Feature Name: Flutter Mobile Application (Betty Boulder App)

Feature Overview:

The Betty Boulder Flutter mobile application will serve as a cross-platform interface (iOS and Android) that allows users to interact with the Betty Boulder AI assistant through voice and text chat. The mobile app will leverage the backend AI and MLS integration to provide localized real estate information for Boulder, including neighborhoods, listings, schools, zoning, trails, and other community features.

This is the first end-user deliverable of the project and will act as a demonstration platform for the AI assistant's capabilities.

Objectives:

1) Deliver a functional, user-friendly mobile interface for Betty Boulder.
2) Allow users to query real estate information through chat and/or voice.
3) Showcase seamless interaction with the backend AI and RAG systems.
4) Provide accurate and quick responses about Boulder-specific real estate topics.
5) Support testing, usability feedback, and iterative enhancement for future releases.

In-Scope:

1) Flutter-based mobile app for iOS and Android
2) Chat and voice interaction with the Betty Boulder AI assistant
3) Integration with backend APIs (for RAG model responses, MLS data, and caching)
4) User interface components: chat screen, settings screen, help screen
5) Display of text-based and map-based results (e.g., listings, neighborhoods)
6) Basic analytics (message count, latency tracking)

Out of Scope:

1) Real estate transactions (buying/selling functionality)

2) Expansion beyond Boulder-specific knowledge

3) Offline usage

4) Persistent user accounts

Functional Requirements

1) The app should allow users to send chat via text with Betty
Acceptance Criteria: User can type a message and receive a response within 3 seconds (with cached data).

2) The app should allow users to communicate with the AI assistant via voice
Acceptance Criteria: Voice button activates the microphone; app transcribes and sends input; assistant responds with synthesized voice.

3) Betty should be able to answer questions about Boulder properties through backend APIs and MLS data
Acceptance Criteria: Successful integration tested with live endpoints.

4) The app shall include a “Help” section with example queries
Acceptance Criteria: Accessible via side menu; lists 5 - 10 common questions.

5) Real estate agents should be able to add, edit, and delete property listings
Acceptance Criteria: Agents can successfully add, modify, and remove their own property listings, with all changes reflected in the live app within seconds.

6) Users should be able to filter through properties (price, location, type, etc.)
Acceptance Criteria: Listings include address, price, image, and link; neighborhoods show summary and map pin

7) Users should be able to connect with real estate agents if needed
Acceptance Criteria: Users can contact agents through chat or call directly from a property listing, and agents receive the inquiry in real time.

Non-Functional Requirements

1) Performance: Response time under 3 seconds for cached responses; under 7 seconds for non-cached responses.

2) Scalability: App must handle at least 500 concurrent users during testing.

3) Availability: 99% uptime for backend services during demo period.

4) Usability: Simple, intuitive UI requiring no prior instructions

5) Security: All communications are encrypted using HTTPS; no local storage of sensitive data.

6) Compatibility: Compatible with both Android and IOS; responsive design for both phones and tablets.

7) Maintainability: Code must be modular, documented, and follow Flutter best practices.

8) Reliability: App should gracefully handle API timeouts, network errors, or missing data.

9) Localization: Default language is English; future support for localization.

10) Accuracy: AI responses should have high accuracy rates (> 90%), when tested against verified Boulder-specific real estate.

Deliverables

- Flutter app source code (hosted in GitHub repo)
- Backend Go source code (hosted in GitHub repo)
- Build packages for demo
- API documentation and integration guide
- Test report and user feedback summary