Software Design Specifications

for

Change Detection in Temporal Satellite Imagery for Road Extraction- Version 1.0

Prepared by:

Team RoadAtlas
Design & Development Team

Document Information

Title: Change Detection in Temporal Satellite Imagery for Road Extraction	
Project Manager: Team Lead- Road Atlas	**Document Version No:** 1.0
	Document Version Date: 07-April 2025
Prepared By: Team Road Atlas	**Preparation Date:** 07-April-2025

Version History

Ver.No.	Ver.Date	Revised By	Description	Filename
1.0	07-April-2025	Team RoadAtlas	Initial draft created	RoadAtlas_SDS_v1.docx

Table of Contents

1 Introduction

The Software Design Specifications for Change Detection in Temporal Satellite Imagery for Road Extraction and Mapping describe the architecture, modules, and interactions necessary to detect and map road changes using satellite images from Bhoonidhi. The system utilizes deep learning models, geospatial databases, and a user-friendly interface to automate road monitoring and alerting. This document includes the purpose, scope, definitions, and references essential for understanding the design direction and implementation plan.

1.1 Purpose

The purpose of this document is to define the software design approach for the Road Change Detection System. It aims to support developers, testers, and stakeholders in understanding the structure and goals of the solution. The design elaborates on subsystems including image processing, change detection, alerting, user access control, and GIS data management. The audience includes developers, project managers, academic evaluators, and clients, all of whom will use this document to guide development, validate features, or assess technical feasibility.

1.2 Scope

This Software Design Specification applies to a web-based application that processes satellite imagery to detect and report road changes over time. It handles:

- Retrieval of images from the Bhoonidhi Portal
- Image preprocessing and road feature extraction using Deep Learning
- GIS database integration for storage
- User-specific access to regions
- Real-time alerts to users and authorities on detected changes

The system is designed to aid emergency response, and infrastructure maintenance by automating what is typically a manual, time-consuming process.

1.3 Definitions, Acronyms, and Abbreviations

- AI – Artificial Intelligence
- API – Application Programming Interface
- Bhoonidhi – ISRO's satellite imagery portal
- GIS – Geographic Information System
- ISRO – Indian Space Research Organisation
- PWA – Progressive Web Application
- RBAC – Role-Based Access Control
- SRS – Software Requirements Specification
- 2FA – Two-Factor Authentication

1.4 References

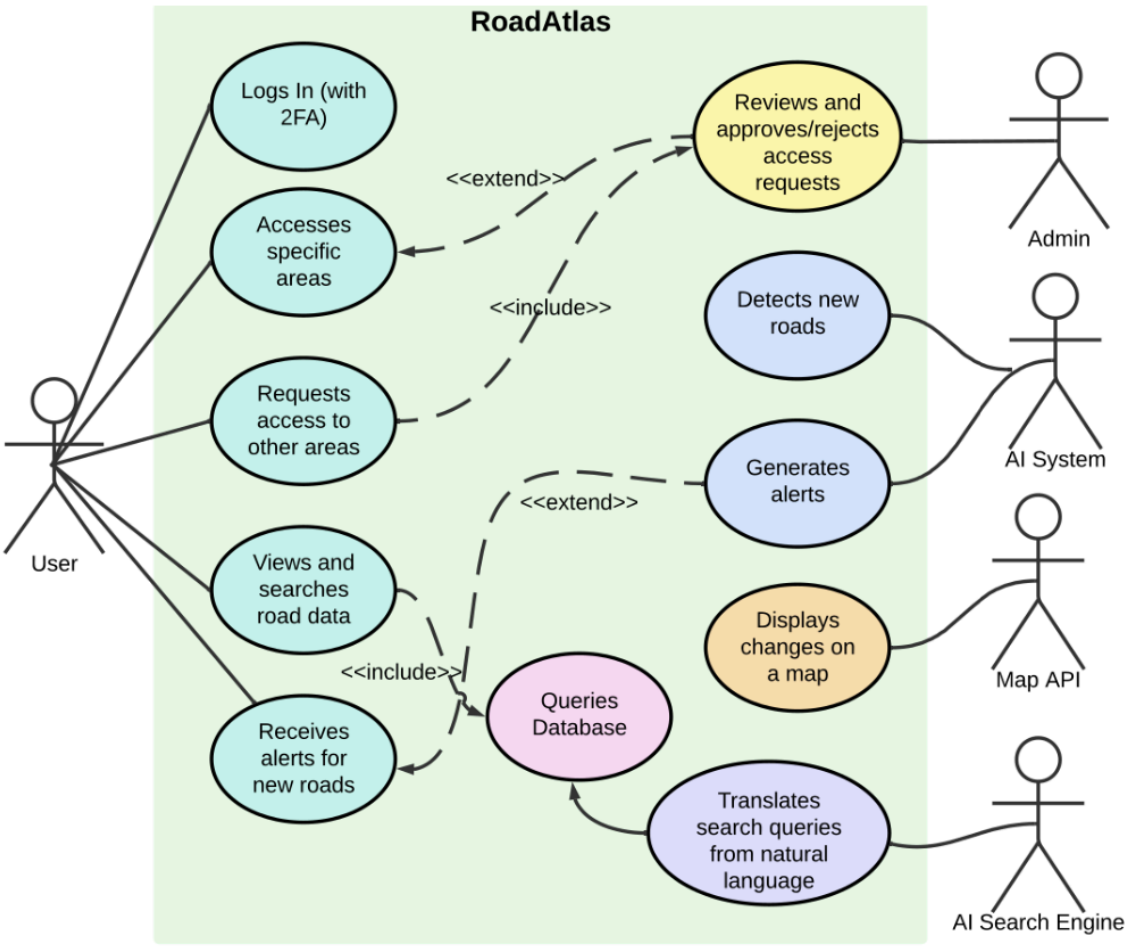
- RoadAtlas SOW (2025): *"Change Detection in Temporal Satellite Imagery for Road Extraction and Mapping"*
- Xu et al. (2018): Deep Learning for Road Extraction – Remote Sensing
- Nethravathi et al. (2020): Natural Language to Query Language – ETASR Journal
- Tan et al. (2022): BSIRNet Road Detection Network – Wiley Journal
- Malczewski (2004): GIS-based Land Use – ScienceDirect
- **Bhoonidhi Portal**: Official ISRO platform for satellite imagery access

