Long-term Flood Prediction in Endangered Areas of Pakistan

****

Final Year Project Report

By

**Abdullah Tahir - BSCS11 - 385714 - SEECS**

**Muhammad Zaeem Khalid - BSCS11 - 371303 - SEECS**

Supervisor

**Dr. Momina Moetesum**

Co-supervisor

**Dr. Nazia Perwaiz**

Department of Computer Science

In partial fulfillment of the requirements for the degree of Bachelor of Computer Science

In

School of Electrical Engineering and Computer Science (SEECS),

National University of Science and Technology (NUST),

Islamabad, Pakistan

(May, 2025)

**Declaration**

We hereby declare that this project titled “Long-term Flood Prediction in Endangered Areas of Pakistan” submitted to the “School of Electrical Engineering and Computer Science (SEECS)”, is a record of an original work done by us under the guidance of supervisor “Dr. Momina Moetesum” and co-supervisor “Dr. Nazia Perwaiz” and that no part has been plagiarized or referenced without proper citation. This project is submitted in partial fulfillment of the requirements for the degree of Bachelor’s in Computer Science

Team Members Signatures

Abdullah Tahir \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Muhammad Zaeem Khalid \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Supervisor

Dr. Momina Moetesum \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Date: 10th May, 2025\_\_

Place: Islamabad, Pakistan

**Dedication**

We would like to dedicate our work to our teachers without whom this project would not have reached completion.

**Acknowledgements**

We would like to thank our supervisor who bore with us throughout the course of this project, her guidance has been eternally beneficial and we hope to have met her expectations. We would also like to thank our co-supervisor who proved to be invaluable considering her helpful remarks.

**Table of Contents**

# **Introduction**

## **Problem Statement**

There is a lack of a robust early warning system designed for severe floods which caters to every

socio-economic class within the country. The floods of 2022 alone caused over 14.9 billion

dollars in damages and 15.2 billion dollars in economic losses. While there are weekly advisories

put out by relevant government authorities, it seems that it is not directly communicated to the

public.

## **Literature Review**

Flood prediction is usually done with hydrological models which depend upon river gauge data and meteorological data but they often fail to generalize across time and geography due to their static nature (Dhital, 2024). With how much machine-learning and deep-learning have advanced, it is not out of the ordinary to imagine that such problems may be solved by data-driven machine-learning or deep-learning models. Recent literature uses Transformer models for flood forecasting with a lead-time of one day (Castangia et al., 2023) or with a lead-time of three days (Demiray & Demir, 2024). These discoveries highlight the success of such models in flood forecasting but Pakistan still lacks a robust early warning system which leverages these technologies effectively (Waseem & Rana, 2023).

## Proposed Solution

To address the limitations of conventional flood forecasting, this project proposes a long-term flood prediction system using a deep learning-based Transformer model. The model is trained on multivariate hydro-meteorological data which includes variables like precipitation, runoff, and total column water vapor, to detect patterns and predict flood events up to 30 days in advance. The system integrates this model into a web-based platform where users can view predictions for specific grid cells defined by latitude and longitude. The backend is powered by a FAST-API service that serves model outputs to the user interface in real-time. This architecture ensures scalability, modularity, and responsiveness for both research and deployment purposes.

## Project Scope

The scope of this project includes the collection and preprocessing of satellite and reanalysis datasets (i.e. ERA5, GloFAS), model design and training using Transformer architectures, deployment of the trained model as a microservice using FAST-API, and development of a web interface that enables users to view flood risk forecasts interactively. The system is designed to operate at a grid-cell level resolution, enabling localized predictions. While the model is currently built for inference on preloaded data, future enhancements may include live data ingestion and alerts. The project does not cover real-time sensor integration or economic impact modeling.

## Stakeholders

The primary stakeholders for this system include government agencies responsible for disaster management (such as NDMA and PDMA), local authorities in flood-prone districts, researchers in climate science and hydrology, and potentially affected communities who rely on timely flood alerts for safety and preparedness. Secondary stakeholders may include NGOs, infrastructure planners, and academic institutions interested in climate resilience.

## Tech Stack

The technology stack comprises several key components. Data preprocessing and model training are performed using Python with libraries such as NumPy, Pandas, Xarray and TensorFlow. The deep learning model utilizes a Transformer-based architecture, optimized for sequential data processing. The backend service is built using FAST-API, which offers high performance for serving model inferences. The frontend is developed using HTML, CSS, and JavaScript (with support from frameworks like React or Leaflet.js for interactive maps). The dataset is stored in NetCDF format and visualized using GeoJSON and vector tiles for map rendering.

# Software Requirements Specifications (SRS)

## Requirements Elicitation Process

The requirements for the Flood Forecast System were established through a combination of supervisor consultations, a review of existing literature and relevant systems, and an analysis of the problem domain. Discussions with project supervisor, Momina Moetesum, provided foundational guidance on scope and objectives. A review of the initial project proposal, and documentation from organizations like WAPDA and PMD informed the technical and data requirements. Analysis of existing flood forecasting efforts, including systems like GloFAS, helped identify common functionalities. Understanding the critical need for flood prediction in Pakistan, as emphasized by historical events, directly influenced the core features related to meteorological and hydrological data integration, geospatial analysis, and user-friendly visualization. The definition of distinct user classes i.e. General Public, Government Authorities, Researchers, and System Administrators, guided the development of tailored functionalities for each group.

## Use Case Diagrams and Descriptions

Key system interactions and user capabilities include:

* **View Flood Risk Map (All User Classes):** Users access the web application to view an interactive map of Pakistan displaying color-coded regional flood risk levels. Hovering over a region reveals specific details such as threat level, risk percentage, and average rainfall (e.g., for Punjab: "Threat level: low," "Risk: 0.04%", "Rainfall: 60.4mm avg.").
* Map interactions include zoom, pan, and planned layer toggling (satellite, terrain).
* **Receive Flood Alerts (Registered Users):** Registered users can configure personalized alerts for specific areas of interest, receiving notifications via SMS/email when predicted flood risk exceeds predefined thresholds.
* **Manage System (System Administrator):** Administrators are responsible for user account management, system configuration, health monitoring, and initiating model retraining processes.
* **Access System Information (All Users):** Users can access informational pages (e.g., "About," "How does it work?") to understand the system's purpose, methodology, and data sources.
* **Register/Login (Prospective/Registered Users):** Functionality for users to create new accounts or log into existing ones to access personalized features such as dashboards.

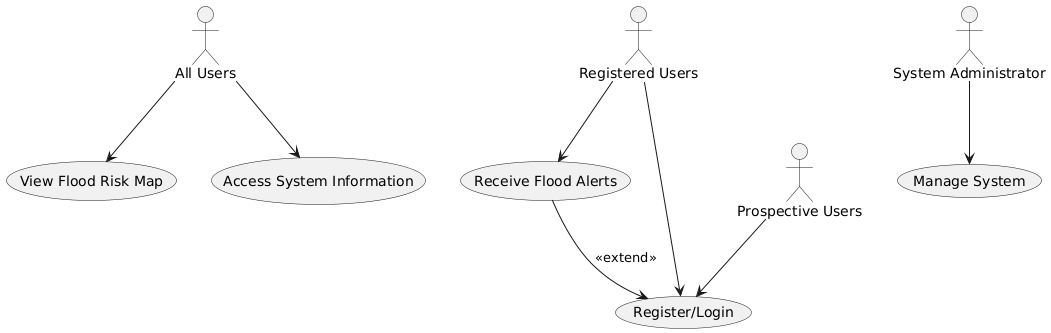


Fig 1. UML Use Case Diagrams

## Functional Requirements (FRs)

The system is designed with the following core functionalities:

