Software Requirements Document for [SD-14 - Personalized Learning]

TEAM: SD-14

AUTHORS:

Adam Hisel

Carter Parks

Morgan Prieskorn

Zac Mueterthies

Ryan Johnson

1 Introduction

1.1 PURPOSE

The purpose of this document is to define the functional and non-functional requirements of the Learn-X Personalized Learning Platform. It outlines how the system should interact with its users and describes the capabilities necessary for successful deployment and user satisfaction.

1.2 SCOPE

The Learn-X platform is a web-based learning system for students and instructors. Instructors can create courses, upload learning materials (PDFs/audio), and manage students. Students can view materials, access personalized versions of files, and interact with a course-specific AI chatbot. The system supports onboarding surveys and adaptive content delivery based on each student's learning profile.

1.3 DEFINITIONS, ACRONYMNS, ABBREVIATIONS

Term	Description
Personalized File	A student-specific version of course material based on onboarding data
Onboarding Quiz	A survey to gather student preferences and learning background
RAG	Retrieval-Augmented Generation – a technique combining search and AI
FAISS	Facebook AI Similarity Search – used for vector-based document indexing
Next.js	React-based web framework used for frontend

2 Overall Description

2.1 PRODUCT PERSPECTIVE

The Learn-X system is a personalized learning platform built with modern web and AI technologies. It includes role-based access (student/instructor), course and file management, and AI-powered content delivery. It integrates React (Next.js), Flask, Firebase, PostgreSQL, and OpenAI APIs. The system replaces traditional static learning with dynamic, tailored content.

2.2 PRODUCT FUNCTIONS

- **User Management**: Registration/login, role assignment, access control
- **Instructor Features**: Course creation, file upload, student management
- **Student Features**: Course joining, material viewing, onboarding quiz, profile management
- **Personalization**: AI-generated file transformations using student data
- **Chatbot**: Contextual Q&A, content-aware limitations, history tracking
- **Preview/Navigation**: In-browser PDF preview, navigation to personalized files

2.3 USER CHARACTERISTICS

- **Instructors**: Educators managing courses and uploading materials; typically use the platform weekly
- **Students**: College-level learners accessing materials and using personalization/chat; frequent users (2-5x/week)