By replacing manual image interpretation with automated analysis, the system ensures more efficient infrastructure updates.

2 Use Case View

This section outlines specific use cases that are central to the functioning of the road detection system. The following are critical for user interaction, road monitoring, and alert generation.

2.1 Use Case



Use Case: Log In (with 2FA)

- Description: A user logs into the system using their credentials followed by a second verification step for added security.
- Actors: User, 2FA Service
- Steps:
    1. User enters email and password.
    2. System sends a 2FA code to the user's email or phone.
    3. User enters the 2FA code.
    4. System verifies and grants access.

Use Case: Access Specific Areas

- Description: Users access road data limited to regions they have permission for.
- Actors: User, Queries Database
- Steps:
    1. User selects a region.
    2. System verifies authorization.
    3. Displays road data for that area.

Use Case: Request Access to Other Areas

- Description: Users request access to road data beyond their current permissions.
- Actors: User, Admin
- Steps:
    1. User submits access request.
    2. Admin receives and reviews the request.
    3. Admin approves or denies.
    4. User is notified of the decision.

Use Case: View and Search Road Data

- Description: Users view road data for authorized areas and apply search filters.
- Actors: User, Queries Database, AI Search Engine
- Steps:
    1. User enters search query in natural language.
    2. AI engine interprets query and retrieves results.
    3. System displays search results.

Use Case: Receive Alerts for New Road

- Description: Users receive notifications when road changes are detected in their areas of interest.
- Actors: User, AI System
- Steps:
    1. AI system detects road changes.
    2. System verifies change significance.
    3. Alert is generated and sent to users.

Use Case: Detects New Roads

- Description: The AI System processes satellite images to identify road changes.
- Actors: AI System
- Steps:
    1. Satellite images are preprocessed.
    2. AI model analyzes for new/widened/removed roads.
    3. Results are stored in GIS database.

Use Case: Generate Alerts

- Description: Trigger notifications based on detected road changes.
- Actors: AI System, User
- Steps:

1. Change detection exceeds threshold.
2. System generates alert message.
3. Sends to all subscribed users.

Use Case: Display Changes on a Map

- Description: System visually presents changes using map overlays.
- Actors: Map API
- Steps:
    1. System fetches spatial data.
    2. Visual overlay is generated.
    3. Map displays side-by-side or highlighted road changes.

