Weather App: Software Development Life Cycle (*SDLC*)

Developing a Weather app that displays current weather and allows users to submit their own weather reports if the official data is inaccurate. We will follow an Agile development approach, allowing for flexibility and iterative progress.

What are the phases of SDLC? *The typical stages include*:

1. **Planning/Requirement Analysis:** Gathering requirements, estimating costs and timelines, and defining the project's scope.
2. **Design:** Creating the architecture and blueprint of the software system.
3. **Development (Implementation/Coding):** Writing the actual code to translate the design into a functional product.
4. **Testing:** Reviewing and debugging the code to find and fix errors and bugs.
5. **Deployment:** Releasing the completed software to users or customers.
6. **Maintenance:** Providing ongoing support, gathering user feedback, and implementing updates for the software.

Phase 1: Planning & Requirements Analysis (1 Week)

Objective: *Define the project scope, identify core features, and understand the target audience.*

1. Define Project Vision:

- **Project Goal:** To create a simple, user-friendly weather app that provides accurate weather information and supplements it with crowdsourced data.
- **Target Audience:** General users who need quick weather updates and community members who want to report local weather variations.

2. Core Feature Identification (Minimum Viable Product - MVP):

- **Weather Display:**
    - Show current temperature, conditions (e.g., "Sunny," "Cloudy"), humidity, and wind speed for the user's current location (or a searched location).
    - Display a 5-day weather forecast.
    - Use a reliable weather API (e.g., OpenWeatherMap, WeatherAPI).

Location Search:

- Allow users to search for weather in different cities.

User-Submitted Weather:

- A simple form for users to submit the current temperature and conditions for their location.
- A way to view user-submitted reports on a map or list for a specific area.

3. Technology Stack Selection:

- **Frontend:** Kivy (using Python for a cross platform mobile and desktop application).
- **Backend & Database:** Firebase (Firestore for the database, Firebase Hosting for deployment) - this simplifies backend setup.
- **Weather Data**: A free weather API key.

4. High-Level Timeline:

- **Week 1:** Planning & Design
- **Weeks 2-3:** Development
- **Week 4:** Testing & Deployment
- **Ongoing:** Maintenance & Future Feature Iteration

Phase 2: Design (Concurrent with Week 1)

Objective: *Create the architectural blueprint and visual design for the app.*

1. System Architecture:

- **Client-Side (Frontend):** Kivy application that fetches data from the weather API and our firestore database using Python's library for the API and a Firebase library for Python (e.g., google cloud firestore) for the database.
- **Server-Side (Backend):** Firebase will handle user data storage. A weather_reports collection in Firestore will store user submissions.
- **Data Flow:**
    1. The app loads and gets the user's location.
    2. It calls the weather API for official data.
    3. It queries Firestore for recent user-submitted reports in that area.
    4. Both sets of data are displayed to the user.
    5. The user submits a new report, which is written to the Firestore database.

2. Database Schema (Firestore):
Collection: user_reports
Document: (*auto-generated ID*)

- city: string
- latitude: number
- longitude: number
- reported_temp: number
- reported_condition: string (e.g., "Rain", "Fog")
- timestamp: timestamp

3. UI/UX Wireframing:

- **Main View:** A clean interface showing the current weather prominently. A "Report Inaccuracy" button should be clearly visible.
- **Forecast View:** A simple list or series of cards for the next 5 days.
- **Submission Form:** A modal or separate page with fields for temperature and a dropdown for weather conditions.

Phase 3: Development (Weeks 2-3)

Objective: *Write the code for the application based on the design specifications.*

Sprint 1 (Week 2): Core Weather Functionality

- Set up the Kivy project.
- Integrate the chosen weather API to fetch and display current weather and the forecast.
- Implement the location search functionality.
- Build the main UI components for displaying weather data using Kivy widgets.

Sprint 2 (Week 3): User Submission Feature

- Set up the Firebase project and configure Firestore. ***(Integrate Python Firebase Library to connect your to my database.)***
- Create the weather submission form using Kivy.
- Write the logic to save user reports to the Firestore database.
- Develop the functionality to fetch and display user-submitted reports alongside the API data.

Phase 4: Testing (Week 4)

Objective: *Ensure the application is bug-free and meets user expectations.*

1. Unit Testing:

- Test individual functions, such as the API data fetching logic and form validation.

2. Integration Testing:

- Verify that the frontend correctly communicates with both the weather API and Firebase. Test the complete user flow from viewing weather to submitting a report.

3. Usability Testing:

- Ask a few friends or potential users to test the app. Is it easy to find information? Is the submission process intuitive?

4. Cross-Browser Testing:

- Ensure the app works correctly on major browsers like Chrome, Firefox, and Safari.

Phase 5: Deployment (End of Week 4)

Objective: *Launch the application and make it available to users.*

1. Build for Production:

- Package Kivy executable.

2. Deploy to Hosting:

- Deploy the application using Firebase Hosting. It's fast, secure, and integrates well with the rest of the Firebase suite.

3. Final Checks:

- Ensure the live API keys are correctly configured and the database security rules are in place to prevent unauthorized access.

Phase 6: Maintenance & Iteration (Ongoing)

Objective: *Monitor the app's performance, fix bugs, and plan for future features.*

1. Monitoring:

- Keep an eye on API usage to stay within free limits.
- Monitor the database for any unusual activity.

2. Bug Fixes:

- Address any issues reported by users or discovered after launch.

3. Future Feature Backlog:

- **V2 Ideas:**
  - User accounts to track submissions.
  - A map view to visualize user reports geographically.
  - Push notifications for weather alerts.
  - A system to "upvote" accurate user reports.