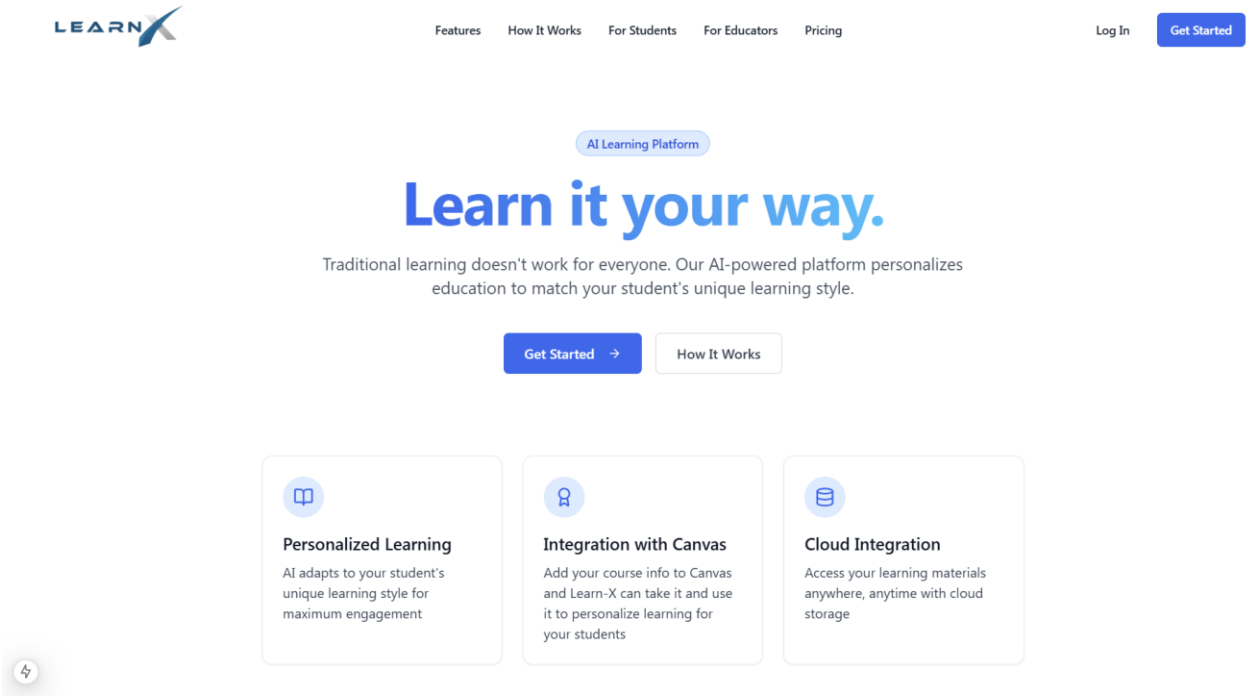
2.4 CONSTRAINTS

- Web-based only; must run in modern browsers
- Files must be under 100MB
- AI content is constrained to course uploads only
- Personalization accuracy is dependent on OpenAI and embedding algorithms

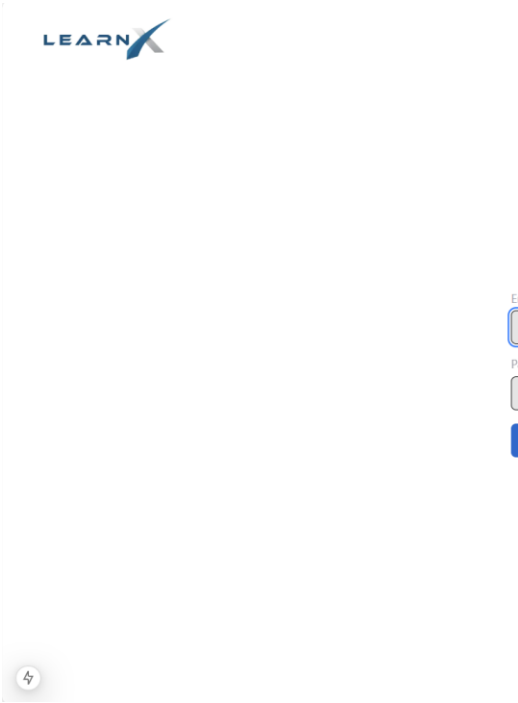
2.5 ASSUMPTIONS AND DEPENDENCIES

- Users have stable internet access
- Authentication depends on Firebase Auth
- Database hosted via PostgreSQL (Neon)
- AI features require OpenAI API and vector similarity search (FAISS + pgvector)