3 Design Overview

Change Detection in Temporal Satellite Imagery for Road Extraction and Mapping is a Progressive Web Application (PWA). It enables users to query and analyze temporal satellite imagery, specifically to detect and map changes in road infrastructure over time.
This system has two types of users - users (government officials who wish to access information about road changes) and admins (higher-ups who control what different users can access). This incorporates authentication, role-based access control (RBAC), and advanced image processing through deep learning.

3.1 Design Goals and Constraints

Design Goals:

1. **Accurate Change Detection**
    - Leverage deep learning models to ensure high-precision identification of road changes between temporal satellite images.
    - Support visual comparison through overlays and highlight detected changes clearly on maps.

2. **Role-Based Access Control (RBAC)**
    - Ensure secure and restricted data access using RBAC, where users and admins have different privileges.
    - Allow fine-grained control over read and write operations based on assigned locations.

3. **Progressive Web App (PWA) Accessibility**
    - Provide a responsive and mobile-friendly interface using PWA technology.
    - Ensure seamless access on multiple platforms including Chrome, Firefox, Edge, and Safari.

4. **User-Friendly Interface**
    - Design intuitive dashboards for both users and administrators.
    - Support filtering, search, and visual tools for interaction with the satellite data.

5. **Scalability and Modularity**
    - Design the system in a modular way for easy maintenance and scalability.

- Enable future integration with other satellite sources beyond the Bhoonidhi Portal.

6. **Data Security and Auditability**
   - Enforce secure authentication using multi-factor authentication (MFA).

Constraints

1. **Hardware Requirements**
   - Requires significant computational resources for image processing and deep learning inference.
   - Needs large storage capacity for satellite imagery and extracted GIS data.
   - High-speed internet is essential to fetch images from the Bhoonidhi Portal.

Security Standards

- Authentication services should include 2FA and secure password storage.
- Must comply with ISO 27001 for secure data handling and risk management.

Integration Constraints

- System must integrate with the Bhoonidhi Portal for satellite image retrieval.
- Must support export options (CSV, PDF) only if permitted by user roles.

Implementation and Team

Development Tools & Technology Stack

- Frontend: HTML, CSS, TypeScript, JavaScript, Tailwind , React
- Backend: Python with Flask
- Database: PostgreSQL with PostGIS for spatial data
- DL Framework: U Net based segmentation model powered by TensorFlow
- Cloud Services: AWS S3 for cloud storage and report exports

Team and Schedule

- We have divided ourselves into three teams - Backend, Frontend and Deep Learning. All of Project team includes 7 student developers guided by faculty supervisors.

Legacy Code

- This project is developed from scratch; no legacy system constraints exist.

3.2 Design Assumptions

Availability of Satellite Imagery

- It is assumed that the Bhoonidhi Portal and other image sources will provide timely and accurate satellite imagery data in supported formats.

Allocated Locations Are Predefined

- User-specific location access is assumed to be predefined by the system or assigned by admins at the time of account creation or later through access requests.

All Users Have Compatible Devices

- The system assumes users will access the platform using modern web browsers (e.g., Chrome, Firefox, Edge) and that their devices can support Progressive Web App (PWA) features.

Admin Oversight for Sensitive Actions

- It is assumed that sensitive actions such as access permission changes, data exports, and manual change validation will be performed by authorized admin users only.

Security Practices Are Followed

- It is assumed that all team members and users will follow best practices for password management, 2FA, and system access to ensure data security and system integrity.

3.3 Significant Design Packages

The system design is decomposed into a modular set of packages to promote maintainability, scalability, and clear separation of concerns.

1. Presentation Layer (Frontend PWA)

Implements the user interface using modern web technologies and supports Progressive Web App features.

- Handles user authentication and role-based access control.
- Allows users to submit location/date-based search queries.
- Displays satellite imagery and visual overlays of detected road changes.

2. API Layer (Backend Interface Layer)

Exposes RESTful endpoints for the frontend to interact with the backend services.

- Routes user requests securely to backend services.
- Validates user roles and permissions.
- Handles query formatting and preprocessing for the AI and DL layers.

Dependencies: Auth Layer, Search Module, DL Pipeline

3. Authentication & Authorization Layer

Manages secure login and role-based access to data and features.

- User registration and login.
- Session and token management.
- Verifying location-specific access rights.

Dependencies: PostgreSQL (user and access control database)

4. Search System with Natural Language Interface

Translates user's natural language queries into structured SQL queries.

- Query parsing and intent recognition.
- SQL generation based on user roles and data permissions.

Dependencies: PostgreSQL

5. Satellite Imagery Fetching and Preprocessing

Connects to external APIs (e.g., Bhoonidhi) and preprocesses imagery.

