Project Requirements (Partial Draft)

Kory Singleton

Custom Keyboard Development for Refugee Language Accessibility

1. Purpose of the Project

The purpose of this project is to design and develop custom Android and iOS keyboards that support a refugee community's native alphabet, enabling them to digitally read, write, and interact with Bible translations in their own language. This project bridges the gap between technology and culture by giving a community the means to engage with Scripture and personal communication in a written form that reflects their identity.

2. Goals of the Project

The main goal is to create an input system that is both *functional* and *authentic* to the community's writing system.

- **Purpose:** To provide a Unicode-compliant keyboard for Android and iOS devices that includes the community's native characters.
- **Advantage:** This will allow translators and native speakers to collaborate on Bible translation efforts, preserve their written language digitally, and increase literacy engagement.
- **Measurement:** Success will be measured through accuracy testing (keyboard inputs producing correct glyphs), community usability feedback, and the number of active users engaging with the translation software post-launch.

3. Stakeholders

Client: The Bible translator leading the refugee language translation effort. **Users:** Native speakers of the refugee community who will use the keyboards to read and write in their own language.

Developers: The student project team responsible for design, implementation, and testing of the custom keyboards.

Support Stakeholders: Linguists or SIL International consultants verifying script accuracy and encoding standards.
Priorities: The translator and native speakers hold the highest priority, as their needs directly define usability and accuracy requirements.

4. Constraints

- **Solution Constraint:** The keyboards must be compatible with Android (API Level 26 and above) and iOS (version 14 and above).
- **Budget Constraint:** Development will use free or open-source frameworks (e.g., Keyman Developer or kbdgen) to reduce cost.
- **Schedule Constraint:** All design, implementation, and testing must be completed within one academic semester (~12 weeks).
- **Enterprise Constraint:** The application should comply with standard Unicode encoding (no custom fonts that replace ASCII mappings) to ensure long-term sustainability.

5. Functional Requirements

1. The system shall allow the user to input all characters of the custom alphabet using both lowercase and uppercase forms.
2. The keyboard shall include support for diacritics or combining marks where applicable.
3. The user shall be able to switch between the custom keyboard and the system default keyboard easily.
4. The keyboard layout shall display correct glyphs when tested in common text fields (e.g., Notes, WhatsApp, Chrome).
5. The keyboard shall store user settings such as preferred layout orientation and theme.

6. Non-Functional Requirements

- **Usability:** The keyboard must be intuitive enough for new users to learn within 10 minutes of use.
- **Performance:** Keypress latency should not exceed 150 milliseconds.
- **Accessibility:** The keyboard should support right-to-left or left-to-right text direction, depending on the language's writing system.
- **Reliability:** The application should maintain stable performance on at least 90% of tested devices.
- **Portability:** Layout and configuration files should be platform-independent, allowing updates without a full rebuild.

7. Risks and Assumptions

Risks:

- The community's language may use non-standard or unencoded characters, requiring custom Unicode proposals.
- Limited device access may restrict testing across multiple hardware configurations.
- Inconsistent feedback from users due to geographic or internet limitations.

Assumptions:

- All required characters exist in Unicode or can be represented through combining marks.
- The translator will assist in verifying script accuracy and linguistic rules.
- The final keyboard will be distributed through direct download or open-source release rather than app stores for simplicity.

8. Expected Benefits

- Empowers a displaced community to use their native language digitally.
- Enhances collaboration in Bible translation efforts.
- Provides a reusable framework for future linguistic or cultural projects.
- Strengthens cross-platform technical literacy within the community.

9. Conclusion

At its core, this project is about accessibility and identity. Creating custom keyboards for a refugee language doesn’t just solve a technical issue—it gives people the ability to express themselves in their own words. By addressing both the linguistic and technological challenges, this project aims to deliver a sustainable and meaningful solution that connects faith, language, and technology in one design.