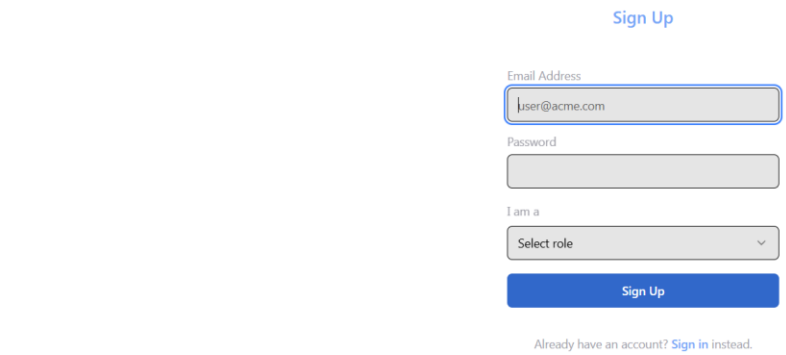
2.6 MAJOR USER INTERFACES



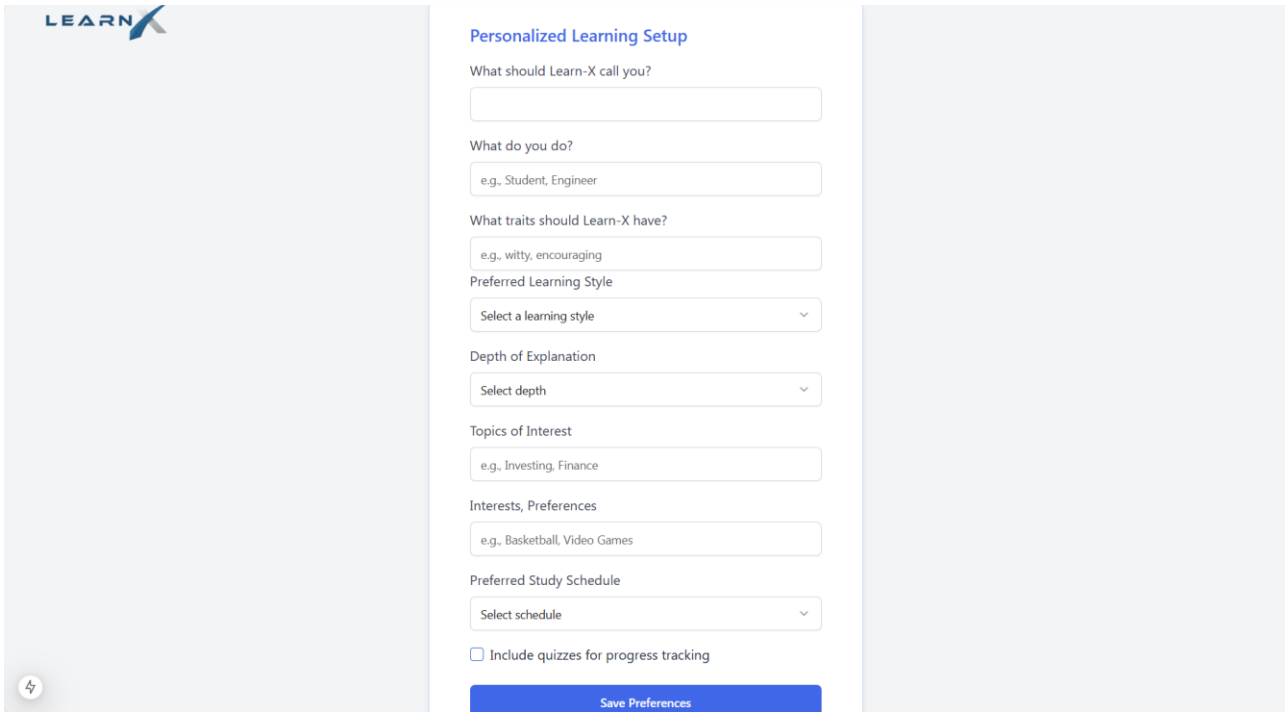