- Fetching satellite imagery for specified dates and locations.
- Ensuring imagery is normalized, time-aligned, and geotagged.

Dependencies: Bhoonidhi API, DL Pipeline

6. Deep Learning Pipeline

Implements the core deep learning model for road change detection.

- Image pair processing using CNNs/transformers.
- Highlighting areas of change and generating overlay masks.

Dependencies: imagery handler, model weights, storage

7. Data Storage Layer

Handles structured and unstructured data persistence.

- Stores user data, logs, search results, and model outputs.

- Maintains historical access and audit logs.

Dependencies: PostgreSQL, S3

3.4 Dependent External Interfaces

External Application Name	Module Using the Interface	Functionality/ Description
Bhoonidhi Portal – Satellite Image API	Image Retrieval and Preprocessing Module	Used to fetch temporal satellite images based on user query parameters such as location and date. These images are then preprocessed before being passed to the road detection model.
AWS S3	Report Export and Backup Module	Used for storing exported CSV/PDF reports and backing up extracted and annotated data securely to the cloud.
PostgreSQL + PostGIS – Database Interface	Backend Services and Query Engine	Manages user data, permissions, search queries, road extraction results, and geospatial data with support for spatial indexing and querying.
Authentication Service (e.g., OAuth/2FA service) – Authentication Interface	Login & Access Control Module	Handles secure user login, two-factor authentication, and session management across web and mobile clients.

3.5 Implemented Application External Interfaces (and SOA web services)

Interface Name	Module Implementing the Interface	Functionality/ Description
User Access API	Authentication & Authorization Module	Provides user registration, login, MFA, and session management services using secure REST APIs.

Search API	Query Processing & Results Module	Allows authorized users to query for road changes in their allocated locations via API.
Road Change Detection Interface	Image Analysis and Change Detection Module	Exposes functionality to trigger change detection on satellite images and return visual results.
Admin Control Interface	Admin Dashboard Module	Enables admins to manage users, roles, and permissions through exposed endpoints.
Report Export API	Reporting & Export Module	Provides an endpoint for users/admins to download filtered results in PDF/CSV formats.
Alert Notification Interface	Notification & Alert Module	Sends email/SMS notifications to subscribed users when significant road changes are confirmed.

4 Logical View

Overview of Application Modules

The Road Change Detection System is designed with a modular architecture to support clear separation of responsibilities and ease of maintenance. The top-level modules involved in key use cases are:

- Image Acquisition Module
  Responsible for fetching satellite images from external sources like Bhoonidhi Portal.

- Preprocessing Module
  Enhances and cleans image data to prepare it for road extraction.

- Deep Learning Road Extraction Module
  Uses a trained deep learning model to identify and extract road features.

- GIS Database Module
  Stores extracted road data and historical images for future comparisons.

- User Interface Module
  Allows users (Admins/Viewers) to query road changes and view results.

- Query Handler Module
  Processes user queries and coordinates fetching and comparison of data.

- Change Detection Module
  Compares old and new road data to detect changes.

- Alert Module
  Sends notifications (email, and optionally SMS) to subscribed users when road changes are detected.

Interaction at Top Layer (Key Use Case Flow)

Use Case: Detect and Alert Road Changes

1. User (Actor) accesses the system via the User Interface.

2. User Interface Module accepts a location/date query.
3. Query Handler fetches data from the GIS Database.
4. Change Detection Module compares image data and highlights changes.
5. User Interface Module displays results.
6. User may trigger an alert.
7. Alert Module sends notifications to subscribed users.

Decomposition of Modules

1) Preprocessing Module

Responsibilities:

- Normalize image contrast
- Remove noise
- Crop or align images

Key Classes & Methods:

- ImageProcessor
    - enhanceContrast(image)
    - removeNoise(image)
    - alignImage(baseImage, newImage)

2) Road Extraction Module (TrainedModel)

Responsibilities:

- Apply CNN model to detect roads
- Generate vectorized road data

Key Classes & Methods:

- RoadExtractor
    - predict(image)
    - postProcess(prediction)
    - convertToGeoJSON(roadMask)

3) GIS Database Module

Responsibilities:

- Store and retrieve geographic image data
- Index by location and time

Key Classes & Methods:

- GISDatabaseHandler
    - storeRoadData(location, roadData)
    - getRoadData(location, date)
    - storeImage(location, image)

4) Query Handler Module

Responsibilities:

- Interpret user queries
- Request relevant data from GIS
- Forward data to comparison module

Key Classes & Methods:

- QueryProcessor
    - parseQuery(query)
    - fetchOldNewImages(location, dateRange)

5) Change Detection Module

Responsibilities:

- Compare old and new road images
- Mark significant changes

Key Classes & Methods:

- ChangeDetector
    - compareImages(oldImage, newImage)
    - highlightDifferences(diffMap)

6) Alert Module

Responsibilities:

- Send alerts via email/SMS
- Handle failures and retry

Key Classes & Methods:

- AlertService
    - sendEmail(userList, message)
    - sendSMS(userList, message)
    - logFailure(alertType, user)

7) User Interface Module

Responsibilities:

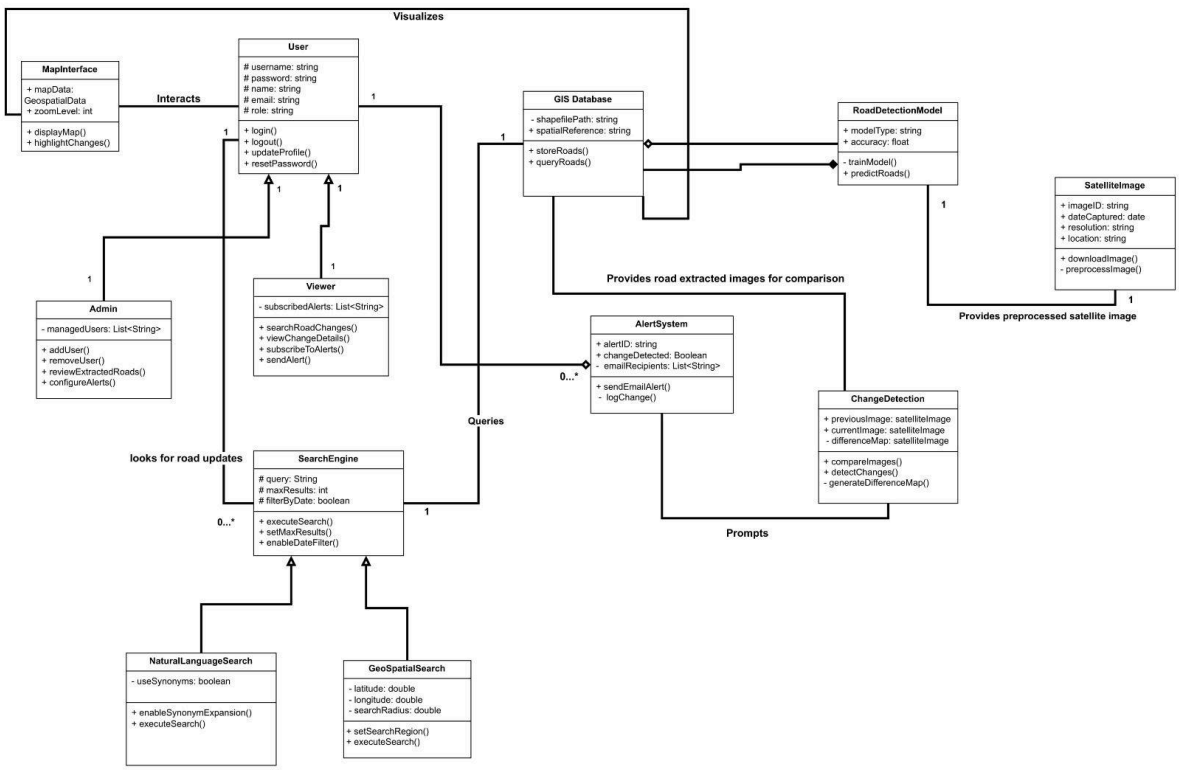
- Provide map view
- Allow interaction with road change data

Key Components:

- MapViewer
    - renderMapWithChanges(data)
- AlertButton
    - onClick() → triggerAlert()