1. **Data Collection and Preprocessing:**
   * Manual collection of data from CDS (Climate Data Store) through their API.
   * Successful conversion of higher-dimensional data into a simpler dataframe structure.
   * Comprehensive data cleaning, normalization, validation, and missing value imputation.
   * Maintenance of a log for all data collection and preprocessing activities.
2. **Flood Prediction Modeling:**
   * Generation of long-term flood predictions with at least a 30-day lead time.
   * Utilization of a Transformer model to deliver effective warnings within certain regions.
   * Output of flood probabilities, risk classifications, confidence levels, and lead time predictions.
3. **Geospatial Analysis and Visualization:**
   * Generation and display of interactive, color-coded flood risk maps.
   * Interactive map features including zoom, pan, and hover-over regional details.
   * Detailed analysis of predictions by clicking upon a certain grid-cell.
4. **Web Application Features:**
   * Provision of a user-friendly web interface accessible via standard browsers.
   * Support for user registration, authentication, and personalized dashboards.
   * Access to queryable historical flood data, including statistical summaries and trend analysis.
   * Informational content about the project methodology and purpose.
5. **Database Management:**
   * Centralized storage for historical flood data, model predictions, user information, and system configurations.
   * Efficient storage and management of geospatial data utilizing PostgreSQL with the PostGIS extension.

## Non-Functional Requirements (NFRs)

The system adheres to the following quality attributes:

1. **Performance:**
   * Web application load time: < 3 seconds on standard broadband.
   * Concurrent user support: Up to 10,000 users without performance degradation.
   * Country-wide flood prediction calculation time: < 1 hour.
   * Interactive map rendering (zoom, pan): < 1 second response time.
2. **Security:**
   * Encryption of all user data at rest (AES-256) and in transit (TLS 1.2).
   * Implementation of Role-Based Access Control (RBAC).
   * Regular security audits, penetration testing, and vulnerability scanning.
   * Secure API endpoints with rate-limiting and API key management.
3. **Reliability and Availability:**
   * Target uptime: 99.9% for critical services.
   * Automated hourly backups of the PostgreSQL/PostGIS database.
   * Implementation of disaster recovery mechanisms.
   * Fault tolerance through load balancers and failover strategies.
4. **Scalability:**
   * Deployment on cloud-based infrastructure (AWS or GCP) supporting dynamic resource allocation.
   * Architecture designed for horizontal scaling to handle increased loads.
5. **Usability & Accessibility:**
   * Intuitive and user-friendly interface design.
   * Compliance with WCAG 2.1 Level AA accessibility standards (screen reader support, keyboard navigation, high-contrast modes).
   * Multilingual support (initially English and Urdu).
   * Optimization for low-bandwidth internet connections.
6. **Maintainability:**
   * Modular code architecture for ease of development and updates.
   * Implementation of CI/CD pipelines for automated testing and deployment.
   * Version control and comprehensive documentation for machine learning models.
   * Preference for open-source technologies where feasible.
7. **Safety:**
   * The system must not compromise user safety.
   * Clear disclaimers regarding the limitations of predictions must be prominently displayed.

## Storyboards and UI Flows

**Primary User Flow: Exploring Regional Flood Risk (e.g., Punjab)**

1. The user accesses the application, landing on the homepage which introduces the system and provides navigation.
2. The user navigates to the "Map" page via the homepage call-to-action or header link. The map page displays an interactive map of Pakistan with a legend for risk levels.
3. The user hovers the cursor over a specific region (e.g., Punjab).
4. A tooltip appears, displaying region-specific information: name, threat level, risk percentage, average rainfall, and a "Get more details" button.
5. (Clicking "Get more details" navigates to a page with more granular information for the selected region.