Screen 1: Landing Page



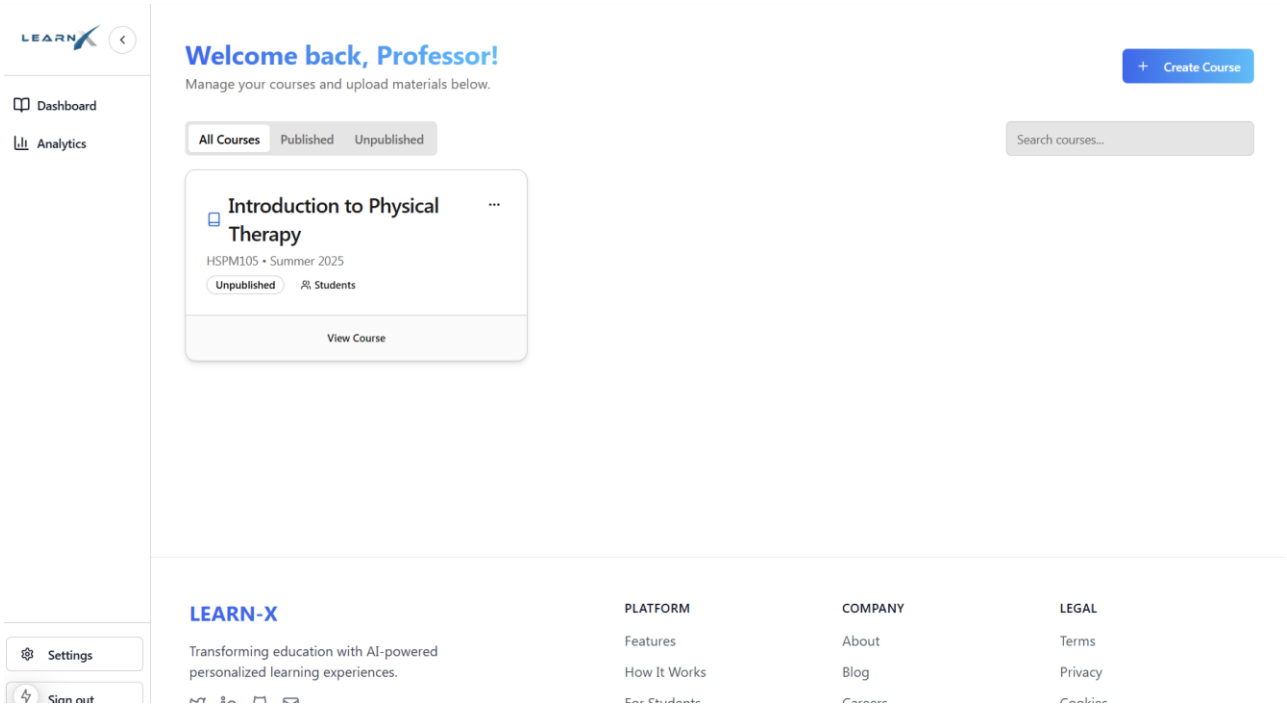
Screen 2: Sign In Page



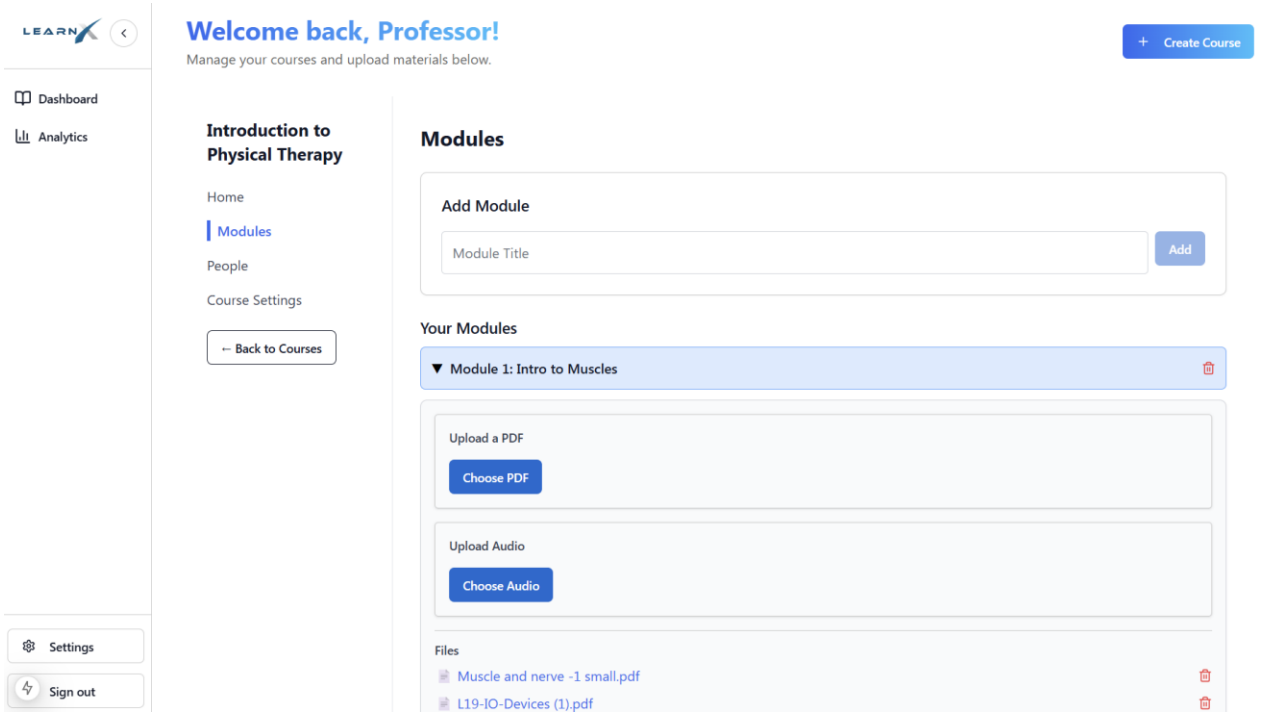
