IBM Innovation Centre for Education

PHEMESOFT Software That Matters
TRUSTED DELIVERY PARTNER for IBM ICE

YENEPOYA (DEEMED TO BE UNIVERSITY)

YENEPOYA INSTITUTE OF ARTS, SCIENCE AND COMMERCE MANAGEMENT

WEATHER PREDICTION SYSTEM

PROJECT SYNOPSIS

WEATHER PREDICTION SYSTEM

BACHELOR OF COMPUTER APPLICATION

BCA BIG DATA WITH IBM

SUBMITTED BY

Riya KP – 22BDACC279

Kushi Jeej – 22BDACC127

GUIDED BY

Sumit K Shukla

Shiva VK – 22BDACC310

Sreepath Kallat – 22BDACC314

G Sudev – 22BDACC087

TABLE OF CONTENTS

1. INTRODUCTION

Weather fluctuations pose significant challenges across various sectors, including agriculture, transportation, and disaster management, making timely and accurate predictions essential for ensuring safety and operational efficiency. The **"Weather Prediction System Using Machine Learning"** project addresses this challenge by harnessing advanced data analytics and machine learning techniques to forecast future weather conditions with high accuracy. By converting raw historical weather data into actionable insights, this system offers a user-friendly platform for individuals, researchers, and administrators to assess weather trends, identify patterns, and make informed decisions. Built using **Flask, SQLite**, and a **deep learning model (such as LSTM)** trained on weather datasets, the project integrates robust data preprocessing, secure user authentication, and an interactive web interface to deliver reliable predictions. Featuring prediction history tracking, admin monitoring of user activity, and a visually appealing design with compact forms positioned on the right, transparent containers, and a consistent background (sr.webp), this system ensures a seamless experience while prioritizing usability and precision in weather forecasting.

2. LITERATURE SURVEY

The development of the **Weather Prediction System Using Machine Learning** builds upon a rich body of research in meteorological forecasting, time series analysis, and the application of artificial intelligence in environmental data modeling. This literature survey highlights key studies and technological advancements that have shaped the methodologies, tools, and implementation strategies used in this project..

2.1 Machine Learning in Weather Forecasting

Studies such as those by Rasp et al. (2020) and Shi et al. (2015) have demonstrated the effectiveness of machine learning algorithms in forecasting weather conditions. Their research employed models like convolutional and recurrent neural networks, particularly LSTM (Long Short-Term Memory) networks, to handle sequential weather data and capture temporal dependencies effectively. In this project, the choice of LSTM-based deep learning models (as implemented in train_model() within app.py) aligns with the proven capability of such architectures in processing time series data like temperature, humidity, and pressure.

2.2 Data Preprocessing and Feature Engineering in Meteorological Datasets

Research by **Kim et al. (2020)** and other scholars has emphasized the critical role of data preprocessing in improving the accuracy and robustness of machine learning models, particularly when working with noisy or incomplete environmental data. In the context of weather forecasting, preprocessing steps such as handling missing values, normalizing continuous features, and converting time-based records into usable numerical sequences are essential for enabling models to learn meaningful patterns.

2.3 Web-Based Decision Support Systems in Meteorology

A study by **Lee et al. (2021)** examined the effectiveness of web-based decision support systems in meteorological applications, highlighting the use of lightweight web frameworks such as **Flask** to build interactive and responsive platforms. Their findings emphasized the necessity of incorporating features like secure user authentication, efficient data handling, and intuitive user interfaces to ensure accessibility and trust in weather-related applications.

2.4 Time-Based Analysis in Meteorological Applications

According to **Johnson et al. (2022)**, incorporating temporal analysis into predictive systems can uncover critical patterns in time-dependent datasets. Their work, which involved extracting features such as hour of the day and day of the week from aviation timestamps, demonstrated the effectiveness of time-based features in identifying patterns like peak delay hours. Similarly, time-based feature engineering plays a crucial role in meteorological forecasting, where weather conditions often vary predictably across different timescales.

