; trained TensorFlow Lite AI model.
Processing	Stage 1: Gauge detection in image. Stage 2: Digit recognition to predict water level; confidence score calculated. If confidence <80%, prompts manual verification. Handles poor lighting or unclear images.
Outputs	Predicted water level; confidence score; auto-filled or manually verified readings; error prompts for low-confidence or failed detection.

1.1.4 Functional Requirement 5: Data Submission and Cloud Storage

Purpose	To securely store water level readings, images, and metadata in a centralized system.
Inputs	Predicted/verified water level; GPS coordinates; timestamp; image file; user ID; site ID; role.
Processing	Validation checks on completeness of data; uploads to Firebase Firestore and Cloud Storage. Offline storage in Room Database if internet unavailable; queued for background sync. Handles upload failures or incomplete data.
Outputs	Submission confirmation message; updated cloud database; error notifications if upload fails; offline sync status updates.

1.1.5 Functional Requirement 6: Dashboard and Analytics

Purpose	To enable supervisors and admins to monitor submissions, detect trends, and manage alerts.
Inputs	Submission data from Firestore; site coordinates; threshold levels; user filters (date, site, user).
Processing	Data aggregated for real-time map visualization; trend graphs generated; colorcoded markers applied; alerts triggered if thresholds exceeded. Handles missing data, delayed submissions, or invalid entries.
Outputs	Interactive map with markers; line/bar charts showing water level trends; statistical summaries; alerts and notifications; exportable reports.

1.2 Performance Requirements

The Water Level Monitoring App must operate efficiently across a wide range of devices and network conditions to ensure timely and accurate water level reporting. The system shall provide near-instant feedback for core user interactions. The app's home screen must load within 3 seconds on a mid-range Android device (2–4 GB RAM, Android 8.0+). Image capture and preprocessing, including compression and EXIF extraction, shall complete within 2–3 seconds to prevent user delays. Machine Learning inference using the integrated TensorFlow Lite model shall predict water levels from gauge photos within 2 seconds, ensuring smooth workflow for field personnel.

Data submission, including uploading images to Firebase Cloud Storage and writing reading metadata to Firestore, must complete within 10 seconds over a 3G connection; on faster networks, submission shall occur almost instantaneously. Offline mode shall ensure that submissions are cached locally and automatically synced when connectivity is restored, without loss of data or duplication.

The web-based dashboard shall render maps, graphs, and tables within 5 seconds of page load, even when accessing several years of historical data. Filtering and analytics operations, such as trend generation or user-specific submission views, must respond in under 3 seconds to maintain usability for supervisors and admins.

Overall, the system shall handle up to 10,000 concurrent users and process 100,000+ submissions without degradation of responsiveness. All critical performance metrics, including ML inference, data uploads, and dashboard rendering, shall be continuously monitored and optimized to ensure reliable and timely flood monitoring and reporting.

1.3 Logical Database Requirements

1.4 Quality Attributes

The Water Level Monitoring App is built for reliability, speed, and ease of use across various users such as field staff, supervisors, and administrators. It loads quickly, processes AI-based readings efficiently, and syncs data smoothly, even with poor connectivity. The app's intuitive interface supports both technical and non-technical users, while GPS and AI ensure accurate water level detection. Security is maintained through Firebase authentication and role-based access control. The system is scalable, easily maintainable, and compatible with multiple devices, ensuring continuous availability. It also handles errors, invalid inputs, and connectivity issues gracefully, providing a stable and dependable experience.

1.5 Other Requirements

None at this time

2. Change History

2025	Version 1.0- Initial release

3. Document Approvers

SRS for Water Level Monitoring App approved by:

_____

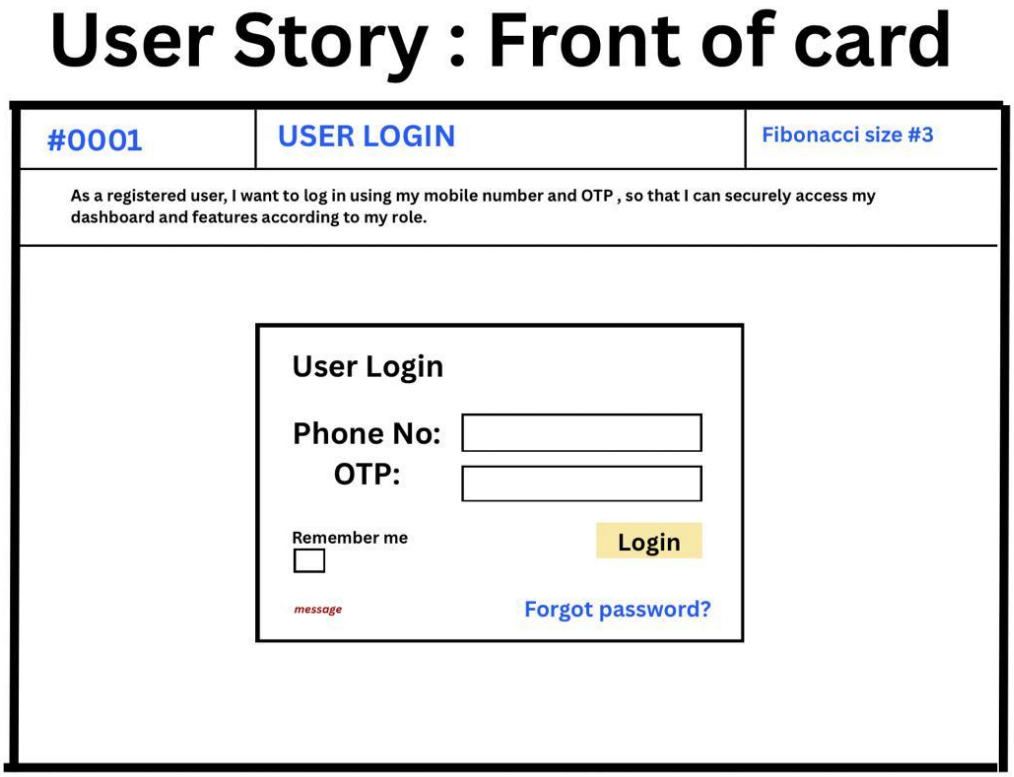
Dr. Deep Mann

Assistant professor

Date: 15/10/25

2.5 User Stories and Story Cards

USER LOGIN



Confirmation

Success Scenarios

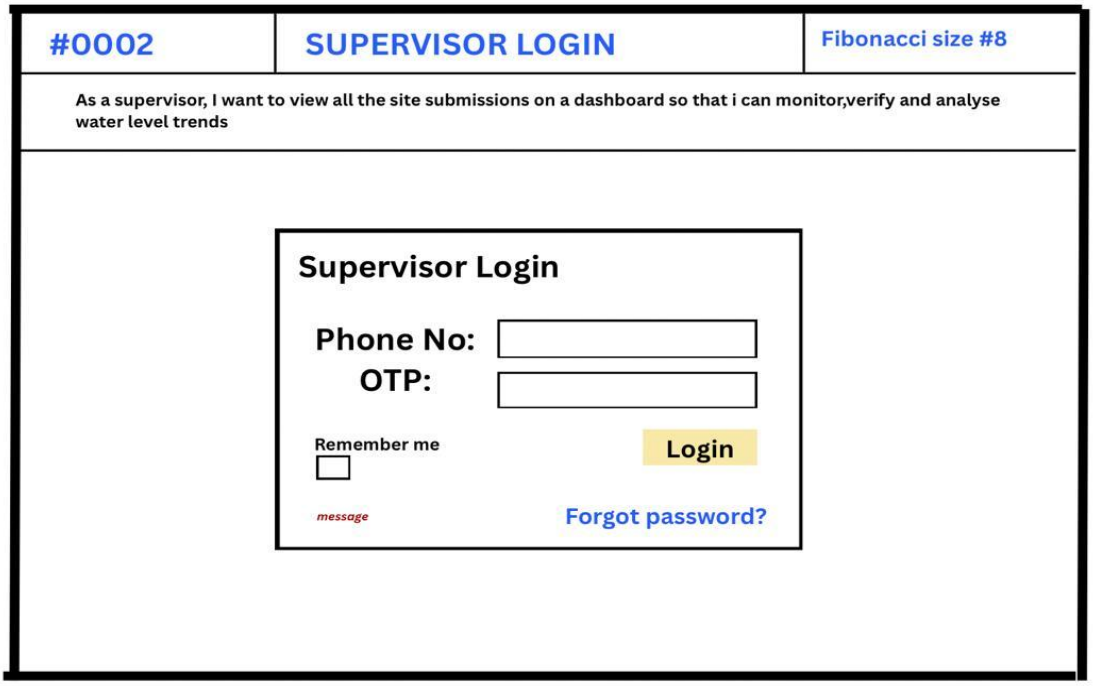
1.Valid OTP entered → Login successful → Redirect to Home
2.User role recognized → Personalized dashboard displayed
3.“Session active” saved → Aut o-login next time

Failure Scenarios

a. “Invalid OTP – please try again”
b. “Network error – unable to verify”
c. “Mobile number not registered”
d. “OTP expired – request new OTP”