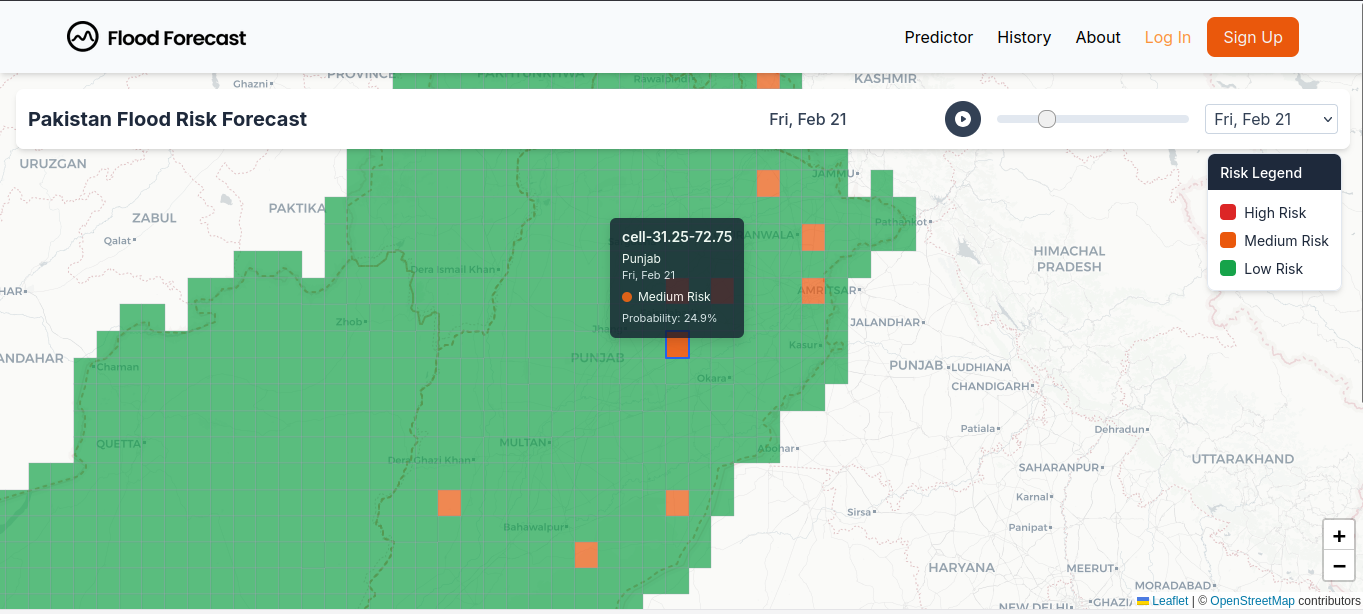


Fig 2. Viewing a prediction for the region of Punjab

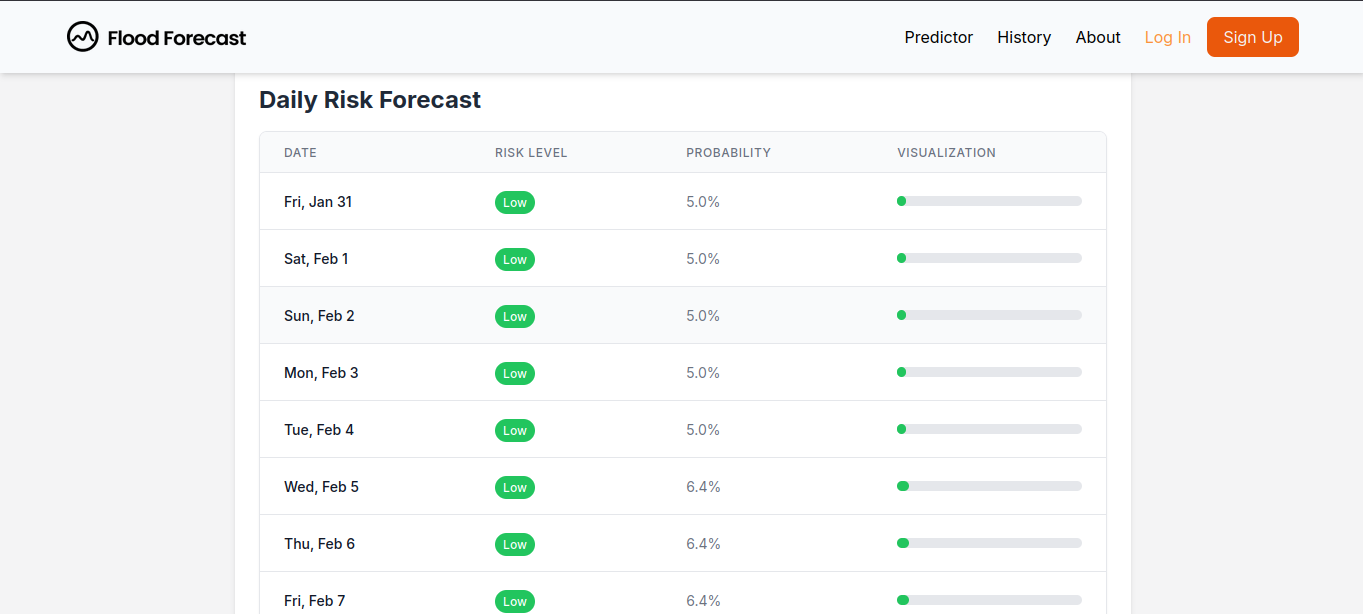


Fig 3. Details about that particular grid-cell

**Secondary User Flow: Accessing Project Information**

1. From any page, the user clicks "About" in the header navigation.
2. The user is navigated to the "About" page, which details the project's goals, methodology, and intended impact.

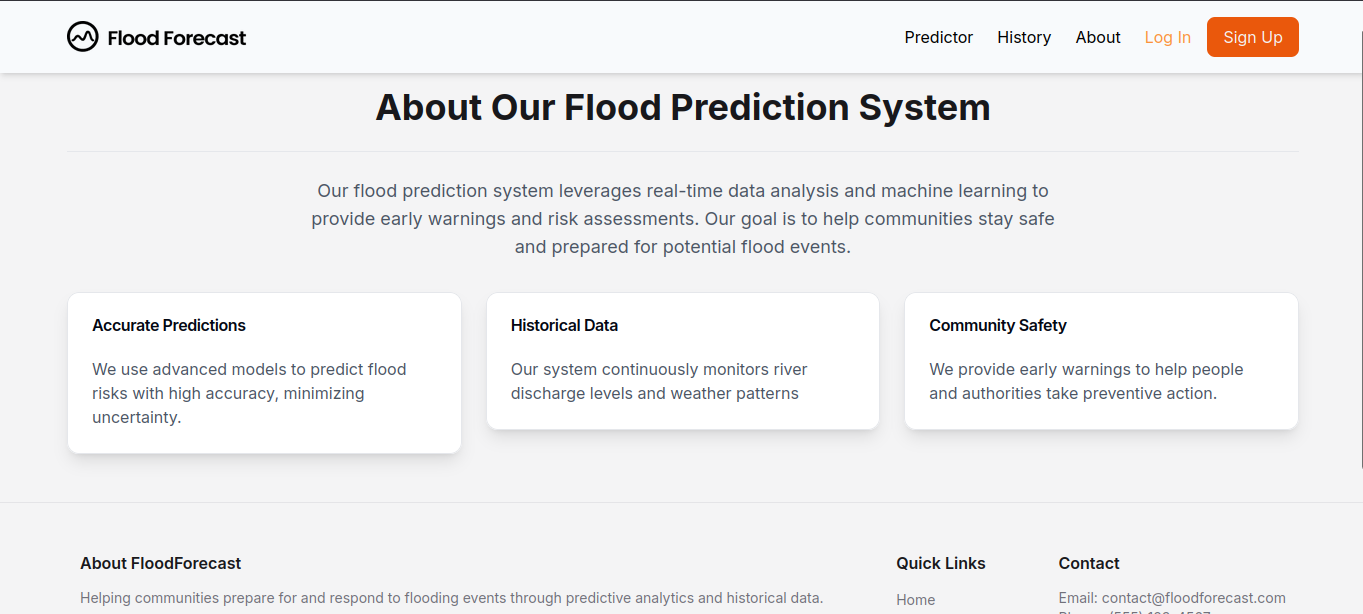


Fig 4. The About page

# Software Design Specifications (SDS)

## Architectural Overview

The Long-term Flood Prediction System employs a component-based architecture, comprising five key interacting components

1. **Machine Learning/Deep Learning Component:** Serves as the analytical core, responsible for generating long-term flood risk predictions. It ingests preprocessed meteorological and hydrological data. The component utilizes a Transformer model to learn the temporal trend within the data. It includes modules for model retraining, and performance evaluation. Outputs include flood probabilities, risk levels, and confidence intervals.
2. **Data Analysis Component:** Manages the collection, preprocessing, and validation of raw data from CDS (Climate Data Store). Tasks include data cleaning, normalization, validation, and the generation of statistical summaries and historical trend analyses. This component provides high-quality, preprocessed datasets to the ML and Geospatial components.
3. **Geospatial Analysis Component:** Combines flood predictions with geographical data to perform spatial analysis and generate risk maps. It processes terrain data and identifies vulnerable regions using the predictions from the model. Built on the PostGIS platform, it outputs color-coded, layered flood risk maps and facilitates impact analysis on infrastructure and land use.
4. **Web Application Component:** Provides the user-friendly interface for accessing predictions, maps, and alerts. Developed using Next.js and React, it features interactive maps with zoom and filter capabilities, an alert system (email, SMS), and role-based access control. It retrieves data from the Database and Geospatial components for presentation.
5. **Database Component:** Functions as the central repository for all system data, including historical flood records, model predictions, user information, and system configurations. It utilizes PostgreSQL with the PostGIS extension for efficient storage and management of geospatial data. Key features include indexing, querying capabilities, and version control.

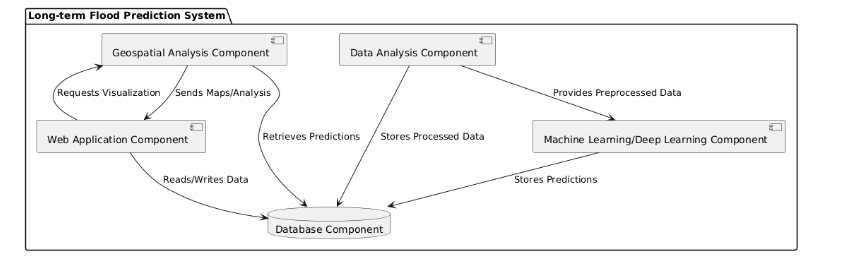


Fig 5. High-level Diagram of Component Interaction

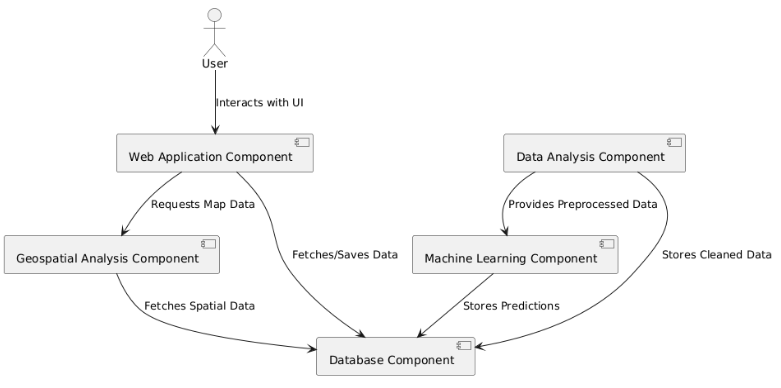


Fig 6. Detailed Component Interaction Diagram

## Design Methodology

The design of the Flood Forecast System adheres to several key methodologies:

* **Component-Based Design:** The system is architected as a collection of distinct, interacting components (Machine Learning, Data Analysis, Geospatial Analysis, Web Application, Database). This modular approach facilitates development, testing, and maintenance.
* **Iterative and Incremental Development:** The implementation has been done iteratively where a skeleton of the current application was built and further refined until its completion.
* **Data-Driven Design:** The system's core functionality relies heavily on the acquisition, processing, and analysis of diverse datasets. The design of the ML models and analytical components is fundamentally shaped by the nature and availability of this data.
* **User-Centric Design:** The Web Application Component is designed with a strong focus on user experience, accessibility, and catering to the distinct needs of different user classes, ensuring that complex information is presented in an understandable and actionable format.

