# Step By Step Process To Execute The Code

## Step 1: Setting Up the Development Environment

### 1. Install Flutter SDK

- Download the latest version of the Flutter SDK from the Flutter website.
- Extract the downloaded ZIP file to a preferred location on your system (e.g., C:\flutter on Windows).
- Add the Flutter SDK's bin folder to the system PATH variable:

  **Windows**: Go to "System Properties > Environment Variables" and add C:\flutter\bin to the PATH variable.

### 2. Install Command-Line Tools for Android

- Download and install Android Studio, which provides the Android SDK, essential for running the app on Android devices.
- Download the Android Command-Line Tools from the Android Developers website under the "Command-line tools only" section.
- Extract the ZIP file to a folder, such as C:\Android\cmdline-tools (Windows) .
- Inside the cmdline-tools folder, create a subfolder named latest and move the extracted files into it (cmdline-tools/latest/...).
- Install the required SDK components:Open a terminal and navigate to cmdline-tools/latest/bin.Run the following command to install essential Android SDK components:

  ./sdkmanager --install "platform-tools" "platforms;android-33" "build-tools;33.0.0"

### 3. Configuring Android Emulator

- Install the tools required to set up an emulator by running:

  ./sdkmanager --install "emulator" "system-images;android-33;google_apis;x86_64"
- Create a new Android Virtual Device (AVD):./avdmanager create avd -n myEmulator -k "system-images;android-33;google_apis;x86_64"
- Start the emulator using: ./emulator -avd myEmulator

4. **Verify Flutter Setup**
   - Run flutter doctor in the terminal. This command checks the Flutter installation and lists any additional requirements. Follow any suggestions to fix issues (e.g., installing missing SDKs or accepting licenses).

**Step 2: <u>Creating the Flutter Project</u>**

1. **Initialize a New Flutter Project**
   - Open the terminal and navigate to the desired folder. Run flutter create floodguard to generate a new Flutter project named **FloodGuard**.
2. **Configure Dependencies**
   - Open the project in a code editor like Visual Studio Code or Android Studio.
   - In pubspec.yaml, add dependencies for packages needed in the app, such as:
     - http: For handling API requests (e.g., OpenWeatherMap API for real-time weather data).
     - geolocator: For accessing device location.
     - provider: For managing state across the app.
   - Run flutter pub get to install the dependencies.
3. **Set Up API Access**
   - Register with OpenWeatherMap and get an API key.
   - Store the API key securely in the app's environment configuration for making requests to fetch real-time weather and flood data.

**Step 3: <u>Developing the App's User Interface</u>**

1. **Design the UI Structure**
   - Start by structuring key screens: Welcome Page, Flood Detection Page, Precautions Page, and Prevention Page.
   - Create widgets for each screen and design them to be visually appealing and easy to navigate.

2. **Add Visual Elements**
   - Add app icons, logos, and background images to enhance the visual appeal.
   - Implement styling themes, colors, and fonts to maintain a consistent look and feel across all pages.
3. **Build Core Features**
   - **Real-Time Data Display**: Use the OpenWeatherMap API to fetch and display weather data, including temperature, rainfall, and flood alerts.
   - **Flood Prediction**: Implement a flood risk model based on weather data and historical flood data to assess the likelihood of floods.
   - **Safety Tips**: Provide users with essential tips on flood preparedness and prevention.

**Step 4: <u>Testing the App on the Emulator</u>**

1. **Run the App on the Emulator**
   - Connect an Android emulator or physical device. Run flutter run in the terminal to test the app on the selected device.
   - Navigate through each screen, ensuring all UI elements are correctly displayed and functional.
2. **Fix Bugs and Optimize Performance**
   - Check for any errors, crashes, or layout issues on different screen sizes.
   - Optimize the code to ensure the app runs smoothly and is responsive across devices.

**Step 5: <u>Deploying the App to a Physical Device</u>**

1. **Enable Developer Mode on Android Device**
   - Go to **Settings > About Phone** and tap the **Build Number** multiple times to enable Developer Mode.
   - In **Developer Options**, enable **USB Debugging** to allow the device to connect to the computer for testing.

2. **Connect the Device and Install the App**
   - Connect the Android device to the computer via USB. Run flutter devices to confirm the device is recognized.
   - Execute flutter install to build and install the app directly onto the device.
3. **Conduct Final Testing on Device**
   - Test all features on the physical device, verifying functionality, UI layout, and response times.
   - Ensure real-time weather data updates, accurate flood predictions, and smooth navigation.