SUPERVISOR LOGIN

User Story : Front of card



Confirmation

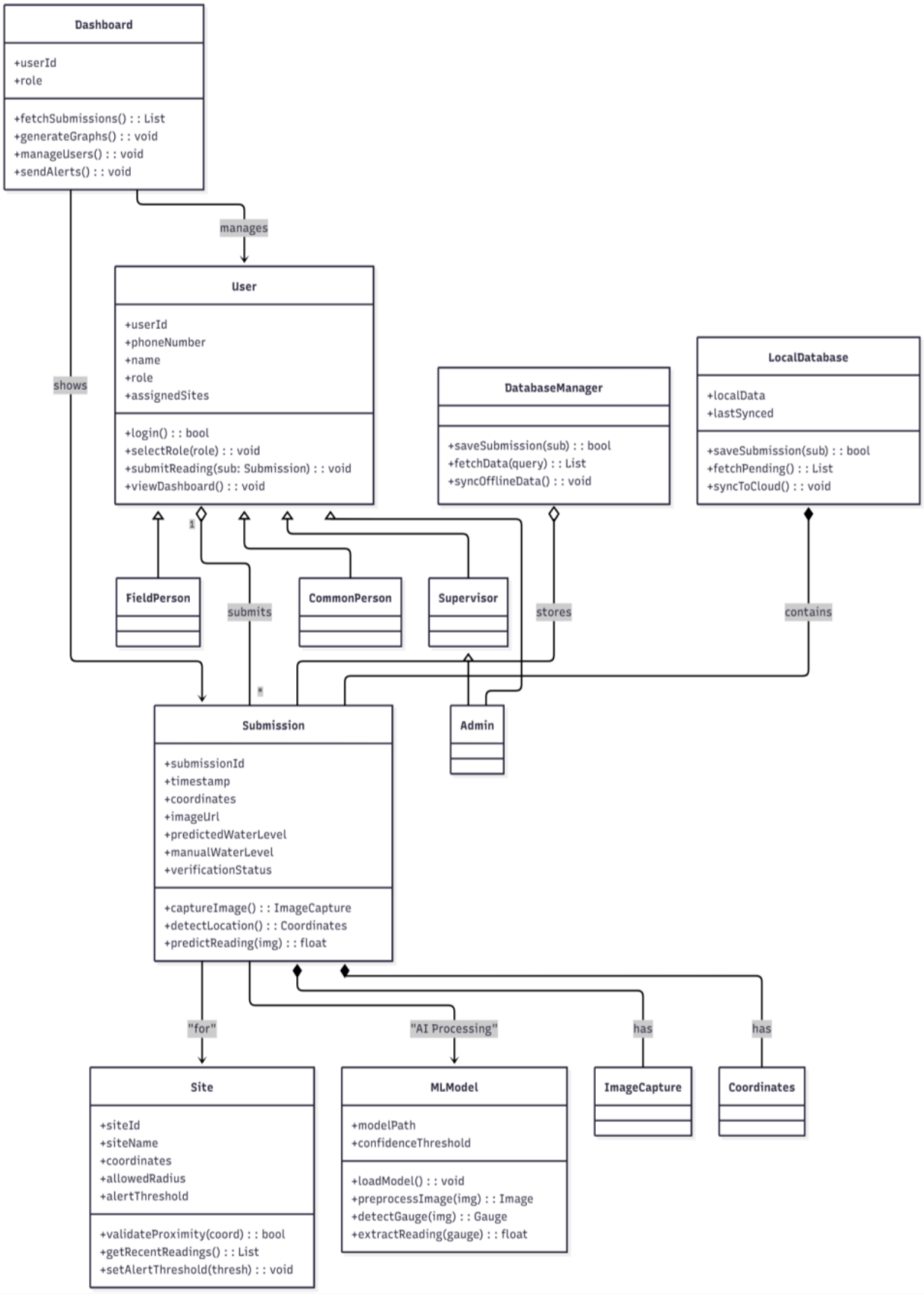
Success Scenarios

a.Dashboard loads successfully → All sites visible
b.Graphs and analytics displayed correctly
c.Alerts shown when threshold exceeded

Failure Scenarios

a. “Data unavailable – please refresh”
b. “Firebase connection lost”
c. “No permissions – restricted access”

