Project Scope Document

1. Project Overview

The objective of this project is to migrate a legacy software module built using Microsoft Silverlight (XAML) and .NET (C#) into a modern tech stack comprising React (frontend) and Python (Django for backend). AI-assisted tools and workflows will be used to accelerate translation, ensure code quality, and preserve feature parity. You will drive this initiative by leveraging large language models (LLMs), code converters, and test generators to modernize the module efficiently.

2. Objectives of the AI Consultant

- Analyze and understand the existing legacy codebase and architecture.
- Use AI tools (e.g., GPT-4, Copilot, Codex) to convert and refactor:
  - XAML UI components → React JSX
  - C# ViewModels/business logic → React Hooks + Python APIs
  - ADO.NET/Entity Framework DB logic → SQLAlchemy ORM
- Assist in regenerating test suites using AI for both frontend and backend.
- Document and establish repeatable AI-assisted migration processes.
- Guide integration, testing, and validation to ensure the new system meets functional parity.

3. Deliverables

3.1 Source Code Translation Deliverables

Area	Legacy Technology	Target Stack	AI Consultant's Responsibility
UI	Silverlight (XAML)	React (JSX) + Tailwind	Translate UI structure and states using LLMs
Logic Layer	C# (ViewModels, Controllers)	React Hooks + Python (FastAPI/Django)	Convert logic and workflows
Service Layer	WCF/SOAP	RESTful APIs	Translate contracts and data flows
Database	ADO.NET / EF	SQLAlchemy / Django ORM	Map DB schema and queries

3.2 Testing Deliverables

- AI-generated unit tests using:
  - React Testing Library + Jest for frontend
  - Pytest for backend
- Edge case identification and test coverage recommendations
- Functional test case review against legacy behavior

3.3 Process & Tooling

- Set up and document AI-powered migration workflows (e.g., prompts, LLM configs)
- Recommend tools:
  - GPT-4 / Claude / Copilot / Refact.ai
  - CodiumAI / Testim (for test generation)
  - Swagger/OpenAPI for API validation
- Establish reusable prompt patterns for code translation

3.4 Documentation

- Technical migration summary: mapping old codebase to new architecture
- Prompt libraries and examples for future migrations
- Test cases and result summaries
- Git repository structure for frontend and backend

4. Constraints & Assumptions

- Consultant will work with a provided legacy module only (not the full system).
- Source code will be accessible in a secure Git repository.
- Access to AI tools (OpenAI, GitHub Copilot, etc.) will be provided or arranged.
- Consultant will work alongside in-house dev teams (React and Python).

5. Timeline & Milestones

Phase	Timeline	Key Outputs
Week 1	Codebase Familiarization	Architecture map, scope definition
Week 2–3	UI + Logic Translation	JSX components, API skeletons
Week 4	Backend Completion	DB models, APIs, tests
Week 5	Integration + Testing	Functional flows, test suites
Week 6	Documentation & Handover	Prompt library, migration report

6. Success Criteria

- Feature parity between legacy and modernized module
- Readable, maintainable React + Python codebase
- Minimum 80% test coverage via AI-generated tests
- Documentation suitable for future module migrations

7. Optional Enhancements (If time permits)

- Auto-documentation of APIs using Swagger or Redoc
- Integration of LLM as a dev assistant in the CI pipeline
- Proof of concept: future module migration with zero-touch AI workflow