Coastal Tide Tracker – Project Documentation

1. Project Overview

Project Name: Coastal Tide Tracker
Objective: Provide real-time location, nearest coastline, and upcoming high and low tide information with interactive maps and countdown timers.

Tech Stack:

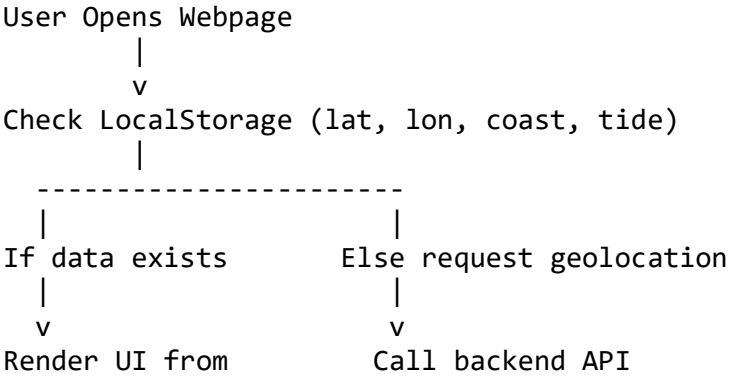
Layer	Technology
Frontend	HTML, CSS, JavaScript, Leaflet.js
Backend	Node.js, Express.js
Database	Optional (localStorage for caching)
APIs	Custom API for nearest coast + tide data

2. Features

- User Geolocation (latitude & longitude) detection.
- Nearest coastline calculation and visualization.
- Tide information display (next high and low tide).
- Countdown timer for tide events.
- Interactive map with markers and polyline.
- Local storage caching to avoid repeated API calls.
- Clear cache button for resetting stored data.
- Responsive UI for desktop and mobile.

3. System Architecture

Flow Diagram:



```
LocalStorage           Get coast & tide
                        Store in LocalStorage
        |                     |
        v                     v
 Display Map + Info + Countdown Timers
```

4. Backend Implementation

Endpoint: `GET /nearest-coast?lat={lat}&lon={lon}`

Response Example:

```
{
  "coastCoordinates": {
    "coastLat": 21.178,
    "coastLon": 72.789,
    "distance_km": 10.345
  },
  "TideData": {
    "hourly": {
      "time": ["2025-09-14T08:00:00Z", "2025-09-14T14:00:00Z",...],
      "wave_height": [1.2, 0.5,....7_days]
    }
  }
}
```

Backend Logic:

- Receive user coordinates.
- Calculate nearest coast using Haversine formula.
- Fetch tide data for nearest coast.
- Based on nearest coast coordinates fetch 7 day TideData(hourly) using marine-api
- Return JSON response with `coastCoordinates` and `TideData`.

5. Frontend Implementation

HTML Layout:

```
<div class="container">
  <div class="map-section" id="map"></div>
  <div class="info-section">
    <h1>Coastal Tide Tracker</h1>
    <div class="info" id="location"></div>
    <div class="info tide-card">
      <h3 id="nextHighTime"></h3>
      <div class="countdown" id="highCountdown"></div>
    </div>
```

```
    <div class="info tide-card">
      <h3 id="nextLowTime"></h3>
      <div class="countdown" id="lowCountdown"></div>
    </div>
    <div class="info" id="coast"></div>
  </div>
</div>
```

JavaScript Highlights:

- Geolocation with `navigator.geolocation`.
- Leaflet.js for map, markers, and polyline.
- Next tide calculation:

```
function getNextTideEvent(tideData) { ... }
```

- Countdown timer:

```
function startCountdown(targetDate, elementId) { ... }
```

- LocalStorage caching:

```
localStorage.setItem("lastLat", lat);
localStorage.setItem("lastLon", lon);
localStorage.setItem("coastCoordinates", JSON.stringify(coast));
localStorage.setItem("tideData", JSON.stringify(Tide));
```

- Clear cache button functionality.

6. UI Design

- **Desktop:** Map on left, tide info on right.
- **Mobile:** Map and info stacked vertically.
- **Elements:** `.tide-card`, `.countdown`, `.clear-btn`.
- **Colors:** Soft background, blue highlights for tide info.
- **Responsive:** Flexbox layout with media queries.

7. Approach & Implementation Steps

- Setup frontend layout with map & info sections.
- Initialize Leaflet map and fetch user location.
- Check LocalStorage → use cached data if available.
- If no cache, call backend API for nearest coast & tide data.
- Store fetched data in LocalStorage.
- Render map markers and polyline.
- Display next tide info with countdown.
- Implement clear cache button.

8. Advantages

- Faster load via cached data.
- Real-time interactive tide visualization.
- Mobile-friendly responsive design.
- Easy to extend with additional coastal info.

9. Future Enhancements

- Automatic cache expiration.
- Tide graphs for next 24 hours.
- Multi-coastline detection.
- Dark mode support.
- Offline caching with service workers.