1. INTRODUCTION

1.1 About the Project

Study in Woods is an AI-powered educational platform that helps students organize academic materials and interact with course content intelligently. The system processes syllabus PDFs using NLP to extract structured information and provides an AI chat interface for study guidance.

1.2 Purpose of the Project

The primary objectives are:

- **Academic Organization:** Centralized platform for managing universities, courses, semesters, and subjects
- **Intelligent Content Extraction:** AI-powered processing of syllabus PDFs with automatic data extraction
- **AI-Assisted Learning:** Conversational AI for answering questions and providing study guidance
- **Exam Preparation:** Past year question paper extraction and categorization linked to syllabus topics
- **Progress Tracking:** Study progress monitoring and personalized study plan management

1.3 System Architecture

The application follows a three-tier microservices architecture:

- **Frontend:** Next.js 15 with React 19 providing responsive UI
- **Backend:** Go (Golang) with Fiber framework handling API and business logic
- **Data Layer:** PostgreSQL for primary storage, Redis for caching and sessions

- **OCR Service:** Python FastAPI microservice for text extraction from scanned documents

1.4 Core Functionalities

1. **User Management:** Secure authentication with role-based access (Admin/Student)
2. **Document Processing:** Syllabus upload with AI-powered content extraction
3. **AI Chat Interface:** Subject-specific conversational assistance
4. **PYQ Management:** Question paper extraction and categorization
5. **Analytics Dashboard:** Usage statistics and performance tracking
6. **RESTful API:** 100+ endpoints for programmatic access

1.5 User Characteristics

Students: MCA or similar program students seeking AI-assisted learning and exam preparation tools.

Administrators: Academic staff managing universities, courses, and system settings with audit capabilities.

1.6 Operating Environment

- **Client:** Modern web browsers (Chrome, Firefox, Safari, Edge) on all devices
- **Server:** Docker containerized deployment on Linux with HTTPS
- **Cloud:** DigitalOcean Spaces for storage, GradientAI (Llama 3.3 70B) for AI features

1.7 Constraints and Dependencies

Constraints: Go/Next.js stack required; PDF-only uploads; must support 1000+ concurrent users.

Dependencies: DigitalOcean services (AI, storage), PostgreSQL, Redis, third-party libraries.