Phase 1: Account Management and Customization

- **Goal:** Establish the basic project structure, set up the necessary dependencies, configure CI/CD, and implement user authentication.

Tasks:

1. **Project Setup**
    - **Initialize Project Repository**: Set up the Git repository with README, license, and project structure.
    - **Install Dependencies**: Add necessary packages for frontend (React, Redux, etc.) and backend (Express, bcrypt, JWT, etc.).
    - **CI/CD Pipeline Setup**: Set up CI/CD pipeline (e.g., GitHub Actions) for automated builds and tests.
    - **Docker Configuration**: Create a Dockerfile and Docker Compose for easier deployment and development.
2. **User Authentication**
    - **Register API Endpoint**: Create the backend endpoint for user registration with encrypted passwords.
    - **Login API Endpoint**: Implement the backend endpoint for login, with JWT-based session management.
    - **Frontend Authentication**: Develop the login and registration pages with form validation.
    - **Token Storage and Session Handling**: Implement secure token storage (e.g., HTTP-only cookies) and session management.
3. **Basic UI Setup**
    - **Header and Navigation Bar**: Create a consistent header/navigation bar for logged-in users.
    - **Dashboard Skeleton**: Set up a simple layout for the dashboard page as a placeholder.

Phase 2: Weather Data Retrieval and Display

- **Goal:** Integrate the weather API, develop the search functionality, and build the dashboard with data visualization.

Tasks:

1. **Weather API Integration**
    - **API Key Management**: Set up secure storage and handling for the weather API key.
    - **Create Weather Service**: Implement a backend service to interact with the weather API and handle requests from the frontend.
2. **Weather Search Functionality**
    - **Backend Search Endpoint**: Create an endpoint to fetch weather data based on city name.
    - **Frontend Search Component**: Develop a search bar component on the frontend.
    - **Error Handling for Search**: Display error messages if a city is not found or the API call fails.
3. **Weather Data Display**
    - **Current Weather Display**: Create components to show the current temperature, conditions, and icons.
    - **5-Day Forecast**: Add components for a 5-day forecast, showing high/low temperatures and conditions.
    - **Unit Conversion**: Implement toggling between Celsius and Fahrenheit.

Phase 3: Account Management and Customization

- **Goal:** Enable users to manage their accounts and customize their dashboard settings.

Tasks:

1. **User Account Management**
    - **Edit Account Endpoint**: Implement API endpoint to allow users to update their profile info.
    - **Delete Account Endpoint**: Add a backend endpoint to securely delete user accounts.
    - **Frontend for Account Settings**: Create an account settings page to allow users to update/delete their accounts.
2. **Dashboard Customization**
    - **Unit Preferences**: Allow users to set Celsius/Fahrenheit as their default.
    - **Date Format Options**: Add settings to let users choose between different date formats.
    - **Layout Customization**: Implement basic layout customization (drag-and-drop to reposition components).

Phase 4: Favorite Locations and Dashboard Sharing

- **Goal:** Enable users to save favorite locations and share dashboards with others.

Tasks:

1. **Favorite Locations Management**
    - **Backend CRUD for Favorites**: Create endpoints to add, remove, and retrieve favorite locations.
    - **Favorites Display on Dashboard**: Show the list of favorite locations on the user's dashboard for quick access.
    - **Add/Delete Buttons for Favorites**: Implement buttons to add/remove cities as favorites on the frontend.
2. **Dashboard Sharing**
    - **Generate Shareable Links**: Implement backend logic to create unique shareable links for each dashboard.
    - **Frontend Sharing Options**: Develop a share button with options to copy a link or share on social media.
    - **View-Only Access for Shared Dashboards**: Restrict access on shared dashboards to view-only.

Phase 5: Testing, Optimization, and Deployment

- **Goal:** Conduct thorough testing, optimize performance, and prepare the application for deployment.

Tasks:

1. **Testing (Automated and Manual)**
    - **Unit Tests**: Write unit tests for user authentication, weather data retrieval, and favorite locations management.
    - **Integration Tests**: Test interactions between the frontend and backend, including user registration, login, and weather search.
    - **UI/UX Testing**: Conduct usability tests to ensure intuitive navigation and design consistency.
    - **Load Testing**: Simulate high traffic and large data loads to ensure the system's performance under load.
2. **Performance Optimization**
    - **Database Query Optimization**: Refine database queries for efficient data retrieval.
    - **Frontend Optimization**: Minimize JavaScript bundles, optimize images, and use lazy loading for components.
    - **API Caching**: Implement caching for frequently accessed data to reduce API calls.
3. **Deployment**
    - **Prepare for Deployment**: Finalize Docker configuration for deployment.
    - **Deploy to Production**: Deploy the application to a cloud platform (e.g., AWS, Heroku).
    - **Set Up Monitoring and Logging**: Enable monitoring tools to track application performance in production.