4.1 Design Model

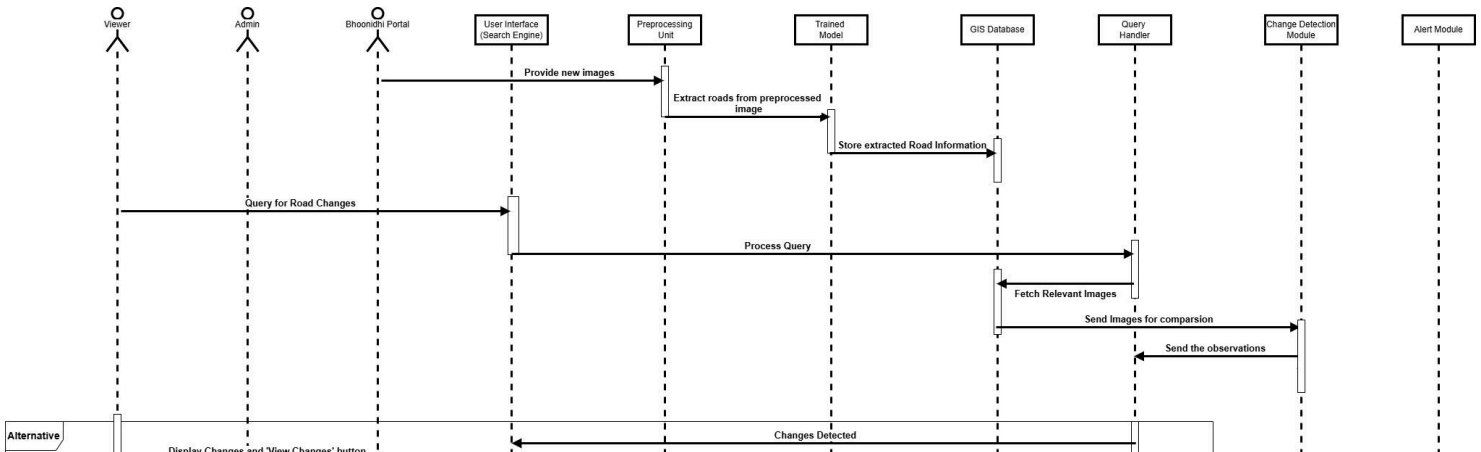
Class Diagram



Class	Responsibilities	Relationships
User	Manages login, profile, and user access. Serves as a base class for all users.	Generalized by Admin, Viewer

Admin	Reviews and approves road data, manages users, views logs.	Inherits from User; interacts with GISDatabase, AlertModule
Viewer	Searches for road changes, views results, sends alerts.	Inherits from User
SatelliteImage	Fetches satellite images from bhoonidhi	Interacts with RoadDetectionModel by providing preprocessed images
Search Engine	Base class for search strategies, defines general search interface.	Generalized by NaturalLanguageSearch, GeoSpatialSearch
NaturalLanguageSearch	Parses and converts natural language queries.	Specialization of SearchEngine
GeoSpatialSearch	Performs location-based search using coordinates or region names.	Specialization of SearchEngine
RoadDetectionModel	Extracts roads from satellite images using a deep learning model	Sends extracted road data to GISDatabase
GISDatabase	Stores satellite images with extracted road data.	Cannot exist without RoadDetectionModel(Composition); accessed by ChangeDetectionModule
ChangeDetectionModule	Compares image pairs to detect and highlight road changes.	Receives images from GIS Database; triggers AlertSystem when needed
AlertModule	Sends email notifications and handles alert logic.	Sends alerts to subscribed users
MapInterface	Interacts with user by displaying road changes if any	Receives images from GIS Database

4.2 Use Case Realization



1. Logs In (with 2FA)
Participating Classes:

- User
- AuthenticationService (assumed helper module for 2FA)
- Admin / Viewer (subclasses of User)

Description:

1. The User initiates login via the login(username, password) method.
2. The AuthenticationService validates credentials using validateCredentials().
3. If valid, the system generates a 2FA code and sends it to the user using send2FACode().
4. The user enters the code, which is verified by AuthenticationService.verify2FA(code).
5. If successful, the system sets the session

2. User Accesses Specific Areas
Participating Classes:

- Viewer
- GISDatabase
- MapInterface

Description:

1. The Viewer selects a region from the UI and calls searchRoadChanges(regionID).
2. The GISDatabase retrieves the data using queryRoads(regionID).
3. MapInterface renders the data with displayMap(data).

3. Admin Reviews and Approves/Rejects Access Requests
Participating Classes:

- Admin
- User
- AccessRequestQueue (helper module)

Description:

1. Admin calls viewPendingRequests() to retrieve access requests.
2. Admin selects an action:
    - Calls addUser() if approved
    - Or calls removeUser() if rejected

4. User Views and Searches Road Data
Participating Classes:

- Viewer
- UserInterface
- SearchEngine
- GISDatabase
- MapInterface

Description:

1. Viewer submits a query via the interface.
2. SearchEngine.executeSearch() interprets the query.
3. MapInterface.highlightChanges() shows the result to the user.