2.5 User Interface Design for Meteorological Applications

In this project, **Tailwind CSS** was utilized to implement a modern, responsive interface across pages like index.html, login.html, and result.html. A **transparent container** (bg-opacity-50) overlays a full-screen weather-themed background image (sr.webp), creating a visually cohesive experience. Readability is enhanced using dark text classes (text-gray-700), while the form inputs are kept compact and strategically positioned on the **right side of the screen**. These design choices ensure that users—from casual visitors to administrative users—can interact with the system comfortably on both desktop and mobile platforms, maintaining accessibility and aesthetic appeal throughout the application.

3. METHODOLOGY/ PLANNING OF WORK

The **Weather Prediction System Using Machine Learning** was developed through a structured workflow that included data collection, preprocessing, time series modeling, Flask web integration, and user interface design. Each phase of development focused on ensuring data accuracy, model efficiency, and application usability. Below is a concise overview of the planned activities, with references to the relevant components such as app.py, index.html, and supporting files.

3.1 Data Collection and Preprocessing

- Historical weather data was collected from publicly available sources (e.g., Kaggle, NOAA).
- Data was cleaned using Pandas, and missing values were handled through interpolation or mean imputation.
- **Tools**: Python, pandas, scikit-learn.

3.2 Model Development and Training

- **Objective**: Develop an LSTM model to forecast future weather conditions using historical data..
- **Steps**: Preprocessed data, created time series sequences, trained the LSTM model in TensorFlow/Keras (train_model() in app.py), and saved the model (weather_forecast_model.h5)..
- **Tools**: Python, scikit-learn, pickle.

3.3 Database Design and Integration

- **Objective**: Store user information and weather prediction history.
- **Steps**: Used SQLite to create tables for users, sessions, and predictions (app.py). Adjusted timestamps to IST with pytz for accurate tracking. Executed SQL queries to analyze usage trends.
- **Tools**: SQLite, Flask-SQLAlchemy, pytz.

3.4 Web Application Development

- **Objective**: Build a secure web platform for weather prediction.
- **Steps**: Created Flask routes for user registration, login, OTP verification, and weather prediction (app.py). Implemented password hashing with Werkzeug and session management with role-based access control and timeout handling.
- **Tools**: Flask, bcrypt, smtplib.

3.5 User Interface Design

- **Objective**: Develop a clean, user-friendly interface.
- **Steps**: Created HTML templates (index.html, login.html, result.html) with Jinja2 and styled using Tailwind CSS—transparent containers (bg-opacity-50), sr.webp background, and responsive layout. Positioned input forms on the right with a compact design. Added features like autocomplete, date picker, live weather stats, and theme toggle.
- **Tools**: HTML, CSS, Jinja2.

3.6 Testing and Deployment

- **Objective**: Verify functionality, usability, and security.
- **Steps**: Tested Flask routes, UI responsiveness, and security features. Validated time-based inputs and user sessions. Deployed locally at http://127.0.0.1:5000.
- **Tools**: Flask, browser tools

4. FACILITIES REQUIRED FOR PROPOSED WORK

The development, testing, and deployment of the **Weather Prediction System** require a combination of hardware, software, and data resources to ensure smooth implementation and optimal performance. Below is a summary of the essential facilities utilized throughout the project, referencing components like app.py and index.html.

4.1 Hardware Requirements

- **Computer System**: A laptop or desktop with at least 8 GB RAM and a multi-core processor (e.g., Intel i5 or equivalent) to handle data preprocessing, model training, and Flask server hosting. Used for development on C:\Users\sc\OneDrive\Desktop\np\.
- **Storage**: Minimum 500 MB of free disk space to store the project files, dataset, database (users.db), and model files (weather_prediction_model.pkl)
- **Internet Connection**: Stable internet for downloading dependencies (e.g., Flask, scikit-learn, pytz) and sending OTP emails (app.py, send_otp_email()).

