Earthquake Prediction System Documentation

Developed by: CHOGE VINCENT

Date: May 04, 2025

A comprehensive guide to the Earthquake Prediction System, a web-based application for predicting earthquake metrics and visualizing data.

For support, visit: https://x.ai

Contents

1 Overview **2**

2 System Architecture **2**

2.1 Key Components 2

2.1.1 Database Models 2

2.1.2 Views 2

2.1.3 Templates 2

2.1.4 External Libraries 3

3 How It Works **3**

3.1 Data Input and Prediction Generation 3

3.2 Data Visualization 3

3.3 PDF Report Generation 4

3.4 Administration 4

4 Setup Instructions **4**

4.1 Prerequisites 4

4.2 Installation 4

4.3 Configuration 5

5 Usage Guidelines **5**

5.1 User Workflow 5

5.2 Developer Notes 5

6 Troubleshooting **5**

7 Future Enhancements **6**

8 Contact **6**

1 Overview

The Earthquake Prediction System is a web-based application built using Django that allows users to input earthquake data (latitude, longitude, magnitude, and depth) to generate predictions. The system provides a prediction dashboard, a data visualization interface, a real-world map display, and the ability to download prediction data as a PDF report. It leverages machine learning for magnitude prediction, geocoding for location names, and interactive visualizations using Chart.js and Leaflet.

- **Developed by:** CHOGE VINCENT
- **Date:** May 04, 2025
- **Technologies:** Python, Django, HTML, JavaScript, Chart.js, Leaflet, reportlab, NumPy, scikit-learn, geopy

2 System Architecture

The application follows a Model-View-Controller (MVC) pattern, implemented via Django's framework:

- **Models:** Define the data structure (`Earthquake` and `Prediction` models) stored in a SQLite database by default.
- **Views:** Handle HTTP requests and responses, including data processing, prediction generation, and PDF creation.
- **Templates:** Provide the HTML/CSS/JavaScript frontend for user interaction.
- **Static Files:** Include stylesheets (Bootstrap, custom CSS) and JavaScript libraries (Chart.js, Leaflet).

2.1 Key Components

2.1.1 Database Models

- **`Earthquake`:** Stores raw input data (latitude, longitude, magnitude, depth, date).
- **`Prediction`:** Stores predicted values (predicted magnitude, confidence, probability, accuracy, predicted location, prediction date) linked to an `Earthquake` instance.

2.1.2 Views

- `home`: Renders the homepage.
- `dashboard`: Displays earthquake inputs and predictions in tables with a map.
- `predict`: Handles form submission, generates predictions, and displays results with graphs and maps.
- `download_pdf`: Generates a landscape PDF with a table of all predictions.
- `admin_dashboard`: Provides an admin interface to delete records.
- `delete_earthquake/delete_prediction`: Handles deletion of records.

2.1.3 Templates

- `home.html`: Homepage with navigation.
- `dashboard.html`: Shows tables and a map of all data.
- `predict.html`: Form for input, prediction display, bar graphs, and map.

- `admin.html`: Admin interface for record management.

2.1.4 External Libraries

- **Chart.js:** For rendering bar graphs of prediction metrics.
- **Leaflet:** For displaying real-world maps with markers.
- **reportlab:** For generating PDF reports.
- **geopy:** For geocoding coordinates to location names.
- **NumPy/scikit-learn:** For simple linear regression-based magnitude prediction.

3 How It Works

3.1 Data Input and Prediction Generation

Process:

1. Users access the "Make Prediction" page (`/predict/`) and enter earthquake data:
   - Latitude (with direction: N/S)
   - Longitude (with direction: E/W)
   - Magnitude
   - Depth (km)
2. The `predict` view validates the input (e.g., latitude between -90 and 90).
3. It checks for duplicates within the last 30 days.
4. The data is saved as an `Earthquake` instance.
5. A predicted magnitude is calculated using a simple linear regression model based on historical data (if available; otherwise, it uses the input magnitude).
6. Confidence (80–95%) and probability (70–90%) are randomly generated as placeholders (to be replaced with a real model).
7. Accuracy is computed as the average of confidence and probability.
8. The predicted location is determined via geopy's Nominatim service (with a fallback to coordinates if geocoding fails).
9. A `Prediction` instance is saved with all calculated values.

Output:

- The exact predicted location is displayed.
- Small horizontal bar graphs show magnitude, confidence, probability, and accuracy.
- A Leaflet map displays the location with a marker.