This structured approach allows for managing the complexity of the system by breaking it down into manageable parts and clearly defining their interactions and individual responsibilities.

## Process Flows

The SDS document includes several diagrams that illustrate key process flows within the system:

1. **Data Preprocessing to Prediction Generation:**This component cleans, validates, and preprocesses the data, storing the processed version in the Database. Preprocessed data is then sent to the Machine Learning Component, which applies its models and stores the resulting predictions in the Database.

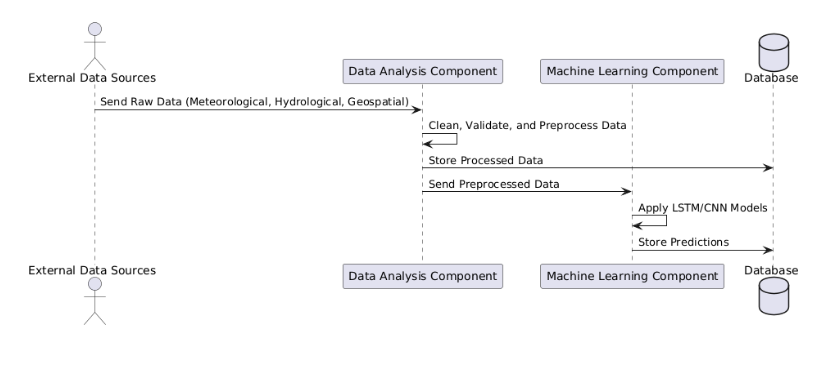


Fig 7. Sequence diagram for data pre-processing to prediction generation pipeline

1. **Data Flow (Level 1):** External Data Sources supply Raw Data to the Data Analysis Component, which sends Processed Data to the Machine Learning Component. The ML Component stores Predictions in the Database. The Database, in turn, sends Predictions to the Geospatial Analysis Component, which generates Visualizations for the Web Application. The Web Application displays these Results to End Users and handles their requests for Data/Alerts.

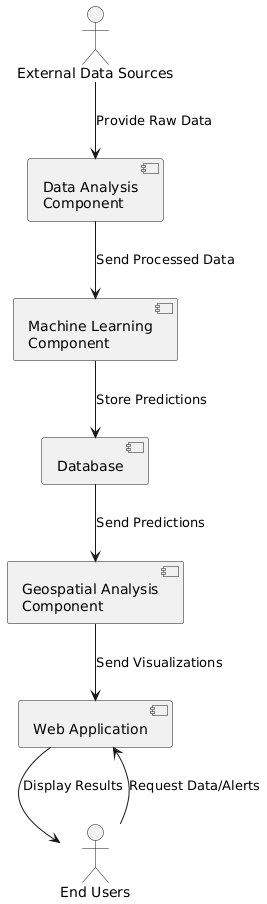


Fig 8. Data Flow (Level 1) Diagram

1. **Machine Learning Component:** Depicts the states and transitions of the ML component, including Data Preprocessing, Model Training, Prediction Generation, Prediction Storage, and an Idle/Waiting state, with transitions triggered by events like receiving new data for retraining or stopping the process.

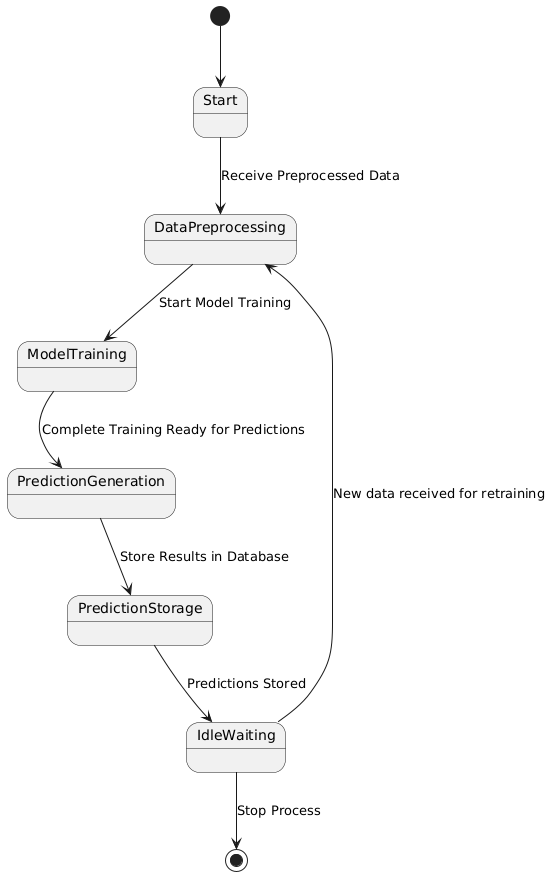


Fig 9. Machine Learning Component State Diagram

1. **Data Flow Diagram (Level 0):** Presents the highest-level abstraction, showing External Data Sources providing Raw Data to the Long-term Flood Prediction System. The system then Delivers Predictions, Risk Maps, and Alerts to End Users, who can also Request Predictions and Alerts from the system.

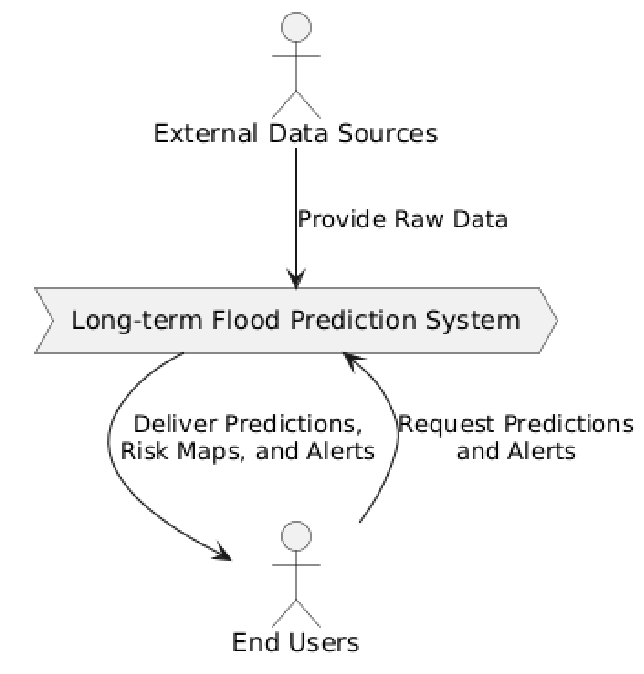


Fig 10. Data Flow (Level 0) Diagram

1. **Web Application Component:** Outlines the states of the web application from a user perspective, including Start, User Authentication, Dashboard Display, Data Request, Map Visualization, Alert Setup, and Log Out, with transitions driven by user interactions.