3.1 Class Diagram



3.2 Sequence Diagram

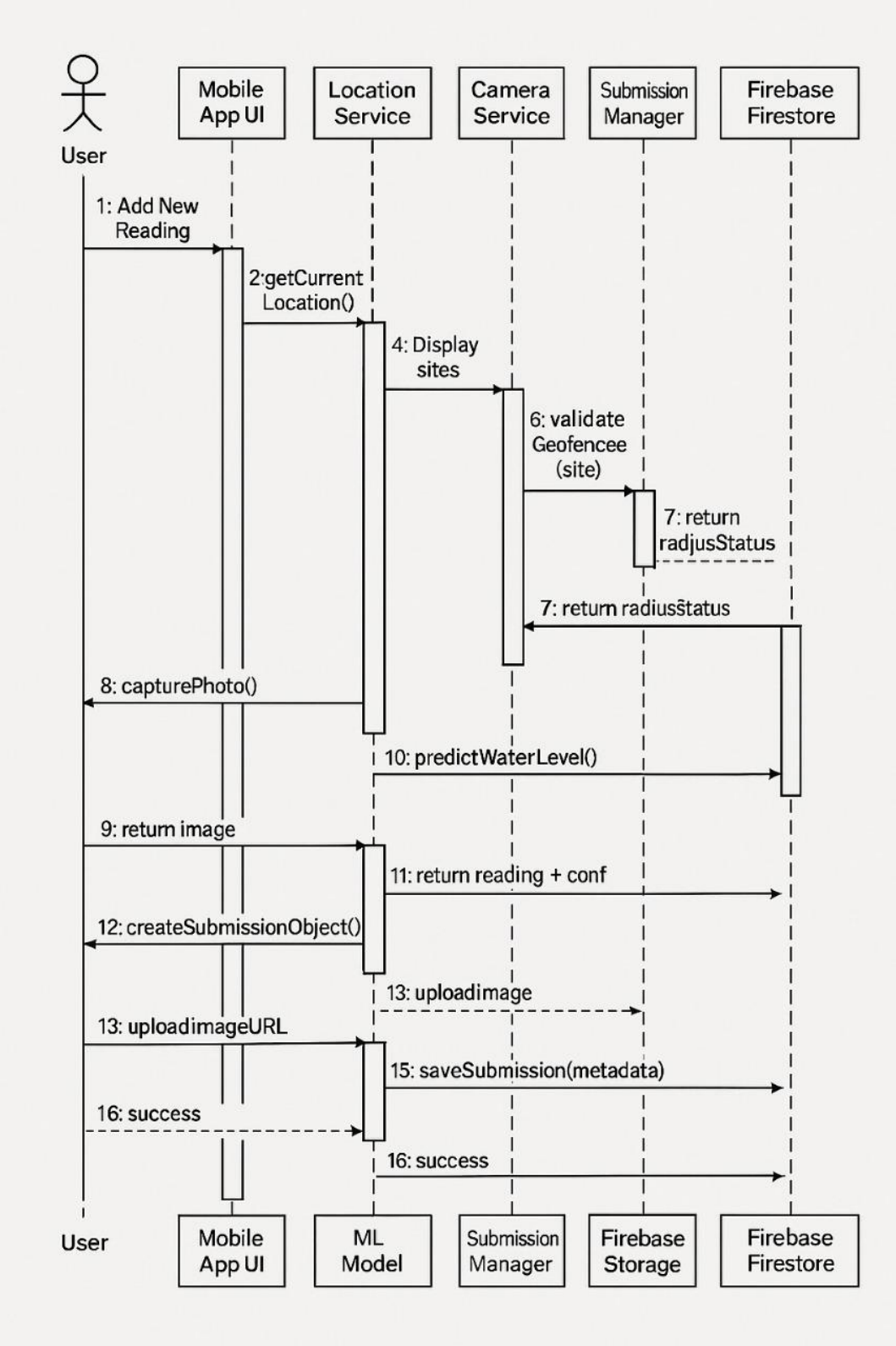


Fig 3.2: shows sequence diagram

3.3 Collaboration Diagram

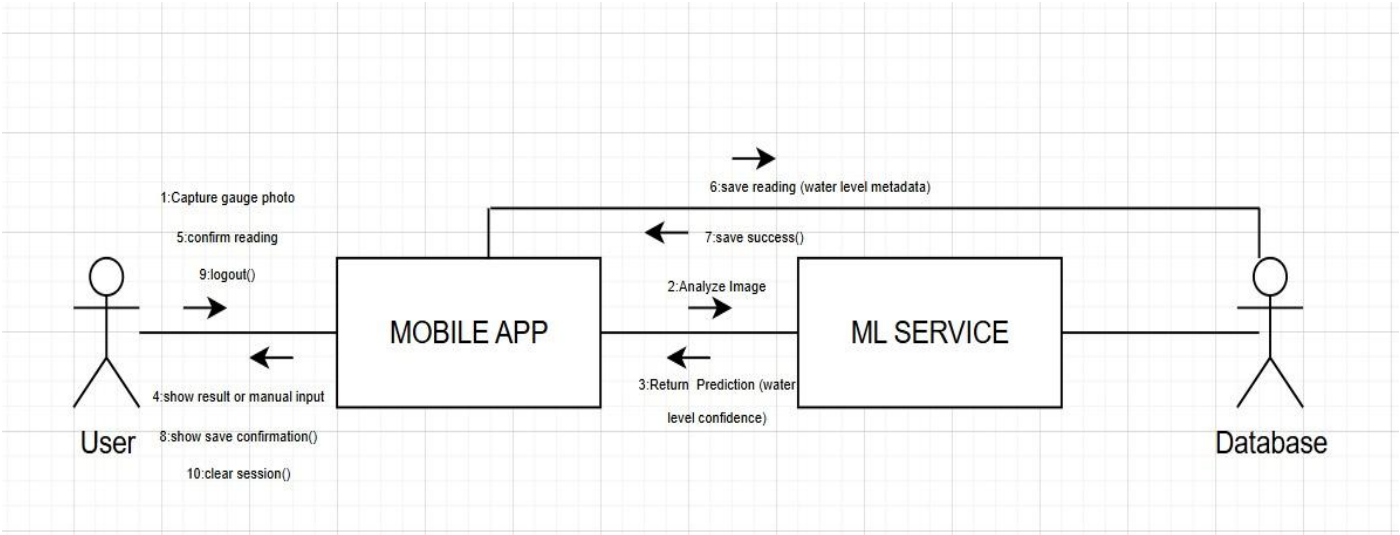


Fig 3.3: shows collaboration diagram

3.4 State Chart Diagrams

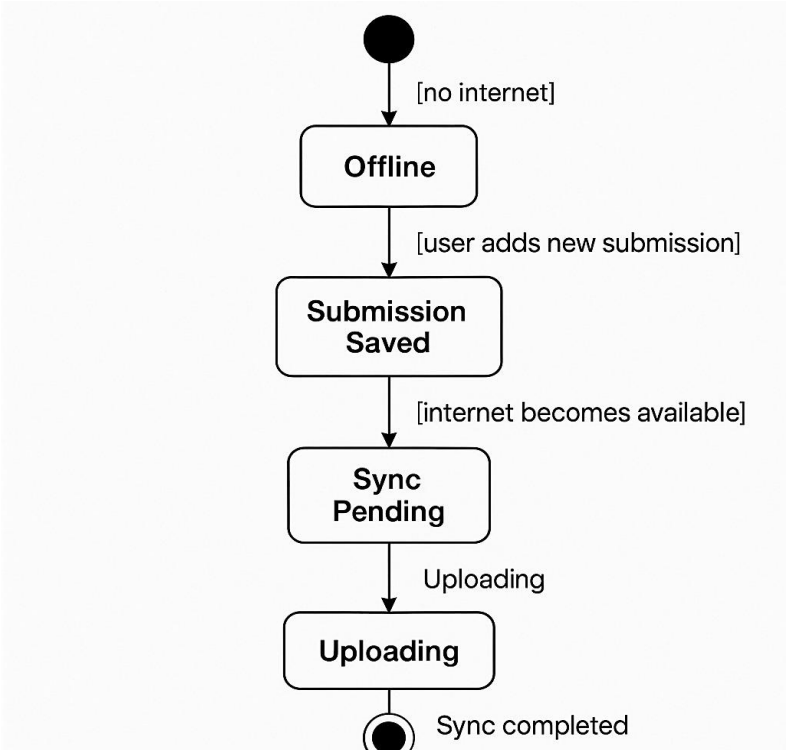


Fig 3.4: shows state chart diagram 1

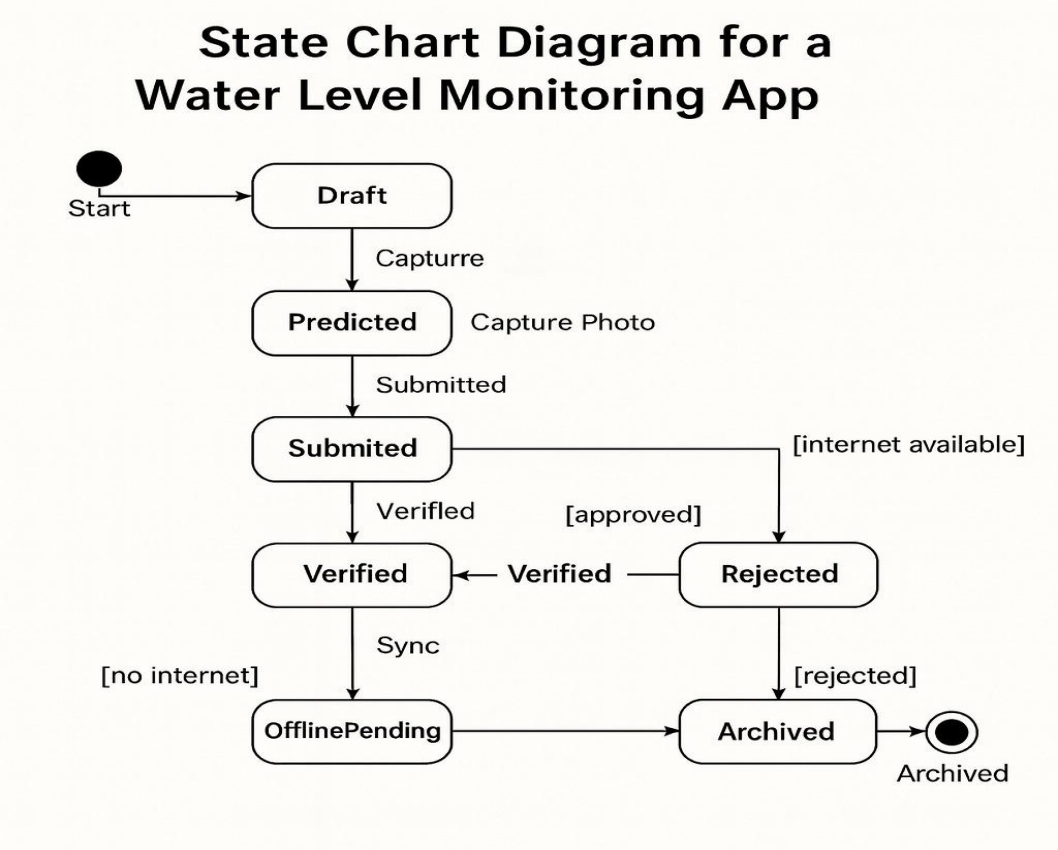


Fig: shows state chart diagram 2

4.Implementation

4.1 Component Diagram

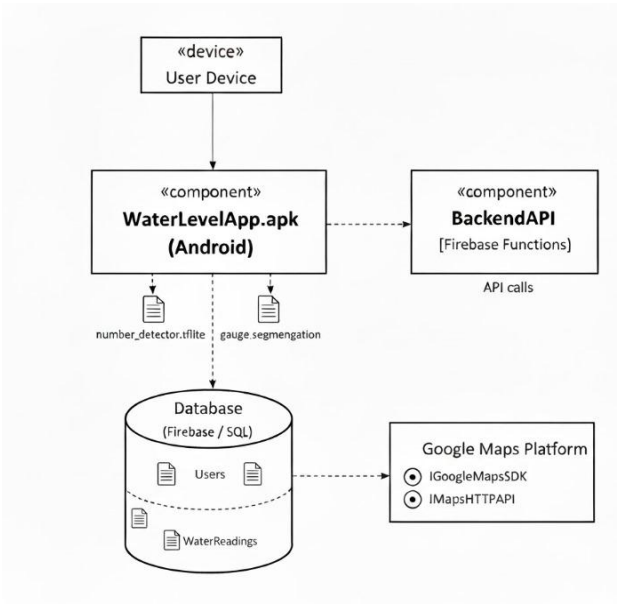


Fig: shows component diagram

4.2 Deployment Diagram

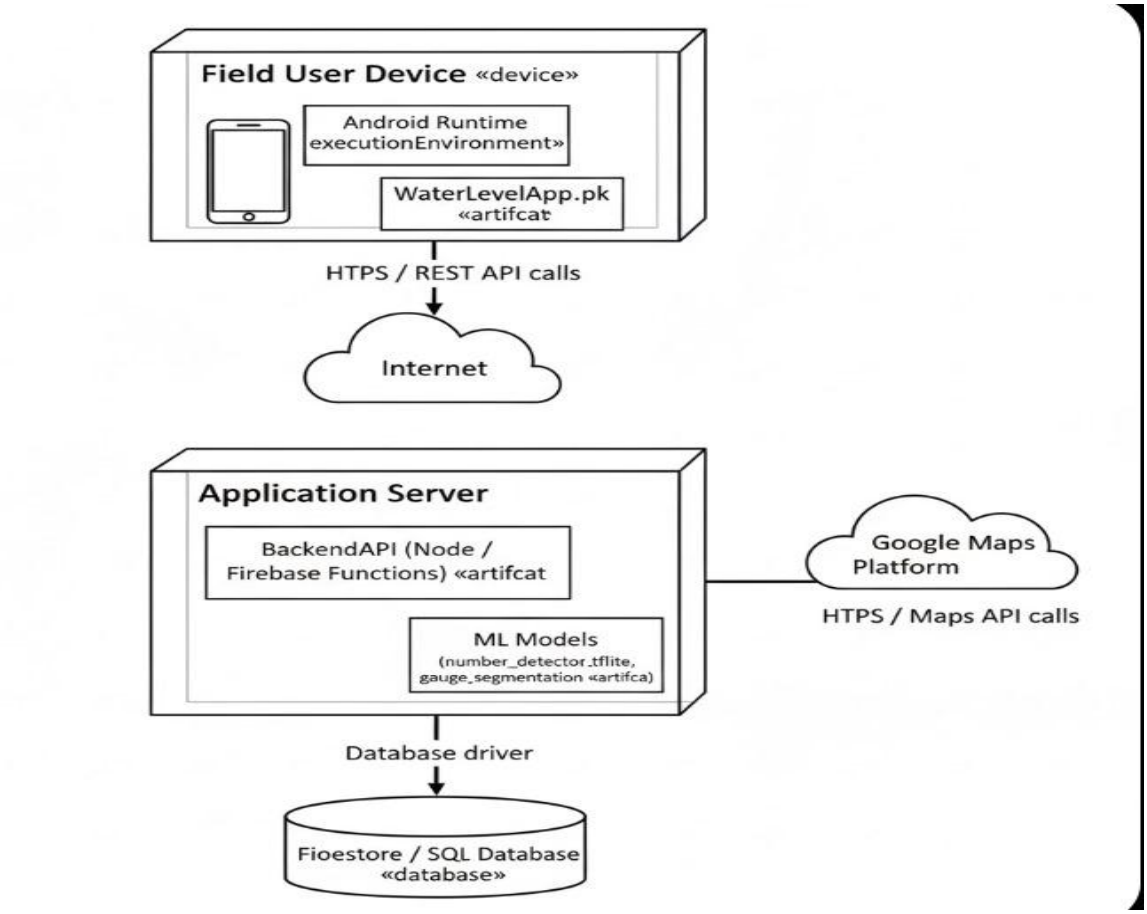


Fig: shows deployment diagram

4.3 Screenshots

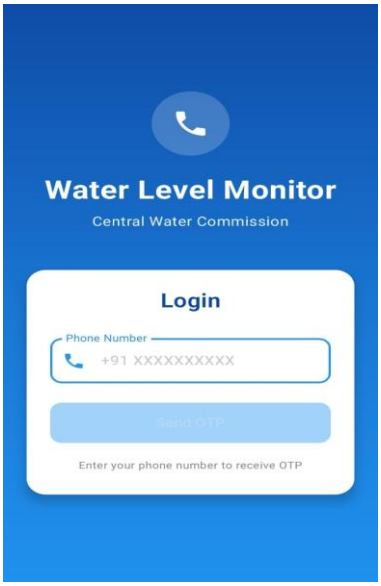


Fig: shows app interface

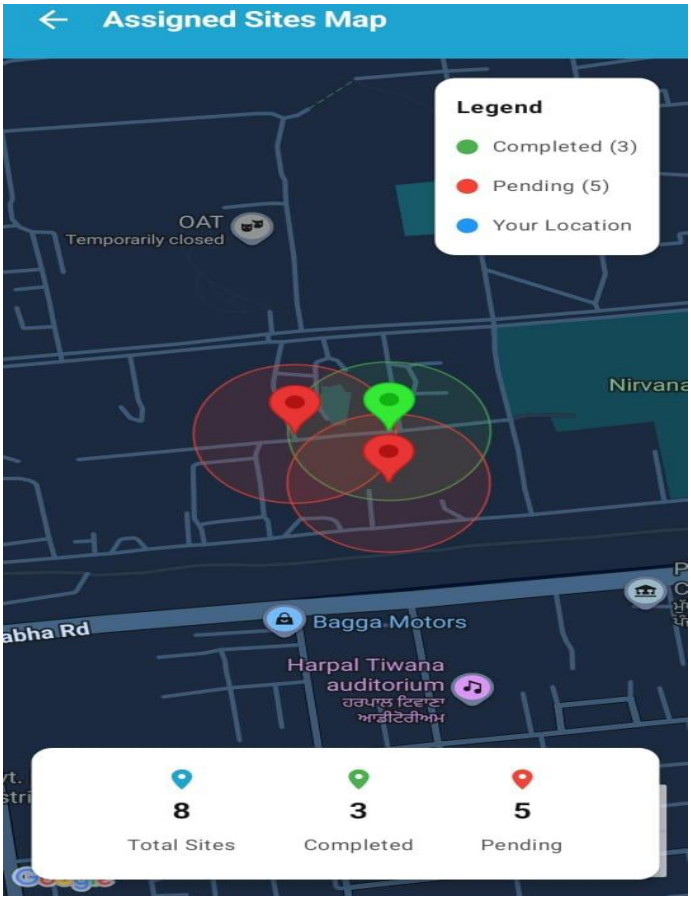


Fig: shows locations on map sing GPS

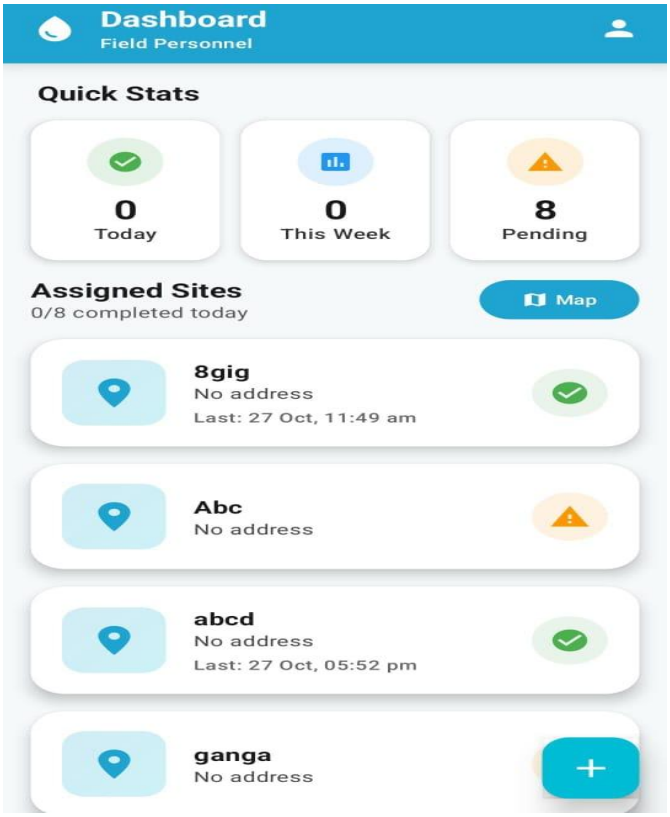


Fig: shows app dashboard

Fig: shows how to calculate water level

5 Testing

5.1 Test Plan

1. Introduction

The purpose of this Test Plan is to define the testing strategy, objectives, scope, resources, schedule, and deliverables for the Water Level Monitoring App.
