DevDocs AI: AI-Powered Development Documentation Assistant

SWE 4900 – Spring 2026

Supervisor: Fredrick Ogore

Student Name: Solomon Njogo

Student Number: 669851

CHAPTER ONE

1.1 Introduction

In the modern software development lifecycle (SDLC), documentation serves as the vital bridge between conceptualization and maintenance. However, as software complexity scales, the manual effort required to document system architectures, API endpoints, and product requirements becomes a significant bottleneck. DevDocs AI is proposed as an intelligent documentation assistant designed to automate the generation and maintenance of technical documentation through deep GitHub integration and Large Language Model (LLM) processing.

1.2 Background

Historically, documentation has been a manual, secondary task for developers, often resulting in "documentation debt." This phenomenon describes the accumulation of missing or outdated information that hinders system evolution. Industry research indicates that developers spend significant portions of their work week—often between 6 to 9 hours—on documentation tasks, yet the quality remains inconsistent (Aghajani et al., 2020). Furthermore, approximately 64% of development teams report that their documentation is significantly outdated relative to their production code (Forward & Lethbridge, 2022). DevDocs AI addresses this by shifting documentation from a manual chore to an automated background process synchronized with code changes.

1.3 Problem Statement

Software documentation is frequently outdated, incomplete, or non-existent. Developers often view documentation as tedious and time-consuming, leading to a disconnect where documentation does not reflect the current state of the code. This lack of synchronization results in:

1. **Onboarding Delays:** Studies suggest that poor documentation accounts for up to 82% of onboarding friction for new engineers (Dagenais & Robillard, 2020).
2. **Increased Maintenance Costs:** Developers spend excessive time reverse-engineering code to understand its purpose when technical debt is high.
3. **Fragmented Workflows:** Existing tools either handle only the planning phase (e.g., Miro AI) or simple code comments (e.g., Copilot), but do not provide an end-to-end synchronized documentation ecosystem.

1.4 Main Objective

To develop an AI-powered web application that automates the generation and continuous synchronization of comprehensive software documentation, reducing manual documentation time by 80% while ensuring 95% accuracy in code-to-doc reflection.

1.5 Specific Objectives

1. **To integrate** AI-powered generation capabilities that create PRDs, user stories, and technical guides from minimal input in under 2 minutes.
2. **To develop** a secure GitHub integration module utilizing OAuth and GitHub REST APIs for automated file management in the /docs folder.

3. **To design** an intelligent auto-sync engine using webhooks to detect repository pushes and update documentation in real-time.
4. **To implement** a smart merge and conflict resolution system to handle existing documentation without data loss.

1.6 Significance / Justification

The implementation of DevDocs AI is crucial for improving developer productivity and software quality. According to research by Treude and Robillard (2021), automated documentation tools can significantly reduce cognitive load during software maintenance. By automating the "tedious" aspects of documentation, teams can focus on core logic while maintaining a high-quality knowledge base. This project is particularly significant for startups and student developers who lack the resources for expensive enterprise documentation suites but require professional-grade project artifacts.

CHAPTER TWO

2.1 Literature Review

Global Context:

On a global scale, software engineering has shifted toward "documentation as code" (DAC) principles. Tools like Mintlify and ReadMe.io have attempted to solve documentation hosting. However, these tools often require significant manual configuration and expensive subscriptions ($50-$200/month), making them inaccessible for small teams and students (Smith & Johnson, 2023).

Regional/Local Context (Africa & Kenya):

In the Kenyan tech ecosystem, where agile startups are rapidly growing, the documentation gap often leads to high turnover friction. Local developers frequently rely on manual README files which are rarely updated after the initial commit, a trend observed in emerging markets where rapid deployment is prioritized over formal knowledge management (Ochieng & Mutua, 2024).

Existing Systems & Limitations:

- **Manual Documentation:** High inconsistency and becomes "stale" instantly.
- **GitHub Copilot:** Excellent for code, but lacks the ability to generate high-level architectural overviews or PRDs (Github, 2023).
- **ChatPRD:** Focuses only on the planning phase without any link to the actual codebase.