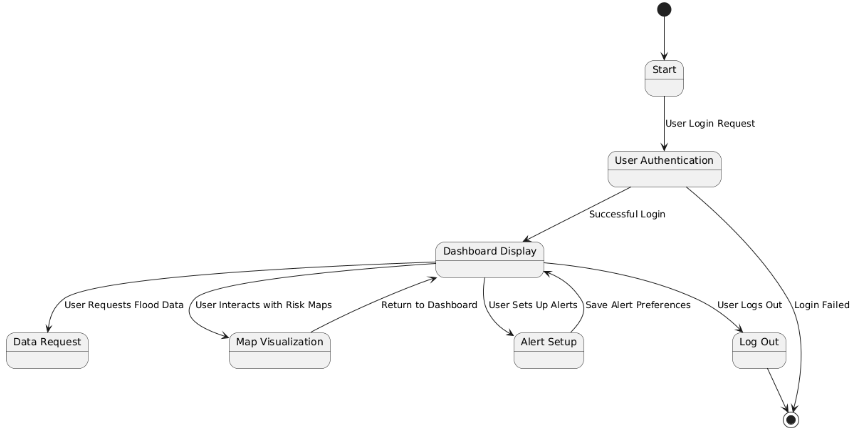


Fig 11. Web Application Component State Diagram

1. **User Request to Flood Visualization:** Details the interaction when a user requests flood information. The User sends a request to the Web Application, which fetches Predictions from the Database. The Web Application then requests Risk Maps from the Geospatial Analysis Component (which may also query the Database for spatial data). The Geospatial Analysis Component processes this and sends Risk Maps (visualizations) back to the Web Application for display to the User.

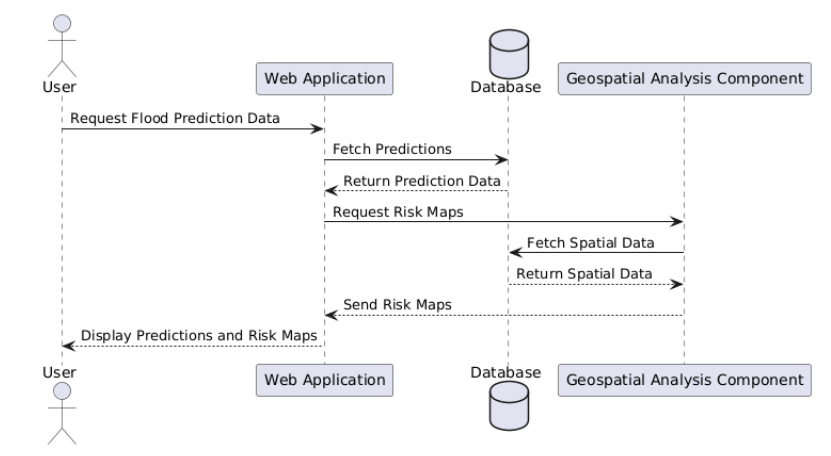


Fig 12. Sequence Diagram for User Request to Flood Visualization

## Transformer Model Architecture

The transformer architecture used in this project is designed to perform sequence-to-sequence binary classification over a 30-day forecast horizon using six hydrometeorological input features. The model begins with an input layer that accepts sequences of shape (30, 6), followed by a dense projection that maps the input features to a higher-dimensional space (d\_model = 128). This is followed by a global self-attention layer implemented using multi-head attention with four attention heads, allowing the model to capture temporal dependencies across all timesteps. A residual connection and layer normalization are applied to stabilize training. The attention output is then passed through a position-wise feed-forward network comprising two dense layers (ff\_dim = 256), followed by another residual connection and normalization. The final output layer is a time-distributed dense layer with a sigmoid activation that produces binary predictions (flood or no-flood) for each timestep in the forecast window. The model is compiled using the Binary Focal Crossentropy loss to address class imbalance, and it is optimized using the Adam optimizer with a learning rate of 1e-4. Performance metrics include accuracy, precision, recall, and AUC to comprehensively evaluate the model's classification effectiveness.

# System Testing

## Functional Testing

Functional testing ensures that the system operates in accordance with its specified functional requirements. The approach is as follows:

### Unit Testing

Unit testing focuses on the isolated verification of individual modules, components, or functions.

* **Machine Learning/Deep Learning Component:** Individual model functions, data transformation steps, and prediction logic within the AI module are tested using controlled datasets before integration.
* **Data Analysis Component:** Data collection scripts from CDS using their API, data cleaning and preprocessing, and data quality checks are implemented separately.
* **Geospatial Analysis Component:** Functions for map layer generation are tested in isolation.
* **Web Application Component:** Individual React components, API interaction handlers, state management logic, and UI rendering for specific views/pages are tested.
* **Database Component:** CRUD operations, query correctness, and PostGIS spatial query functions are verified.  
  The objective is to ensure each module functions correctly before system integration.

### Integration Testing

Integration testing verifies the interactions and data flow between different system components.

* **Data Analysis <=> Machine Learning:** Validates the flow of preprocessed data to the ML Component and its ability to utilize this data for model input.
* **Machine Learning <=> Database:** Tests the storage of predictions from the ML Component into the Database and their subsequent retrieval.
* **Geospatial Analysis <=> Database:** Verifies the retrieval of predictions and spatial data by the Geospatial Component from the Database, and the storage of generated map data/metadata.
* **Web Application <=> Backend Services:** Tests API calls from the frontend to backend services for fetching map data, predictions, historical data, and handling user authentication.
* **External Data Sources <=> Data Analysis:** Assesses the reliability of data ingestion pipelines from external providers.  
  The goal is to ensure seamless communication and accurate data exchange between integrated components.

### System Testing

System testing evaluates the complete, integrated system end-to-end to confirm it meets all specified requirements.

* **End-to-End Scenarios:** Involves testing complete user workflows, such as a user accessing the web application, viewing a flood risk map for a specific region, and receiving an alert.
* **Requirement Verification:** All functional requirements outlined in the SRS are validated.
* **Data Pipeline Validation:** The entire data pipeline, from raw data ingestion through processing, prediction, to final visualization and alert generation, is tested.
* **User Acceptance Testing (UAT):** Validation is conducted by representative user groups (general public, researchers) to ensure the system meets their needs and is usable. This includes testing interface usability, multilingual features, and accessibility features across different user classes.  
  The objective is to confirm that the entire system functions correctly and fulfills user expectations.

## Non-functional Testing

Non-functional testing assesses the quality attributes of the system, such as performance, security, and usability.

### Performance Testing

Performance testing evaluates system responsiveness, stability, and scalability under various load conditions.

* **Key Performance Metrics and Targets:**
  + Web application load time: Target of under 10-15 seconds.
  + Flood prediction calculation time: Target of within 30 seconds for country-wide predictions.
  + Concurrent user support: Target of 10,000 users without performance degradation.
  + Interactive map rendering speed (zoom/pan): Target of under 1 second.
  + API response times.
* **Testing Types:** Includes load testing (simulating expected user traffic), stress testing (pushing the system beyond normal operational limits), scalability testing (verifying ability to handle increasing load), and endurance testing (assessing stability over extended periods).  
  The goal is to ensure the system meets defined performance NFRs and provides an optimal user experience under varying loads.

### Security Testing

Security testing aims to identify and mitigate vulnerabilities to protect system data and integrity.