Screen 3a: Sign Up Page



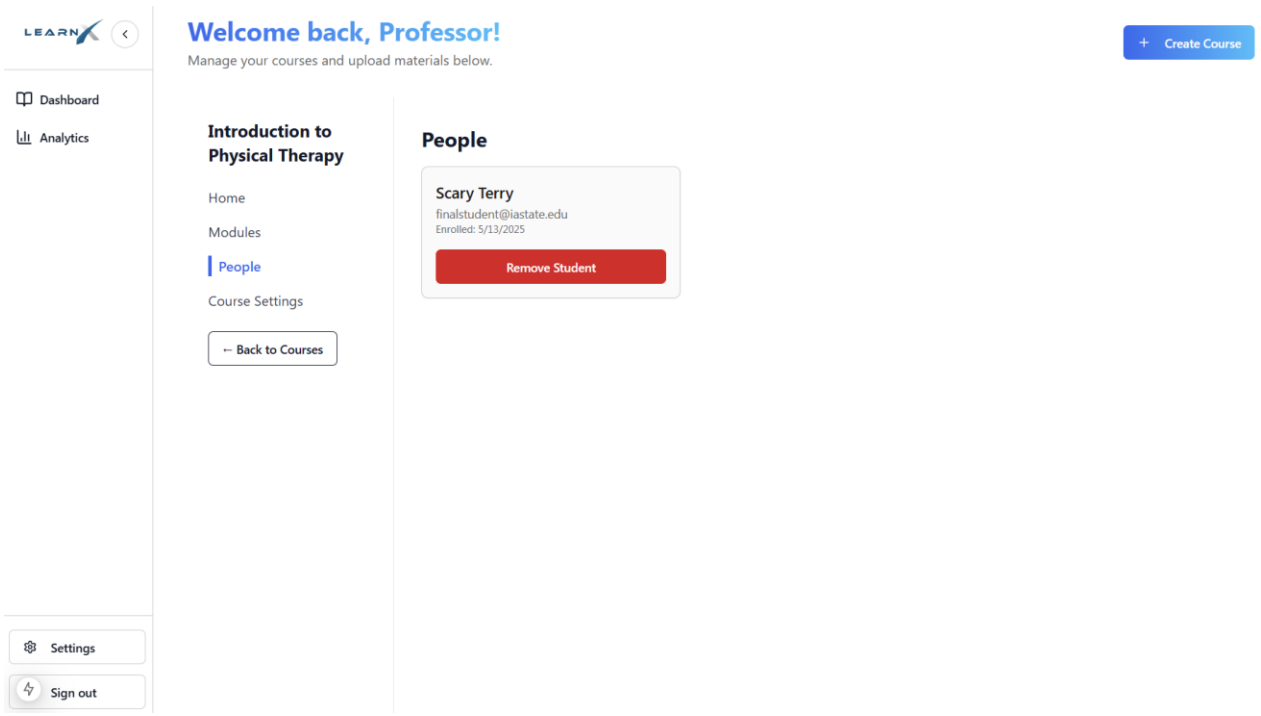
Screen 3b: Onboarding Page (after sign up)