5. User Gets Alerts for New Roads
Participating Classes:

- Viewer
- AlertModule
- ChangeDetectionModule
- GISDatabase

Description:

1. Viewer subscribes via subscribeToAlerts(region, preferences).
2. ChangeDetectionModule.detectChanges() runs on image update.
3. If a new road is found, it uses AlertModule.sendEmailAlert() to notify all subscribed users

7. Change Detection System Detects New Roads and Generates Alerts
Participating Classes:

- SatelliteImage
- RoadDetectionModel
- GISDatabase
- ChangeDetectionModule
- AlertModule

Description:

1. SatelliteImage receives a new image via downloadImage().
2. Image is passed to RoadDetectionModel.predictRoads(image).
3. GISDatabase.storeRoads(roadData) stores the output.
4. ChangeDetectionModule.compareImages() finds changes.

5. If changes are significant, AlertModule.sendEmailAlert() is triggered.

8. Map Interface Displays Changes on a Map
Participating Classes:

- MapInterface
- GISDatabase
- ChangeDetectionModule

Description:

1. ChangeDetectionModule.detectChanges() provides geodata.
2. MapInterface.highlightChanges(oldData, newData) overlays both and highlights differences.
3. Viewer interacts with the map and optionally clicks "Send Alert" for updates.

5 Data View

This section describes the persistent data storage perspective of the road extraction system. The system relies on a combination of geospatial and metadata to persist detected road segments and their temporal evolution. Persistent data is crucial for storing road change logs, alert history, and satellite image metadata for audit and review purposes.

5.1 Domain Model

The domain model consists of several core entities:

- **SatelliteImage**
  Represents a satellite image captured by the LISS IV sensor.
  Attributes: image_id, capture_date, region

- **RoadSegment**
  Represents an extracted road segment from a given satellite image.
  Attributes: segment_id, geometry (GeoJSON or WKT), image_id , extraction_date.

- **TemporalChange**
  Represents a detected change across time, such as a new road or modification of existing roads.
  Attributes: change_id,change_type (e.g., NEW, MODIFIED, REMOVED), timestamp.

- **Alert**
  Represents alerts generated for new road formations.

Attributes: alert_id, region, status (PENDING, ACKNOWLEDGED, DISMISSED), priority.

- **User**
  Represents a government or authorized user who accesses the dashboard.
  Attributes: user_id, name, email, role , password.

5.2 Data Model (persistent data view)

Key Database Tables

Users:

Stores user authentication and authorization information
Contains role assignments and contact information for alerts

Regions:

Geographic boundaries defining monitoring areas
Contains polygon geometries and metadata

UserRegionAccess:

Junction table linking users to their authorized regions
Implements the role-based access control (RBAC)

SatelliteImages:

Metadata about satellite imagery including capture date, resolution, and source
References to actual image files stored in cloud storage

RoadSegments:

Vector representations of extracted road networks
Includes linestring geometries and associated attributes

5.2.1 Data Dictionary

User Table:

Column Name	Data Type	Description	Constraints
user_id	UUID	Unique identifier for each user	Primary Key
username	VARCHAR(50)	Login username	Unique, Not Null
password_hash	VARCHAR(256)	Bcrypt hashed password	Not Null

email	VARCHAR(100)	Email address for notifications	Unique, Not Null
role	ENUM	User role (Admin, Viewer)	Not Null, Default 'Viewer'
created_at	TIMESTAMP	Account creation timestamp	Not Null, Default NOW()
last_login	TIMESTAMP	Last successful login time	Nullable
is_active	BOOLEAN	Account status flag	Not Null, Default TRUE

Regions Table

Column Name	Data Type	Description	Constraints
region_id	UUID	Unique identifier for each region	Primary Key
name	VARCHAR(100)	Human-readable region name	Not Null
geometry	GEOMETRY(POLYGON)	Geographic boundary as polygon	Not Null
admin_level	INTEGER	Administrative hierarchy level	Not Null
created_at	TIMESTAMP	Region creation timestamp	Not Null, Default NOW()
metadata	JSONB	Additional region metadata	Nullable

RegionAccess Table

Column Name	Data Type	Description	Constraints
access_id	UUID	Unique identifier for access record	Primary Key

user_id	UUID	Reference to user	Foreign Key (Users)
region_id	UUID	Reference to region	Foreign Key (Regions)
access_level	ENUM	Permission level (Read, Write, Admin)	Not Null
granted_by	UUID	Admin who granted access	Foreign Key (Users)
granted_at	TIMESTAMP	When access was granted	Not Null, Default NOW()