The system includes an Android app for field users/common users and a web dashboard for supervisors and admins.

2. Objectives

- Verify that all functional requirements are implemented correctly.
- Ensure app stability, reliability, security, and performance.
- Validate Machine Learning predictions for water-level extraction.
- Confirm that geolocation, camera, offline sync, and Firebase services work properly.

- Ensure dashboard analytics and admin functionalities work accurately.

3. Scope of Testing

3.1 In-Scope

- User registration (OTP-based)
- User login
- Role selection & permissions
- GPS tracking & Geofencing
- Camera-based gauge capture
- ML model prediction (TFLite)

3.2 Out-of-Scope

- Backend ML model training
- Network provider OTP delivery delays
- Third-party map rendering (Google Maps)

4. Test Environment

4.1 Hardware

- Android devices (Android 8 to 14)
- Laptops for dashboard testing

4.2 Software

- Android Studio
- Firebase Emulator Suite
- Mock GPS tools
- REST API tools (Postman)

4.3 Network

- 4G/5G, WiFi
- Offline mode simulation

5.2 Test Cases

Test Case Example

Test Case#: 2.2
Test case name: User Login with OTP Verification
System : Water Level Monitoring System
Subsystem: Authentication
Designed by: SACHLEEN
Design date: 17/11/25
Executed by:
Execution date:
Short Description: Verify that user can log in using mobile number and OTP through Firebase.

Pre-conditions
1.App installed on phone
2. User has a valid mobile number
3. Firebase authentication service is active
4. User is logged out (fresh login)

Step	Action	Expected System Response	Pass/fail	Comment
1	Open the app	App displays login screen		
2	Enter mobile number and click "Send OTP"	Firebase sends OTP and app shows OTP entry screen		
3	Enter correct OTP	System verifies OTP and logs the user in		
4	Select user role (Common/ Field/ Supervisor/ Admin)	System saves role and redirects to Home screen		
5	Check post-condition 1			

Post-conditions
1.User session is created and stored locally
2. User role saved in database

Test Case Example

Test Case#: 2.3
Test case name: Geofence validation
System : Water Level Monitoring System
Subsystem: Authentication
Designed by: NANDINI
Design date: 17/11/25
Executed by:
Execution date:
Short Description: Verify taht the app shoes an alert if the user tries to upload a photo outside the geofence

Pre-conditions
1.App installed on phone
2. User has a registered account.
3. Phone gps should be activated

Step	Action	Expected System Response	Pass/fail	Comment
1	user opens photo upload screen	App checks device location	Pass	
2	User outside geofence taps " upload photo "	App displays message"you are outside the allowed are cannot upload photo"	Pass	
3	User attempts to force upload by other means	System displays model reading with no manual prompt	Fail	

Post-conditions
1.No gauge photo is uploaded
2. System log records denied attempt

Test Case Example

Test Case#: 2.4
System : Water Level Monitoring System
Designed by: SWASTIK
Executed by:
Test case name: User Login with OTP Verification
Subsystem: Authentication
Design date: 17/11/25
Execution date:
Short Description: Verify the system allows the user to manually input the water level reading if the ML Model's confidence is below 0.80.

Pre-conditions
1.App installed and logged in
2. ML model is integrated and functional
3. A gauge image is available for processing

Step	Action	Expected System Response	Pass/fail	Comment
1	Select and upload gauge image	App runs ML model on the image	Pass	
2	ML model predicts water level with confidence <0.80	app displays prompt for manual input("unable to determine water level, please enter manually")	Pass	
3	User enters water level manually	System accepts input, saves to database, and confirms entry	Pass	
4	ML model predicts water level with confidence >=0.80	System displays model reading with no manual prompt	Pass	
5	Check post-condition 1		Pass	

Post-conditions
1.If low confidence, user provided reading is saved
2. System log records whether reading was ML or user generated
3. No false readings are auto accepted when confidence is below 0.80

5.3 Test Reports by Peers

Test Case ID	Description	Result
TC_01	Verify that user can log in using mobile number and OTP through Firebase.	PASS
TC_02	Verify that the app shoes an alert if the user tries to upload a photo outside the geofence	PASS
TC_03	Verify the system allows the user to manually input the water level reading if the ML Model's confidence is below 0.80.	PASS