# HYDROSPHERE — Full System Architecture

FOR FULL FLOW REFER FLOW CHART –

BACKEND ONLY = https://www.mediafire.com/file/upkp648ajowr0qe/Untitled+diagram-2025-10-22-221823.png/file

FULL STACK (FRONTEND + BACKEND + MACHINE LEARNING FAST API ) =

https://www.mediafire.com/file/k8h1s7clo9756qa/Untitled+diagram-2025-10-22-224201.png/file

## FRONTEND

The frontend serves as the user-facing interface built using React.js.
It handles user authentication, visualization of flood risk data, and communication with the backend API.

Not all explained but few of them like for each tab or route have separate "jsx" file and have

Components of those files explaining all will make pdf messy.

| Component / File | Purpose / Functionality |
|---|---|
| SignUp.jsx | Collects username, email, phone, and password → Calls /auth/signup and /auth/send-otp. |
| Login.jsx | Handles user authentication → Sends /auth/login and stores JWT cookie. |
| ForgetPassword.jsx | Requests temporary password reset → POST /auth/forget. |
| Dashboard.jsx | Displays flood maps and AI predictions → Fetches from /flood/susceptibility and /ai/predict. |
| Redux & Context | Maintains global userData and token states. |
| Axios Service | Centralized HTTP handler (baseURL=http://localhost:5000/api/v1/hydrosphere). |
| Visualization Layer | Uses Recharts, Leaflet, and Framer Motion for interactive maps. |

## BACKEND — Node.js / Express Layer

The backend acts as the core API and orchestrator for all modules.
It connects the frontend, Python ML services, and external geospatial APIs such as Google Earth Engine.

• index.js — Initializes Express, applies middleware, and mounts routes.
• db.js — Connects MongoDB via Mongoose.
• authMiddleware.js — Verifies JWT tokens and attaches user ID.
• multerConfig.js — Handles file uploads securely.
• nodemailer.js — Sends OTP and reset password emails via Gmail/SMTP.
• pythonBridge.js — Connects Node to Python prediction service (spawn or FastAPI).
• geeClient.js — Handles Earth Engine data requests and integration.

## ROUTES — API Endpoints

Each route file organizes REST endpoints into specific modules, ensuring separation of concerns.

| | |
|---|---|
| authRoutes.js | /signup, /login, /send-otp, /verify-otp, /logout, /forget<br>→ Handles all authentication and OTP-based verification. Integrates with Nodemailer for email verification and password reset. |
| floodRoutes.js | /susceptibility, /export, /district/:id<br>→ Fetches and exports flood susceptibility data using Google Earth Engine (GEE) and Rudraprayag dataset integration. |
| aiRoutes.js | /predict — Forwards normalized hydrological features to the Python ML model for prediction.<br>→ Returns probability and flood risk class to the frontend dashboard. |
| postRoutes.js | CRUD endpoints: /post/create, /post/update/:id, /post/delete/:id, /post/all<br>→ Manages user-generated posts, image uploads (via Multer), and stores metadata in MongoDB. |
| feedbackRoutes.js | /feedback — Accepts user feedback with |

| | |
|---|---|
| | message and timestamp.<br>→ Saves entries into Feedback collection for admin review or analytics. |
| riskRoutes.js | /risk/zone, /risk/summary<br>→ Performs district-level flood risk classification and summary generation for reporting dashboards. |

## CONTROLLERS — Business Logic Layer

• userController.js — Handles registration, OTP verification, password reset, login, and logout.

• floodController.js — Integrates Rudraprayag dataset with Google Earth Engine flood maps.

• aiController.js — Forwards numerical features to Python ML model and returns predictions.

• riskZoningController.js — Performs zonation and classification of districts into risk categories.

• postController.js & feedbackController.js — Manages user-generated posts and feedback storage.

## MODELS — MongoDB Collections

Mongoose models define database schema for persistence in MongoDB Atlas.

• User.js: Stores user details (username, email, password, isVerified).

• TempUser.js: Temporary OTP-based user validation with expiry.

• Post.js: Stores posts with title, description, image path, and user reference.

• Feedback.js: Saves feedback messages from verified users.

• PredictionLog.js: Logs ML predictions with input features and probability scores.

## PYTHON — AI Prediction Layer

A Python-based machine learning service handles flood prediction using a pre-trained

Random Forest model.

| File | Description |
|---|---|
| prediction.py | Loads model.pkl, scaler.pkl, and feature_names.pkl, returns {probability, label}. |
| model.pkl | Trained RandomForestClassifier. |
| scaler.pkl | StandardScaler used for normalization. |
| feature_names.pkl | Maintains correct feature order for prediction. |

## DATA & EXTERNAL SERVICES

The system integrates with external APIs and datasets for real-time geospatial and hydrological analysis.

• Rudraprayag_Flood_Data.json: Local dataset containing elevation, slope, NDVI, rainfall, etc.

• Google Earth Engine: Provides remote sensing layers like Sentinel-2, CHIRPS rainfall, SRTM DEM.

• MongoDB Atlas: Main database for all models and logs.

• SMTP (Gmail / SendGrid): Used by Nodemailer for OTP and password reset emails.

• Browser Cookie Storage: Stores JWT token securely in HttpOnly cookies.

## END-TO-END FLOW SUMMARY

1. Signup Flow: User submits data → Backend creates record → OTP emailed → Verified → JWT cookie stored.
2. Login Flow: Credentials validated → JWT issued → Dashboard access granted.
3. Flood Analysis:Frontend requests data → GEE and local datasets merged → GeoJSON sent back.
4.AI Prediction:Frontend sends features → Python model predicts flood risk → Returns probability.
5. Feedback & Posts: User uploads feedback/posts → Stored in MongoDB.