

1. `app/__init__.py`

1. `create_app()` - Initialises the Flask application
2. `db = firestore.Client()` - Sets up configurations and database connections
3. `app.register_blueprint` - Registers blueprints for routes and authentication

Key functions:

- Creates Flask app instance
- Configures environment variables
- Initialises Firestore database
- Sets up CORS and authentication

2. `app/routes.py`

1. Handles all main application routes
2. Processes weather API requests
3. Manages review submissions and retrievals

Features:

- `WEATHER_ICONS` - Defines weather icons to display for varying weather conditions
- `/` - Home page with search
- `/search` - City weather and reviews
- `/api/weather/<city>` - Weather data
- `/api/reviews/<city>` - Review management
- `login / register` - Handles user access

3. `app/auth.py`

1. Manages user authentication
2. Handles JSON web tokens (JWTs)
3. Processes user registration and login

Features:

- `def register()` - Registration endpoint
 - Processes new user registrations, validates email and password inputs, checks for existing users to prevent duplicates, hashes passwords before storage, creates new user records in Firestore, and returns a JWT upon successful registration.
- `def login()` - Login endpoint
 - Authenticates existing users, verifies email and password combos, checks password hashes for security, and generates and returns JWTs for authenticated sessions.
- `token = jwt.encode` - Token generation

3. `app/utils.py`

Contains a validation function that just checks that user ratings are between the accepted bounds (1-5).

4. `templates/`

- `base.html` - Base template with navigation and styling
- `index.html` - Search interface
- `city.html` - Weather and reviews display
- `login.html` - User login form
- `register.html` - Registration form

These are just html files and can be very tedious to write and easy to make errors, so for these I tend to make use of AI tools to help in putting these together (as this is a simple task but easy to make errors on).

Key Features

1. Weather Information

OpenWeatherMap API integration

Displays current weather conditions

Shows temperature, humidity, wind speed, etc.

2. User Reviews

CRUD operations for reviews

Stored in Firestore

Includes ratings and comments

3. Authentication System

Secure user registration

JWT token-based authentication

Protected routes for authenticated users

Data Flow

1. Weather Data:

User Search → OpenWeatherMap API → Process Data → Display Results

2. Reviews:

User Input → Authentication Check → Firestore Storage → Display Reviews

3. User Authentication:

Other Files

`.env` contains keys for APIs and for my google cloud firestore access (secret key is just random):

```
WEATHER_API_KEY=aeb5c042704145fdc46fe60254d12fb8
```

```
SECRET_KEY=hello
```

```
GOOGLE_APPLICATION_CREDENTIALS=instance1-438012-68c10ad17369.json
```

These should all be left as they are.

`.gitignore` is a common file used to exclude chosen directories or files from being sent in a git push.

`generate_reviews.py` was just a python file I made to fill the database with reviews for the more common cities, so when we give a demo it is already populated with user reviews.

`requirements.txt` is obviously just the environment requirements in terms of python packages etc.

`instance1-438012-68c10ad17369.json` is my Google credentials file, which is used to access the Firestore database.

`run.py` is the function to start up the application. This will output a couple of URLs in the terminal that you can open in the browser to view and use the app website.

Database Structure

- `users`: User information and credentials
- `reviews`: City reviews and ratings