# **📄 System Requirements Specification (SRS)**

**Project Name:** Defeah Bible Study  
 **Version:** 1.0  
 **Author:** Fenn Ignatius Saji  
 **Date:** July 2025

## **1. ✅ Functional Requirements**

| **ID** | **Feature** | **Description** |
| --- | --- | --- |
| FR-01 | Input Query | Users can input either a Bible reference (e.g., “John 3:16”) or a topic (e.g., “forgiveness”). |
| FR-02 | Guide Generation | The system will generate a contextual study guide using an AI model trained or tuned on Bible content. |
| FR-03 | Study Guide Output | Each guide includes: Summary, Historical Context, Related Verses, Reflection Questions, and Prayer Points. |
| FR-04 | Response Storage | Recent guides (5–10 per user) are stored locally or in the cloud for reuse. |
| FR-05 | Shareable Format | Output can be shared via WhatsApp, email, or copyable text. |
| FR-06 | Optional Authentication | Google/Apple login (via Firebase) allows syncing of saved content across devices. |
| FR-07 | Jeff Reed Study Flow | For predefined topics, users can access a 4-step structured study: Context, Scholar Guide, Group Discussion, and Application. |
| FR-08 | Feedback Submission | Users can rate the relevance of guides and leave written feedback. |
| FR-09 | Topic Localization | Predefined Jeff Reed topics are localized and cached client-side. |

## **2. 🔒 Non-Functional Requirements**

### **⏱️ Performance**

* LLM response time must be under 3 seconds on average.
* Cached or repeated queries should return in under 1 second.

### **🔐 Security**

* Use Firebase Auth or Supabase Auth for secure login sessions.
* Encrypted HTTPS communication enforced across all endpoints.
* Protect against prompt injection and prompt hijacking by sanitizing user inputs.
* PII storage is minimized; access is authenticated and role-based.

### **⚙️ Scalability**

* Support at least 50–100 concurrent users initially.
* Architecture should scale to 5,000+ users via autoscaling (Cloud Functions, Supabase Edge Functions).
* LLM backend should be modular to support provider switching (OpenAI, Claude, etc.).
* Portable backend to work on Firebase/GCP or Supabase without vendor lock-in.

## **3. 📘 Use Cases and User Stories**

### **✅ Use Case 1: Generate Bible Study Guide**

**Actor:** Logged-in or anonymous user  
 **Steps:**

1. User inputs "Romans 12:1" or "faith"
2. App checks cache (client/server-side)
3. If not cached, backend sends query to LLM
4. Structured response returned to frontend
5. User views, saves, or shares guide

### **✅ Use Case 2: Save and Revisit Guides**

**Actor:** Logged-in user  
 **Steps:**

1. User generates a study guide
2. App stores the guide\_id in user history
3. On future login, the user can revisit and re-share the saved guides

### **✅ Use Case 3: Jeff Reed Guided Study**

**Actor:** Logged-in or anonymous user  
 **Steps:**

1. User selects predefined topic like “Grace”
2. App fetches cached or fresh 4-step guide
3. Each step (Context, Scholar Guide, Discussion, Application) is displayed individually
4. User completes steps progressively; timestamps are optionally logged

### **✅ User Story: Pastor Daniel**

“As a pastor, I want to input a verse and get a concise but deep explanation I can use for my Sunday sermon.”

### **✅ User Story: Sister Anjali**

“As a churchgoer, I want to input a topic and receive a short reflection and prayer that I can share with my prayer group.”

### **✅ User Story: Dev Team**

“As a developer, I want to ensure the backend is low-latency and cost-efficient, so we don’t need to scale infra manually.”

Let me know if you'd like a diagram for any of the flows above or turn this into a testable checklist.