* **Comprehensive Security Evaluation Methods:**
  + **Penetration Testing:** Simulating real-world attack scenarios to identify exploitable vulnerabilities.
  + **API Endpoint Security Assessments:** Evaluating APIs for common vulnerabilities (e.g., injection, broken authentication/authorization, improper data exposure).
  + **Authentication and Authorization Mechanism Verification:** Testing the implementation of RBAC, session management, and password policies.
  + **Encryption Protocol Effectiveness Evaluation:** Verifying the correct implementation and strength of TLS 1.2 for data in transit and AES-256 for data at rest.
  + **Vulnerability Scanning:** Utilizing automated tools to scan code, dependencies, and infrastructure for known vulnerabilities.
  + **Security Audits:** Reviewing code and configurations against security best practices.

The objective is to ensure the system meets security NFRs and is resilient against common cyber threats.

# Deployment and System Integration

## Back-end Deployment with FAST-API

The backend, utilizing FastAPI, will be deployed following modern practices:

* **Containerization:** The FastAPI application will be containerized using Docker. A Dockerfile will define the Python environment, dependencies (FastAPI, Uvicorn, relevant libraries), and application code.
* **Application Server:** Uvicorn will serve as the ASGI server, likely managed by Gunicorn in a production environment for process management and worker scaling.
* **Cloud Platform Deployment:** As specified in the SDS (p.25), deployment will be on either Amazon Web Services (AWS) or Google Cloud Platform (GCP).
  + **AWS Options:** Elastic Beanstalk, Elastic Container Service (ECS), Elastic Kubernetes Service (EKS), or AWS Lambda for specific serverless functions.
  + **GCP Options:** App Engine, Cloud Run, Google Kubernetes Engine (GKE).
* **Database Connectivity:** Secure connection strings and credentials for accessing the PostgreSQL/PostGIS database will be managed via environment variables or dedicated secret management services.
* **CI/CD Pipeline:** A Continuous Integration/Continuous Deployment pipeline (SDS p.25) using tools such as Jenkins, GitLab CI, or GitHub Actions will automate the deployment process. This typically involves code commit triggering automated tests, Docker image building and pushing to a container registry (e.g., AWS ECR, Google Container Registry), deployment to a staging environment for validation, and subsequent promotion to the production environment.
* **API Gateway (Recommended):** Services like AWS API Gateway or Google Cloud API Gateway may be implemented to manage request routing, rate limiting, authentication, caching, and logging for the backend APIs.

## Model Hosting and Inference Pipeline

The Machine Learning models (LSTM, CNN ensemble) require a robust hosting and inference pipeline:

* **Model Serialization:** Trained models will be serialized using appropriate methods (e.g., joblib, pickle, TensorFlow SavedModel, PyTorch torch.save) for deployment.
* **Model Hosting Options (Cloud-based):**
  1. **Managed ML Platforms:** AWS SageMaker or Google AI Platform (Vertex AI) offer dedicated services for deploying models as scalable inference endpoints.
  2. **Containerized Deployment:** Models can be packaged with their inference code (e.g., within a FastAPI service) into Docker containers and deployed on compute services like ECS/EKS (AWS) or Cloud Run/GKE (GCP).
* **Inference Pipeline Workflow:**
  1. **Trigger:** Predictions can be generated on a schedule (orchestrated by Apache Airflow, AWS EventBridge, or Google Cloud Scheduler) or on-demand.
  2. **Data Retrieval:** The inference job fetches the latest preprocessed input data from the Database Component.
  3. **Prediction:** The hosted model processes the input data to generate flood predictions.
  4. **Postprocessing:** Raw model outputs are converted into user-interpretable formats (e.g., risk classifications).
  5. **Storage:** Generated predictions are stored in the Database Component.
* **Model Retraining Pipeline:** Automated periodic retraining (SDS p.11, 26) will involve fetching updated training data, retraining models, evaluating performance, and deploying improved model versions. This is typically managed by MLOps tools like Kubeflow, MLflow, or cloud-native pipeline services.

## Front-end Deployment and Integration

The Next.js/React frontend application is deployed and integrated as follows:

* **Deployment Platform:** The application is deployed on Vercel, a platform optimized for Next.js applications, providing CI/CD, global CDN, and serverless functions. This is evidenced by the live application URL.
* **Vercel Deployment Process:** Connection of the code repository (GitHub, GitLab, Bitbucket) to Vercel enables automatic builds and deployments upon code pushes to specified branches. Preview deployments for feature branches are also supported.
* **Backend Integration:** The Next.js frontend communicates with the FastAPI backend via RESTful API calls over HTTPS. API endpoints are used for fetching flood predictions, map data, historical information, and (planned) user authentication. The backend API base URL is configured as an environment variable in Vercel.
* **Mapping Services Integration:** Leaflet.js / Mapbox API (SDS p.21) are integrated into the frontend for rendering geospatial layers. Mapbox API keys are managed as secure environment variables.

## Cloud Services

The system leverages various cloud services for its operation:

* **Frontend Hosting:** Vercel (for Next.js application deployment, CDN, CI/CD).
* **Backend & ML Model Hosting (AWS or GCP):**
  + **Compute:** AWS EC2, ECS, EKS, Lambda; GCP Compute Engine, Cloud Run, GKE, App Engine.
  + **Database:** AWS RDS for PostgreSQL, Aurora for PostgreSQL; GCP Cloud SQL for PostgreSQL (with PostGIS extension).
  + **ML Platforms:** AWS SageMaker; GCP Vertex AI.
  + **Storage:** AWS S3; GCP Cloud Storage (for data, model artifacts, backups).
  + **Networking:** AWS VPC, API Gateway, Elastic Load Balancing; GCP VPC Network, Cloud Endpoints/API Gateway, Cloud Load Balancing.
  + **Orchestration & CI/CD:** Apache Airflow (self-hosted or managed services like AWS MWAA, Google Cloud Composer); AWS CodePipeline suite, Google Cloud Build/Deploy, or third-party tools.
  + **Monitoring & Logging:** AWS CloudWatch; GCP Cloud Monitoring & Cloud Logging.
* **External Mapping Services:** Mapbox API.

# User Interface

## Overview of UI Pages

The web application interface comprises the following key pages:

1. **Homepage (/):** Serves as the primary entry point, introducing the system's purpose and capabilities. It includes a hero section with a call-to-action ("Go to map"), informational cards ("How can it help?"), a visualization of Pakistan's flood vulnerability, and a section explaining "How does it work?". Standard footer links are present.