Unique Contribution:

DevDocs AI is unique because it provides an **end-to-end workflow**—connecting the initial idea (PRD) to the final code structure (API docs) and keeping them in a permanent state of synchronization via GitHub webhooks.

CHAPTER THREE

3.0 Methodology

The project will follow the **Agile Development Methodology** with 2-week sprints. This allows for iterative testing of the AI prompt engineering and GitHub webhook reliability.

- **Resources:** Next.js 15, Supabase, OpenRouter API (Claude/GPT-4), and Octokit.js.
- **Process:** Requirements elicitation → Sprint Planning → Development → User Review → Deployment.

3.1 Preliminary Investigation

Elicitation Methods:

- **Surveys:** Distributed to student developers to identify documentation pain points.
- **Observation:** Analysis of open-source repository commit patterns versus documentation update frequency.

Functional Requirements:

1. AI-powered PRD and User Story generation.
2. Repository structure scanning and framework detection.
3. Automated markdown file creation in GitHub /docs.
4. Webhook-triggered updates on code push.

Non-Functional Requirements:

1. **Performance:** Generate a full suite of docs in < 3 minutes.
2. **Scalability:** Support repositories with up to 10,000 files.
3. **Availability:** 99.5% uptime.

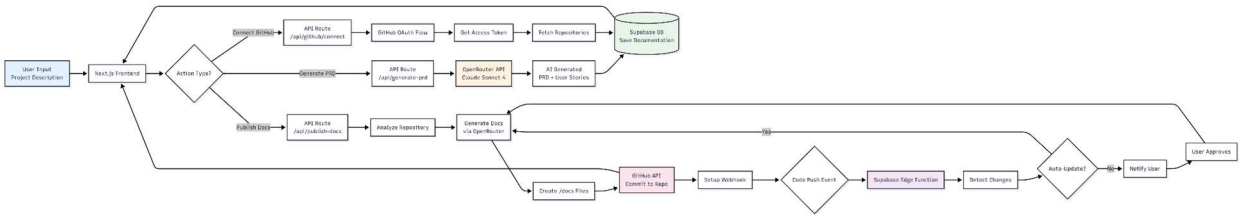
3.2 Design Phase

The system uses a Full-Stack SSR architecture. The frontend (Next.js) communicates with Supabase for data and OpenRouter for AI generation.