3.2 Data Visualization

Dashboard (`/dashboard/`):

- Displays two tables:
  - "Recorded Earthquake Inputs": Lists all `Earthquake` data.
  - "Prediction Results": Lists all `Prediction` data, including accuracy.
- A Leaflet map shows markers for all predicted locations.

Graphs:

- Generated using Chart.js, showing metrics as horizontal bars for clarity.

3.3 PDF Report Generation

Process:

1. Users access the “Download & Print PDF” link (`/download_pdf/`).
2. The `download_pdf` view retrieves all `Prediction` instances.
3. A landscape PDF is created using reportlab, containing:
   - A title with the generation date.
   - A table with columns: Location, Magnitude, Confidence (%), Probability (%), Accuracy (%).
4. The PDF is sent as a downloadable file (“predictions.pdf”).

Features:

- Landscape orientation for wider table display.
- Text wrapping in the Location column to handle long names.
- Styled table with grid lines, centered text, and color coding.

3.4 Administration

Admin Dashboard (`/admin_dashboard/`):

- Allows deletion of `Earthquake` and `Prediction` records via POST requests.
- Displays all records in tables for manual review.

4 Setup Instructions

4.1 Prerequisites

- Python 3.13.2 (or compatible version)
- pip (Python package manager)
- Git (optional, for version control)

4.2 Installation

1. **Clone the Repository** (if applicable):
   ```
   git clone <repository-url>
   cd earthquake_project
   ```
2. **Create a Virtual Environment:**
   ```
   C:\Users\Butler\Desktop\EARTHQUAKE\earthquake_project\env\Scripts\activate
   ```
3. **Install Dependencies:**
   ```
   pip install django==5.2 numpy scikit-learn geopy reportlab
   ```
4. **Apply Migrations:**
   ```
   python manage.py makemigrations
   python manage.py migrate
   ```
5. **Run the Server:**
   ```
   python manage.py runserver
   ```
   Access the app at `http://127.0.0.1:8000/`.

4.3 Configuration

- **Database:** Uses SQLite by default (configured in `settings.py`). For production, modify `DATABASES` to use PostgreSQL or MySQL.
- **Static Files:** Ensure `STATIC_URL` and `STATICFILES_DIRS` are set in `settings.py`.
- **Logging:** Configured in `views.py` to log errors and info to the console.

5 Usage Guidelines

5.1 User Workflow

1. **Navigate to Home:**
   - Visit `http://127.0.0.1:8000/` for an overview and navigation links.
2. **Make a Prediction: Symptom**
   - Go to `http://127.0.0.1:8000/predict/`.
   - Enter earthquake data and click "Predict".
   - Review the location, graphs, and map.
3. **View Dashboard:**
   - Go to `http://127.0.0.1:8000/dashboard/`.
   - Check tables and map for all records.
4. **Download PDF:**
   - Go to `http://127.0.0.1:8000/download_pdf/`.
   - Download and print the landscape PDF with all prediction data.
5. **Admin Actions** (if authorized):
   - Go to `http://127.0.0.1:8000/admin_dashboard/`.
   - Delete records as needed.

5.2 Developer Notes

- **Extending Predictions:** Replace the random confidence/probability with a machine learning model (e.g., Random Forest) for better accuracy.
- **Geocoding:** Handle rate limits or failures by caching results or using a paid API (e.g., Google Maps).
- **Scalability:** For large datasets, optimize database queries or switch to a NoSQL database.
- **Testing:** Add unit tests for views and models using Django's testing framework.

6 Troubleshooting

- **PDF Not Downloading:**
  - Ensure `reportlab` is installed (`pip install reportlab`).
  - Check logs for errors in `views.py`.
- **Graphs/Map Not Loading:**
  - Verify internet access for CDN libraries (Chart.js, Leaflet).
  - Check browser console for JavaScript errors.

- **Database Errors:**
  - Run `python manage.py migrate` to apply schema changes.
  - Clear the database with `python manage.py flush` if needed.

7 Future Enhancements

- **Real-Time Data:** Integrate with seismic data APIs (e.g., USGS).
- **Advanced Models:** Use deep learning for more accurate predictions.
- **User Accounts:** Add authentication for personalized data tracking.
- **Mobile App:** Develop a companion app using Django REST Framework.

8 Contact

For support or contributions, contact CHOGE VINCENT at:

- Email: vincentchoge71@gmail.com
- Telephone: +254 726 390 388