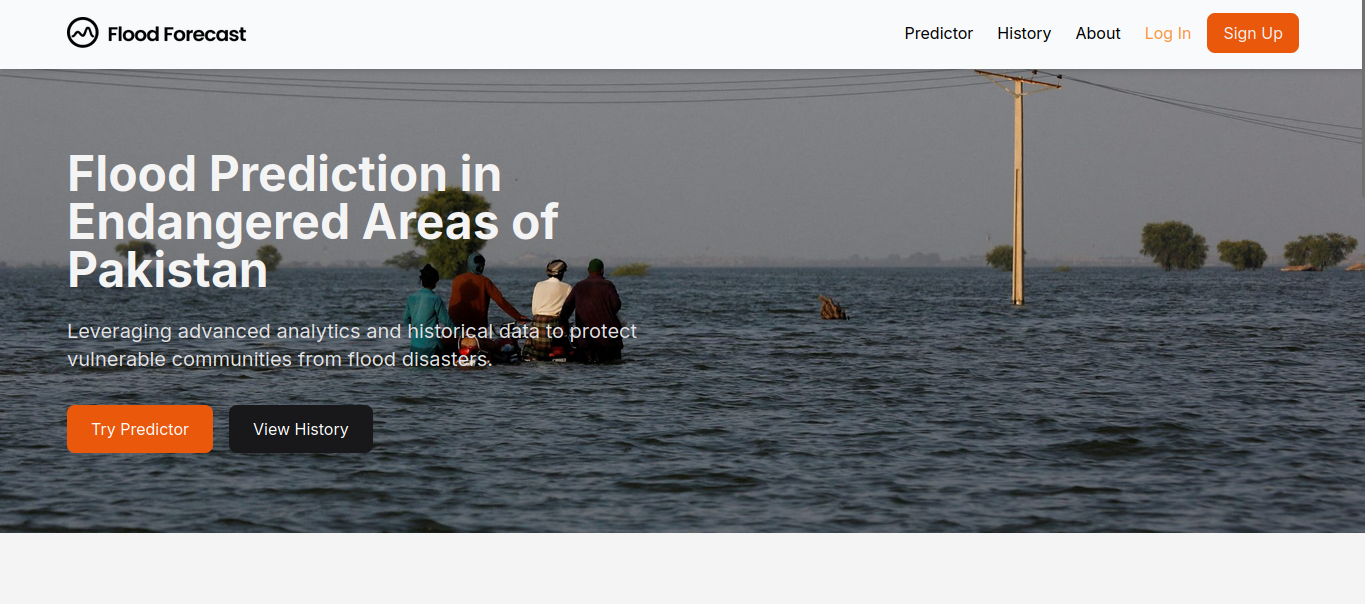


Fig 13. The main landing page of the Flood Forecast System, providing an overview of its purpose, key features, and navigation to core functionalities.

1. **Map Page (/predictor):** The central interface for flood risk visualization. It features an interactive map of Pakistan, instructional text ("Hover over the region to check the threat levels"), and a clear legend for risk levels (Safe, Slight Risk, Danger). Hovering over regions displays a tooltip with detailed risk information.

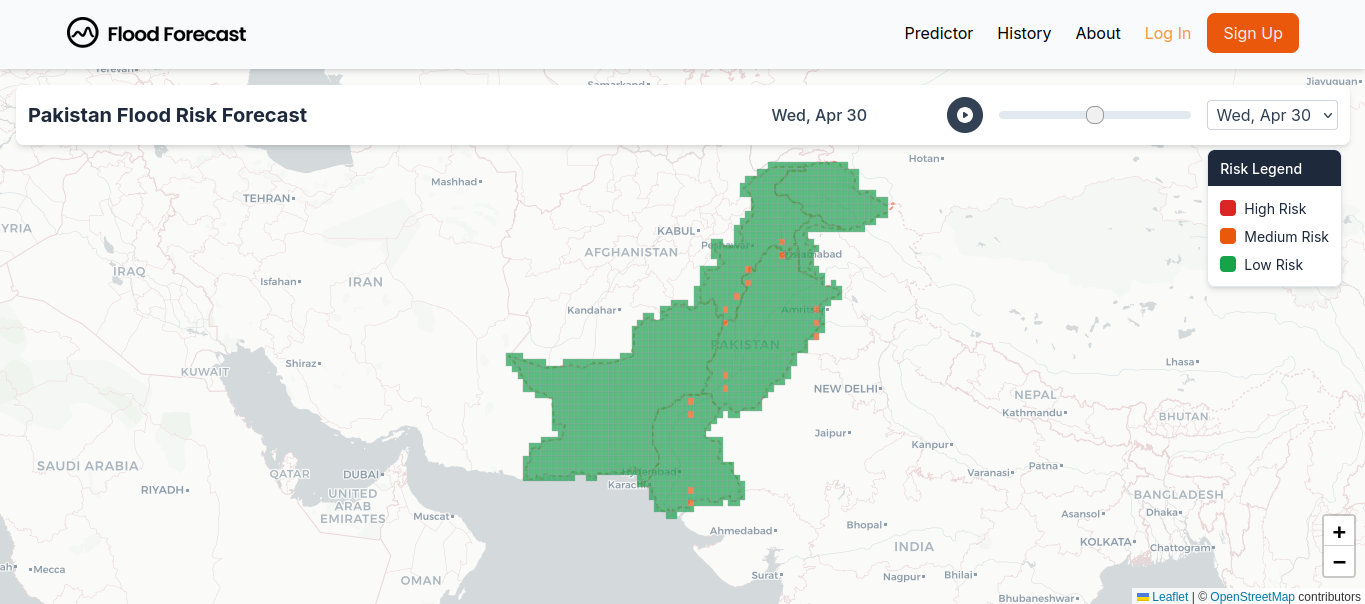


Fig 14. The interactive map view, displaying regional boundaries within Pakistan and the color-coded legend indicating different flood threat levels.

1. **About Page (/about):** Provides background information on the Flood Forecast project, including its objectives, methodology (ML, diverse data sources), and intended impact on disaster preparedness.

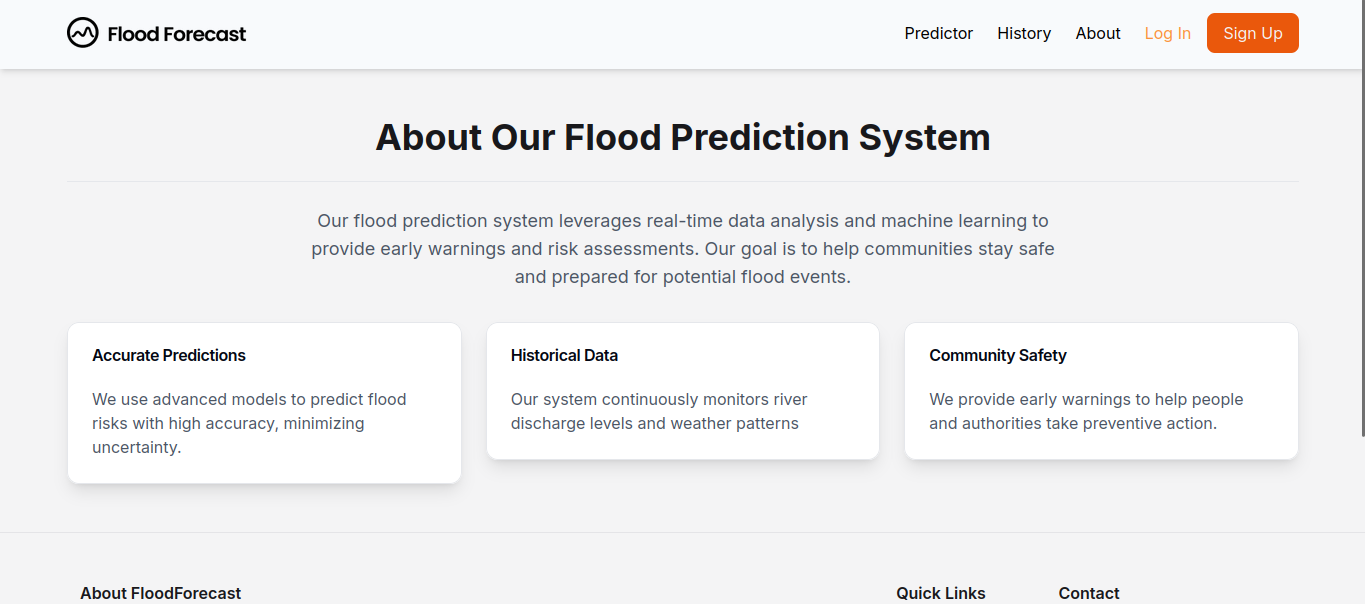


Fig 15. The "About" page, offering users detailed background information on the Flood Forecast project's objectives, methodology, and its significance for disaster preparedness in Pakistan.

1. **History Page (/history):** Designed to display historical flood data and analysis. The current live application features a placeholder; envisioned functionality includes interactive charts, timelines, and queryable data for past flood events.
2. **User Dashboard (Planned):** A personalized space for registered users, intended to display saved areas of interest, customized risk views, and alert management features.
3. **Login/Registration Pages (Planned):** Standard interfaces for user account creation and authentication.