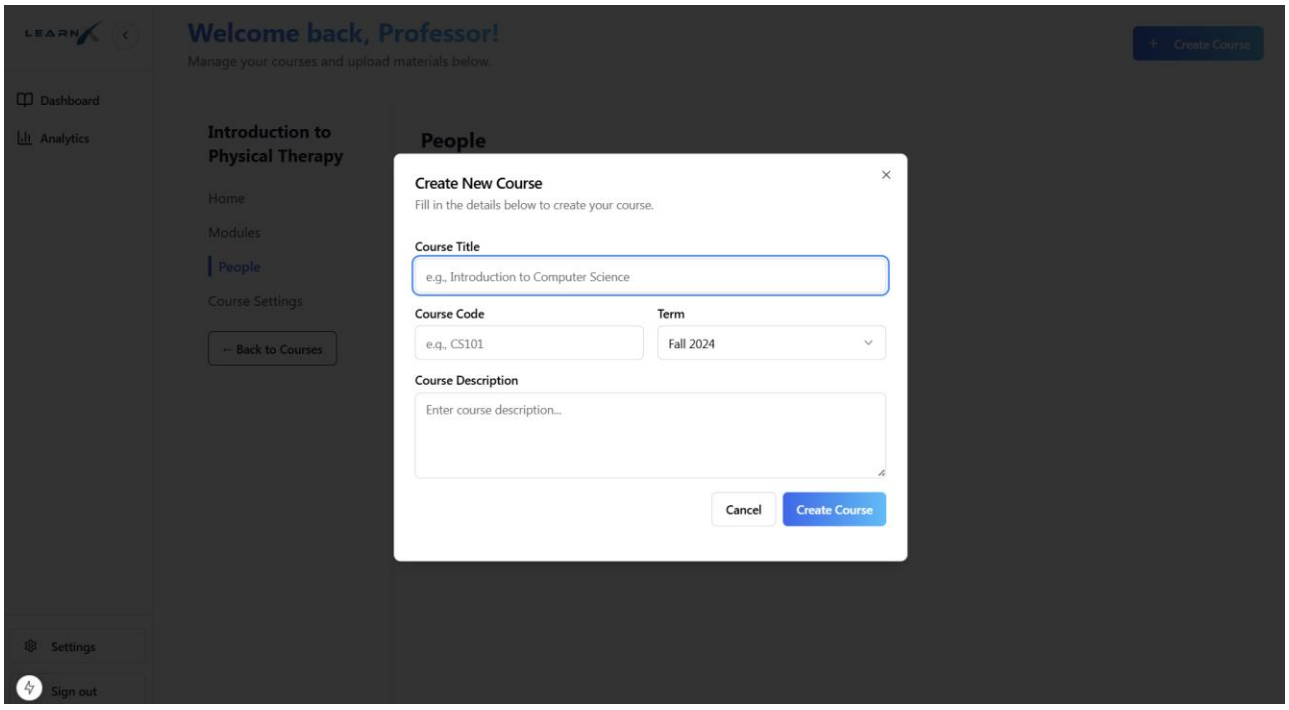
Screen 4: Professor Dashboard



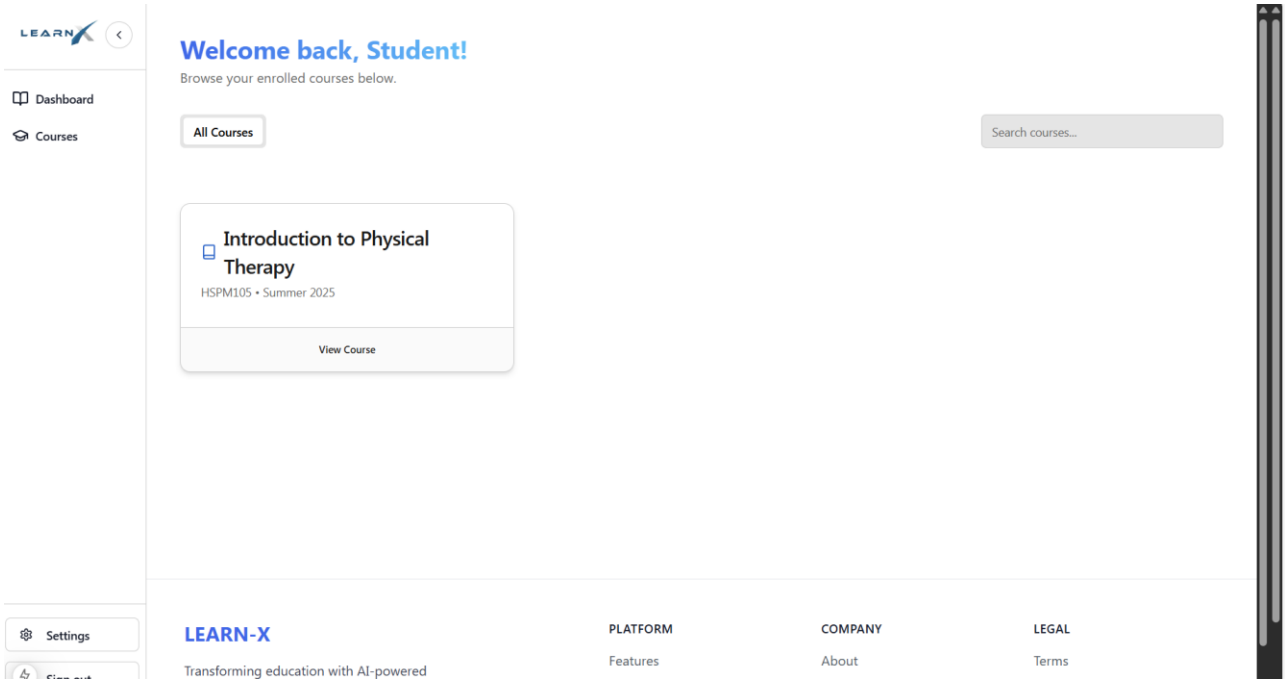