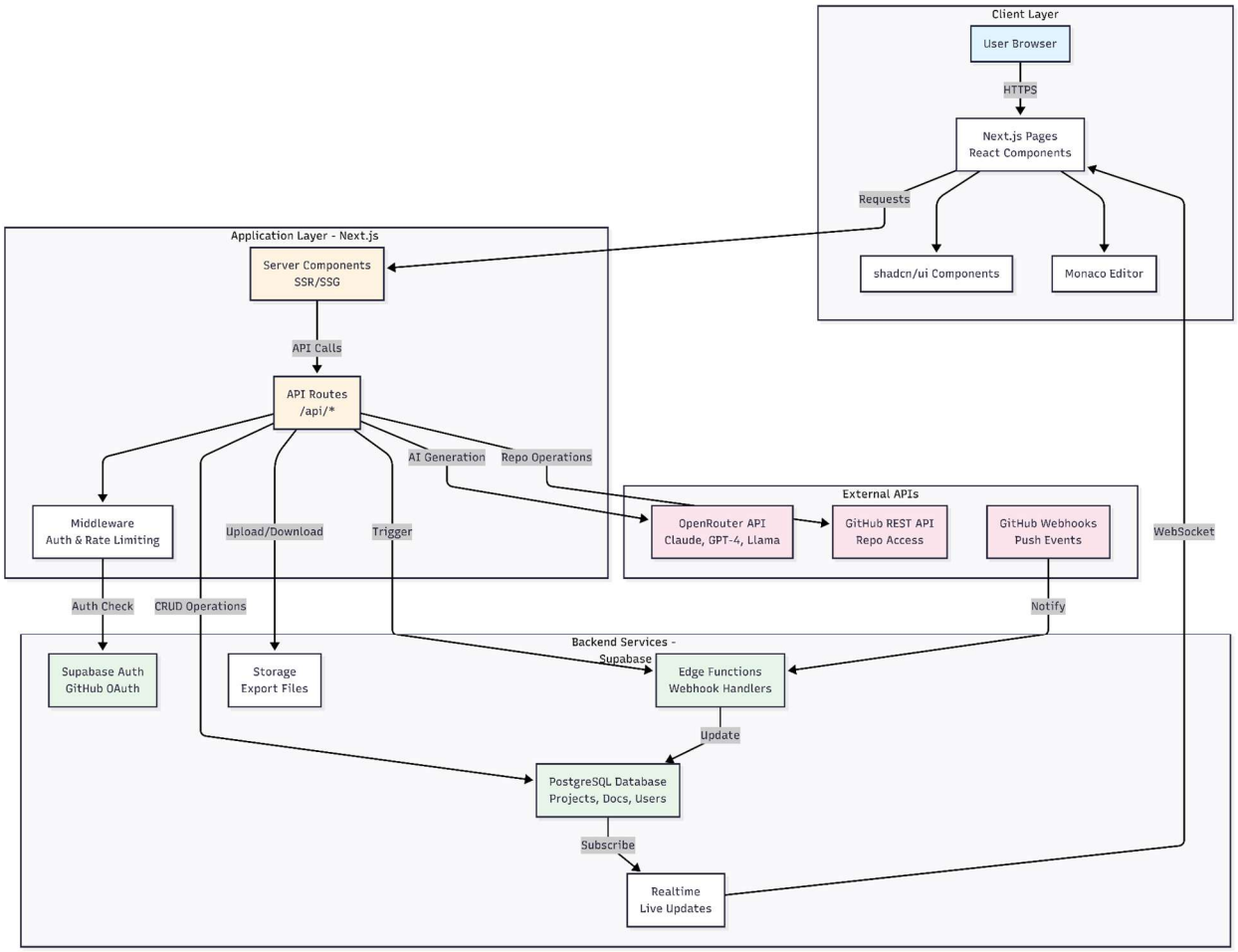
CHAPTER FOUR

4.0 System Analysis & Design

4.1 Data Flow Diagram (DFD)



4.2 Entity Relationship (ER) Diagram



4.3 Proposed Timelines

Week	Phase	Key Deliverables
1-2	Foundation	Project setup, Auth flow, Database schema
3-4	AI Planning	OpenRouter integration, PRD/User Story engine
5-6	GitHub Core	OAuth integration, Repository file fetching
7-8	Doc Generation	Code analyzer, README/ARCHITECTURE generators
9-10	Sync & Merge	Webhook implementation, Conflict resolution

11-12	Polish/Deploy	Wiki UI, Export features, Final Vercel deployment

Critical Path: The integration of GitHub Webhooks (Week 10) is the most critical dependency, requiring a fully functional Repository Connection (Week 6).

REFERENCES

Aghajani, E., Nagy, C., Vega-Márquez, O. L., Linares-Vásquez, M., Moreno, L., Bavota, G., & Lanza, M. (2020). Software documentation issues: Unveiling the problems and challenges that practitioners face. *IEEE Transactions on Software Engineering*, 46(11), 1215-1234.

Dagenais, B., & Robillard, P. N. (2020). Creating and using developer profiles to support software maintenance. *Journal of Systems and Software*, 161, 110471.

Forward, A., & Lethbridge, T. C. (2022). The relevance of software documentation in agile projects: A survey. *Proceedings of the 2022 ACM/IEEE International Conference on Software Engineering*.

Github. (2023). *GitHub Copilot and the future of software development documentation*. GitHub Engineering Blog. https://github.blog/engineering/

Ochieng, P., & Mutua, S. (2024). Knowledge management practices in Kenyan tech hubs: Challenges and opportunities for sustainable software engineering. *African Journal of Information Systems*, 16(1), 45-62.

Smith, T., & Johnson, L. (2023). Documentation-as-Code: Evaluating the efficiency of automated documentation tools in cloud-native environments. *International Journal of Computer Science & Information Technology*, 15(4), 88-102.

Treude, C., & Robillard, P. N. (2021). Augmenting software documentation with technical discussions. *IEEE Software*, 38(3), 67-73.