Developer Documentation: Subjective Wellbeing Platform

This document provides a comprehensive overview of the **Subjective Wellbeing Platform**, including its architecture, folder structure, and functionality of its key components. It is intended to guide future developers in maintaining, extending, or replicating the platform.

1. Project Overview

The platform is designed to analyze and visualize subjective wellbeing data for the City of Melbourne. It enables stakeholders to explore trends, geographic patterns, and demographic insights using clean and interactive visualizations.

Features

1. **Interactive Dashboard**:
    - Visualize trends over time (2018–2023).
    - Includes filtering options for subtopics, age groups, gender, and suburbs.
    - Displays future predictions (2024–2025) using linear regression.
2. **Interactive Map**:
    - Visualizes geographic data using Folium.
    - Includes heatmaps and clustered markers with detailed popups.
3. **Data Viewer**:
    - Provides a table to view and filter raw survey data.
    - Enables filtering by subtopics, categories (age group, gender, suburb), and years.
4. **Responsive Web Design**:
    - Uses Flask to serve pages and Dash/Folium for interactivity.
    - Styled with Bootstrap for consistency and responsiveness.

2. Architecture

Core Components

- **Flask (Backend)**:
    - Manages routing and serves HTML templates.
    - Integrates with Dash for the interactive dashboard.
    - Hosts the Folium map for geographic visualizations.
- **Dash (Dashboard)**:
    - Generates interactive charts and graphs.
    - Provides filtering options for exploring trends in subjective wellbeing.
- **Folium (Map)**:
    - Displays geographic data with markers and heatmaps.
    - Dynamically updates based on user-selected filters.
- **Pandas (Data Processing)**:
    - Merges, cleans, and preprocesses six years of survey data (2018–2023).
    - Handles missing values using interpolation and median substitution.
- **Bootstrap (Frontend Styling)**:
    - Provides a consistent, responsive user interface.

3. Folder Structure

The project is organized into logical directories to ensure clarity and maintainability:

```
src/
├── app.py                    # Main Flask application
├── dash_app.py               # Dash app for the interactive dashboard
├── map.py                    # Folium map generation logic
├── templates/                # HTML templates for Flask
│   ├── index.html            # Home page
│   ├── dashboard.html        # Embedded Dash app page
```

```
│   ├── map.html              # Interactive map page
│   ├── data.html             # Data viewer page
├── static/                   # Static assets (CSS, JS, images, etc.)
├── data/                     # Data storage
│   ├── processed/            # Processed datasets
│   │   ├── subjective_wellbeing_cleaned.csv
│   ├── predictions/          # Predictions data
│       ├── predictions.csv
```

4. Key Components

4.1 `app.py`: Flask Application

This is the main entry point for the platform. It handles:

- Serving HTML templates for the home page, map, and data viewer.
- Integrating the Dash dashboard.
- Routing HTTP requests to the appropriate views.

4.2 `dash_app.py`: Dash Dashboard

This script defines the interactive dashboard. It:

- Embeds the Dash app into Flask using `url_base_pathname`.
- Provides dropdown filters (Subtopics, Age Groups, Gender, Suburbs).
- Displays a dynamic line graph showing trends and predictions.

4.3 `map.py`: Interactive Map

This script generates the geographic map using Folium. It:

- Visualizes geographic wellbeing patterns.
- Includes heatmaps to highlight wellbeing percentages.
- Uses clustered markers to display details about each location.

4.4 `templates/`: HTML Templates

Contains Flask HTML templates:

- **index.html**: Home page with navigation links.
- **dashboard.html**: Embeds the Dash app using an iframe.
- **map.html**: Displays the Folium map with dropdown filters.
- **data.html**: Renders the table-based data viewer with filters.

4.5 `data/`: Data Storage

Stores all datasets:

- **`processed/`**: Contains cleaned datasets ready for analysis.
- **`predictions/`**: Stores predictions for future trends.

5. Workflow

5.1 Data Cleaning and Integration

1. Import yearly datasets into Pandas.
2. Standardize column names and formats.
3. Handle missing values using interpolation or median substitution.
4. Merge datasets into a single file for analysis.

5.2 Visualization

1. **Dashboard**:
   - Filters data dynamically using Dash callbacks.
   - Displays trends with Plotly line graphs.
   - Includes predictions for 2024–2025.
2. **Map**:

- Filters data for a specific subtopic and year.
- Visualizes locations with clustered markers and heatmaps.

6. Running the Platform

6.1 Prerequisites

1. **Python**: Version 3.8 or higher.
2. Install required libraries:
   ```
   pip install flask dash pandas folium plotly dash-bootstrap-components
   ```

6.2 Starting the App

1. Navigate to the project folder:
   ```
   cd src
   ```
2. Run the Flask application:
   ```
   python app.py
   ```
3. Open the provided URL in your browser (e.g., `http://127.0.0.1:2022/`).

7. Future Enhancements

- **Export Feature**:
  - Add functionality to download filtered data as a CSV file.
- **User Accounts**:
  - Allow personalized dashboards and maps for registered users.
- **Advanced Visualizations**:
  - Include new chart types like bar graphs and scatter plots.
- **Performance Optimization**:
  - Use caching mechanisms for faster data loading.