Satellite Images Table:

Column Name	Data Type	Description	Constraints
image_id	UUID	Unique identifier for image	Primary Key
region_id	UUID	Region covered by image	Foreign Key (Regions)
capture_date	DATE	When image was captured	Not Null
cloud_cover	DECIMAL(5,2)	Percentage of cloud coverage	Not Null
resolution	DECIMAL(5,2)	Ground resolution in meters	Not Null
source	VARCHAR(50)	Image source (e.g., LISS-IV)	Not Null
storage_path	VARCHAR(255)	Path to image file in storage	Not Null
processed	BOOLEAN	Whether image has been processed	Not Null, Default FALSE

6 Exception Handling

The Road Change Detection system implements a comprehensive exception handling strategy to ensure system stability and provide meaningful feedback to users and system administrators.

Exception Type	Description	Thrown When
`AuthenticationException`	Authentication or authorization failures	Invalid credentials, expired sessions, insufficient permissions
`ImageProcessingException`	Problems with image processing	Corrupt image data, format issues, preprocessing failures
`ModelPredictionException`	Deep learning model failures	Inference errors, incompatible input data
`GISDataException`	GIS data handling errors	Invalid geometries, coordinate reference system mismatches
`ExternalAPIException`	External service communication errors	Bhoonidhi API connectivity issues, rate limits exceeded
`DatabaseException`	Database operation errors	Connection failures, constraint violations, query timeouts
`ValidationException`	Input validation errors	Invalid user input, malformed queries

6.2 Exception Handling Process

Logging:
All exceptions are automatically logged with:

- Timestamp
- Exception type and message
- Stack trace
- User ID (if authenticated)
- Request details
- System state information

User Notifications:

- **Authentication errors:** Clear login failure messages without exposing system details
- **Validation errors:** Specific guidance on how to correct input
- **Processing errors:** User-friendly messages with error reference codes

Retry Logic:

- API calls implement exponential backoff for transient failures
- Image processing tasks are automatically retried up to 3 times
- Database connections have connection pooling with retry mechanisms

Graceful Degradation:

- If change detection fails, system falls back to showing most recent successful data
- If real-time processing is unavailable, batch processing is scheduled

6.3 Critical Exception Handling

For critical exceptions that might affect system integrity:

Alert Mechanism:

- Immediate email notification to system administrators
- Dashboard alert for admin users
- Entry in the system health monitor

Recovery Actions:

- Automatic service restart for non-responsive components
- Rollback to previous state for database transactions
- Circuit breaker pattern for external API calls

7 Configurable Parameters

This table describes the simple configurable parameters

Configuration Parameter Name	Definition and Usage	Dynamic?
SESSION_TIMEOUT	Duration before a user session expires due to inactivity.	Yes
MAX_RETRY_LOGIN_ATTEMPTS	Maximum number of failed login attempts before locking the account is 3.	Yes
DATASET_ACCESS_LEVELS	Defines which user roles can access which datasets or regions	No
NOTIFICATION_EMAIL	Admin email to which alerts and request updates are sent	Yes
ENABLE_TWO_FACTOR_AUTH	Flag to enable/disable two-factor authentication during login..	Yes

8 Quality of Service

8.1 Availability

The system is designed to be available at most times, with minimal downtime. Heavy tasks like image processing or road change detection are done in the background to keep things smooth for users. Any maintenance or updates are scheduled during off-peak hours. The system uses Boonidhi's database to store satellite images and road data, with regular backups to prevent data loss.

8.2 Security and Authorization

Only authorized users such as admins and government officials can access specific parts of the system. Login credentials are required, and access is limited based on user roles. User permissions are handled securely through predefined roles, and the admin panel allows easy management of users and their access levels.

8.3 Load and Performance Implications

The system is optimized to handle many users and large images without slowing down. Image processing and data storage are done efficiently to keep up with road updates. The design supports future growth, such as more users or more frequent image uploads, without performance issues.

8.4 Monitoring and Control

The system keeps logs of actions like image uploads, road change detections, and alerts. These logs help find any problems quickly and make the system easier to maintain. It also runs scheduled tasks in the background to keep road data updated and send alerts without delay. Admins can use their dashboard to check what the system is doing, see recent activity, and make sure everything is working properly.