Project Phoenix: Koralai v0.3.1 Development Schedule

Phase 1: "Atypical" Login and Privilege Handshake

Objective: Implement a mandatory, custom login screen that appears before the main browser window. This will handle a unique user ID and authenticate against a local or private network data source using an implicit, behavioral biometrics concept.

- **Rationale:** The current app's load_settings function in v0.1 and v0.2 provides a "hardcoded" user profile. We will replace this with a dynamic, authenticated state that dictates the user's access privileges. This fulfills the "atypical user login only status" requirement. The new version will experiment with a **behavioral biometrics login** as a unique alternative to traditional methods.

Phase 2: Add-on for Private Domain Communication

Objective: Create a dedicated add-on that houses the private community's UI and data logic. This add-on will serve as the "centralized 'activity' communication application."

- **Rationale:** By putting this functionality in an add-on, we maintain the core browser's integrity and allow the private domain to be uncrawlable by the public internet, as its content is never hosted on a public web server. The add-on's UI will render within the QSplitter panel from v0.2.

Phase 3: Centralized Data Management & Messaging

Objective: Implement the backend for secure, permission-based data access. This phase focuses on the social community' DB and a simple messaging system.

- **Rationale:** This establishes the core functionality of the private domain. Messages and activity logs will be tied to a central data source ID, allowing for a personalized and secure experience. This prepares for future features like "gated community" content.

Phase 4: Gated Content & 'Gated Community' Markup

Objective: Add functionality to the add-on to display privileged content and manage user permissions within the private network.

- **Rationale:** This is where the concept of uncrawlable mark up 'class' becomes a reality. The add-on will be responsible for parsing data and dynamically generating the UI based on the user's privilege level, ensuring content is never exposed to the public web engine.

Prototype Dev File Comparison Chart

Feature	koralai.v0.1.py (Current)	koralai_web_browser.v0.2.0.py (Current)	Project Phoenix v0.3.0 (Next Version)	Project Phoenix v0.3.1 (Implicit Login Prototype)
Login/Auth	load_settings() on startup. No user-specific logic.	load_settings() on startup.	**Custom Login Add-on** at boot. Authenticates against a private database. User privileges are set.	**Implicit Login Add-on** using timed, behavioral biometrics.
Core UI	Single web view with tabs.	Web views with a QSplitter for add-on panels.	**Web views + a private, local UI** in the add-on panel.	Same as v0.3.0, but with a unique login screen.
Data Source	Public internet only.	Public internet + read-only access for add-ons (get_page_text).	**Private network/database** accessible via the Add-on only.	Same, with local JSON as a data source for the login pattern.
Data Security	None.	None.	**Secure, authenticated access** to private data. Content is not publicly crawlable.	**Behavioral pattern recognition** replaces static credentials.
Core Function	Simple web browser.	Extensible browser via add-ons.	**Centralized communication app** within	Same, with a focus on an experimental

			a secure browser shell.	login method.
AI Integration	Not applicable.	KoralaiBridge provides basic text retrieval.	**Two-way KoralaiBridge** to a local LLM or API for messaging and data analysis.	Same. The login method itself can be a "training" data source for an AI.

Minimum Requirements & Checkpoints

- **Check-in 1:** Successful implementation of the Login Add-on. The application must not proceed to the browser until a hardcoded user_id is successfully entered.
- **Check-in 2:** The Add-on panel displays a basic "Welcome,Username
  !" message upon successful login.
- **Check-in 3:** The add-on can successfully fetch and display a message from a local JSON file that simulates the private domain DB.
- **Check-in 4:** The add-on can send a message from the user to the local JSON file.
- x
  Check-in 5 (New): A prototype login add-on is created that uses a non-password-based authentication method. This meets the new login concept requirement.

Comments on Design Obstacles:

- private domain Selection: A major obstacle is how to securely manage which domain/backend an add-on connects to. The current manifest.json approach is simple. For a secure application, this would need to be encrypted or managed by a core configuration that only the KoralaiMainWindow can access, passed securely to the add-on.
- Ai assist / Ai Training DB relay: This requires a significant modification to the KoralaiBridge. It needs to become a two-way communication channel, allowing add-ons to send prompts to an AI and receive structured data in return. This goes beyond the current get_page_text functionality.