## Interactive Map Functionality

The interactive map is a critical component for user engagement and risk assessment:

* **Basemap and Regional Display:** Displays geographical boundaries of Pakistan and its provinces.
* **Hover Interaction:** Moving the mouse cursor over a province highlights its boundary and dynamically displays a tooltip with key risk information: region name, threat level (color-coded), a descriptive summary, specific metrics (risk percentage, average rainfall), and a "Get more details" button (linking to planned detailed views).
* **Legend:** A clearly visible legend associates distinct colors with defined risk levels (Green: Safe, Yellow: Slight Risk, Red: Danger).
* **Zoom and Pan:** Standard map navigation capabilities, allowing users to zoom in/out and pan across the map, are provided by the underlying mapping library (Leaflet.js).
* **Layer Toggling (Planned):** Future functionality will allow users to toggle between different map layers, such as satellite imagery, terrain maps, and risk classification overlays, to enhance contextual understanding.
* **Underlying Technologies:** Leaflet.js is utilized for rendering map tiles, vector layers (province boundaries), and managing map interactions.

## Usability

The UI/UX design of the Flood Forecast System aims for optimal usability:

**Usability:**

* **Clarity and Simplicity:** The interface is designed to be clean and uncluttered, with straightforward navigation through a minimal set of header links ("About," "History," "Map").
* **Learnability:** Core interactions, such as hovering on the map for information, are intuitive and guided by on-screen prompts and a clear legend.
* **Efficiency:** The map page allows for efficient access to regional risk information through direct hover interactions.
* **Feedback:** The system provides immediate visual feedback for user actions, such as map highlighting and tooltip display on hover.
* **Accessibility (Design Goals):** The system is designed with WCAG 2.1 Level AA compliance in mind, including support for screen readers, keyboard navigation, high-contrast modes, and multilingual capabilities (English/Urdu). The live application demonstrates partial keyboard navigability for header elements.
* **Consistency:** A consistent visual style (fonts, colors, layout) is maintained across the implemented pages.

The current implementation provides a solid foundation for usability and basic responsiveness. Full realization of accessibility goals and advanced interactive features will require further development and rigorous testing.

# Conclusion and Future Work

## Project Summary

The Long-term Flood Prediction System for Pakistan is a comprehensive initiative designed to address critical challenges in national disaster preparedness. By integrating deep learning architectures, sophisticated geospatial analysis (utilizing PostGIS), and data integration from CDS (Climate Data Store), the system aims to deliver accurate, long-term flood risk predictions with a lead time of at least 30 days. Its component-based architecture, comprising Data Analysis, Machine Learning/Deep Learning, Geospatial Analysis, Web Application, and Database components, facilitates a modular and scalable solution. The user-facing web application, built with Next.js and React, is designed to provide an accessible and intuitive interface featuring interactive risk maps and (planned) personalized alerts for diverse stakeholders, including the general public, government authorities, and researchers. The project emphasizes robustness, scalability, security, and usability as core quality attributes.

## Challenges Faced

The development and implementation of a system of this complexity encountered several challenges:

1. **Data Acquisition and Quality:** Securing consistent, reliable, and high-resolution historical and real-time data from our source, along with ensuring data integrity through rigorous preprocessing, presents a significant hurdle.
2. **Model Complexity and Accuracy:** Developing deep learning models capable of accurate long-term flood prediction is inherently complex due to the multifaceted nature of hydro-meteorological systems. Achieving targeted accuracy and minimizing prediction errors requires extensive experimentation and computational resources.
3. **System Integration:** Ensuring seamless interoperability and data flow between the five major system components, as well as with external APIs, demands careful design and thorough testing.
4. **Scalability and Performance:** Architecting the system to handle large data volumes and support a high number of concurrent users while meeting stringent performance targets is a key challenge.
5. **User Interface and User Experience (UI/UX):** Translating complex scientific data into an intuitive and actionable format for a diverse user base, while adhering to accessibility standards, requires iterative design and user feedback.
6. **Scope Management:** Maintaining project focus and delivering a comprehensive suite of features within defined timelines and resource constraints is an ongoing consideration.

## Lessons Learned

The development process yielded these valuable insights:

1. **Primacy of Data Quality:** The accuracy and reliability of the underlying data are paramount for the success of any data-driven or machine learning project.
2. **Benefits of Iterative Development:** An iterative approach allows for incremental progress, early feedback integration, and effective risk management in complex system development.
3. **Advantages of Modular Architecture:** A component-based design enhances maintainability, facilitates parallel development efforts, and simplifies debugging and updates.
4. **Importance of Cross-Disciplinary Collaboration:** Effective communication and collaboration among team members with diverse expertise (software engineering, data science, geospatial analysis, UI/UX) are crucial.
5. **Necessity of Comprehensive Testing:** Rigorous multi-level testing (unit, integration, system, UAT, performance, security) is indispensable for ensuring system robustness and reliability.
6. **Value of User-Centric Design:** Prioritizing end-user needs and usability throughout the development lifecycle leads to more impactful and effective solutions.
7. **Significance of Realistic Planning:** Accurate estimation of effort and resources for each development phase is critical for successful project execution.

## Possible Enhancements

Future development and expansion of the Flood Forecast System may include:

1. **Advanced ML Models:** Incorporation of state-of-the-art models such as more complex Transformers for time-series forecasting or spatio-temporal analysis, and exploration of physics-informed neural networks.
2. **Expanded Data Variables:** Integration of more data variables from CDS so that model performance may be improved.
3. **Advanced Predictive Algorithms:** Development of capabilities for short-term, high-resolution flood inundation modeling and prediction of secondary impacts (e.g., disease outbreaks, detailed infrastructure damage).
4. **Enhanced User Interaction Capabilities:** Full implementation of user registration, personalized dashboards, advanced search and filtering functionalities, downloadable reports and data exports (CSV, GeoJSON, Shapefile), and potentially a dedicated mobile application.
5. **Sophisticated Geospatial Analysis:** Dynamic flood inundation mapping based on varying prediction severities, more detailed vulnerability assessments incorporating socio-economic factors, and integration of real-time sensor data.
6. **Third-Party API Integration:** Provision of a public API to enable researchers and other government agencies to programmatically access prediction data.
7. **Extended Multilingual Support:** Addition of support for more regional languages within Pakistan to improve accessibility.
8. **Community Engagement Features:** Development of mechanisms for users to report local conditions or validate predictions, fostering a collaborative approach to flood monitoring.

# References

* Castangia, M., Grajales, L. M. M., Aliberti, A., Rossi, C., Macii, A., Macii, E., & Patti, E. (2023).Transformer neural networks for interpretable flood forecasting. *Environmental Modelling & Software*, *160*(N/A), N/A. <https://doi.org/10.1016/j.envsoft.2022.105581>
* Demiray, B., & Demir, I. (2024, 06 01). *Towards Generalized Hydrological Forecasting Using Transformer Models for 120-Hour Streamflow Prediction*. SSRN. Retrieved May 10th, 2025, from https://papers.ssrn.com/sol3/papers.cfm?abstract\_id=4918783
* Dhital, S. (2024, August 5th). *Methods to improve run time of hydrologic models: opportunities and challenges in the machine learning era*. arXiv. Retrieved May 10th, 2025, from <https://arxiv.org/abs/2408.02242>
* Waseem, H. B., & Rana, I. A. (2023, 09). Floods in Pakistan: A state-of-the-art review. *Natural Hazards Research*, *3*(3), 359-373. https://doi.org/10.1016/j.nhres.2023.06.005