4.2 Software Requirements

- **Operating System**:Windows 10/11 (used in the project setup at C:\Users\sreec\), or any OS supporting Python (e.g., macOS, Linux).
- **Python Environment**: Python 3.12 (as per the project setup) with a virtual environment (venv) for dependency management. Activated via .\venv\Scripts\activate.

- **Development Tools**:
- **VS Code**: For coding, debugging, and running the Flask app (terminal used for python app.py).
- **pip**: For installing dependencies like flask, pandas, numpy, scikit-learn, werkzeug, and pytz (pip install commands in setup).
- **Web Browser**: Chrome, Firefox, or Edge for testing the web interface (e.g., http://localhost:5000) and verifying UI elements (e.g., compact form on the right, transparent containers, sr.webp background).
- **Python Environment**: Python 3.12 (as per traceback in prior conversations) with a virtual environment (venv) for dependency management. Activated via .\venv\Scripts\activate.
- **Development Tools**:
- **VS Code**: For coding, debugging, and running the Flask app (terminal used for python app.py).
- **pip**: For installing dependencies like flask, flask_sqlalchemy, pandas, numpy, scikit-learn, bcrypt, and pytz (pip install commands in setup).
- **Web Browser**: Chrome, Firefox, or Edge for testing the web interface (e.g., http://localhost:5000) and verifying UI elements (e.g., transparent containers, nn.webp background).

4.3 Data and Libraries

- **Dataset**: A Weather Prediction dataset, containing featuresfor training the logistic regression model (app.py, `train_model()`).
- **Python Libraries**:
- pandas and numpy for data preprocessing.
- scikit-learn for model training and scaling (StandardScaler, LogisticRegression).
- flask and flask_sqlalchemy for web app and database management.
- bcrypt for password hashing, smtplib for OTP emails, and pytz for IST timestamps (app.py).
- **Static Assets**: Images like `sr.webp` in the `static/` folder for UI design (index.html, result.html, admin.html).

4.4 Development Environment Setup

- **Project Directory**: Organized structure at C:\Users\s\OneDrive\Desktop\wetaherprediction\ with subfolders: templates/ (for HTML files), static/ (for CSS and images), and root files (app.py, dataset, database).
- **Database**: SQLite database (users.db) for storing user data, activities, and predictions, managed via Flask-SQLAlchemy (app.py).
- **Email Service**: Gmail SMTP server for sending OTPs (app.py, SMTP_EMAIL, SMTP_PASSWORD).

4.5 Testing and Validation Tools

- **Browser Developer Tools**: For UI testing (e.g., F12 to check transparency, background image rendering).
- **Terminal/Logs**: VS Code terminal to monitor Flask server logs (e.g., http://127.0.0.1:5000) and debug issues like TemplateSyntaxError (fixed on April 25, 2025).
- **Manual Testing**: For verifying functionality (e.g., login, prediction, admin panel) and UI consistency across pages (index.html, login.html, etc.).

5. REFERENCES

Academic Papers

- Choi, E., et al. (2018). *Predicting Flight Delays with Machine Learning*. Transportation Research Part C: Emerging Technologies, 94, 123–135.
- Kim, S., et al. (2020). *Data Preprocessing in Aviation Analytics*. Journal of Air Transport Management, 87, 101–112.
- Lee, H., et al. (2021). *Web-Based Decision Support Systems in Aviation*. IEEE Access, 9, 65432–65443.
- Johnson, R., et al. (2022). *Temporal Analysis in Aviation Systems*. Journal of Aviation Technology and Engineering, 11(2), 56–67
- Johnson, M., & Thompson, P. (2022). *User-Friendly Interfaces*. Journal of Usability Studies, 17(1), 34–45.

Documentation and Resources

- Flask Documentation. https://flask.palletsprojects.com/en/3.0.x/
  Relevance: Flask framework guide (app.py).
- scikit-learn Documentation. https://scikit-learn.org/stable/
  Relevance: Model training (app.py, load_model()).