Screen 5: Professor Modules Tab



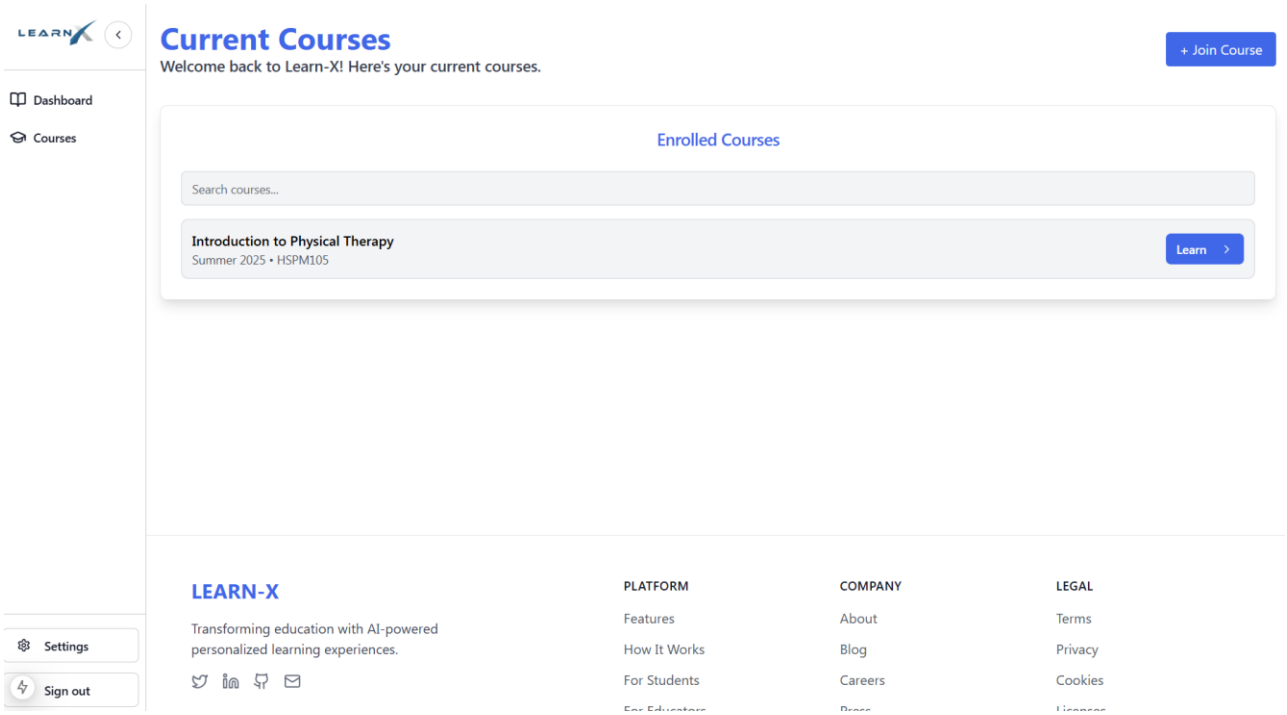
Screen 6: Professor People Tab



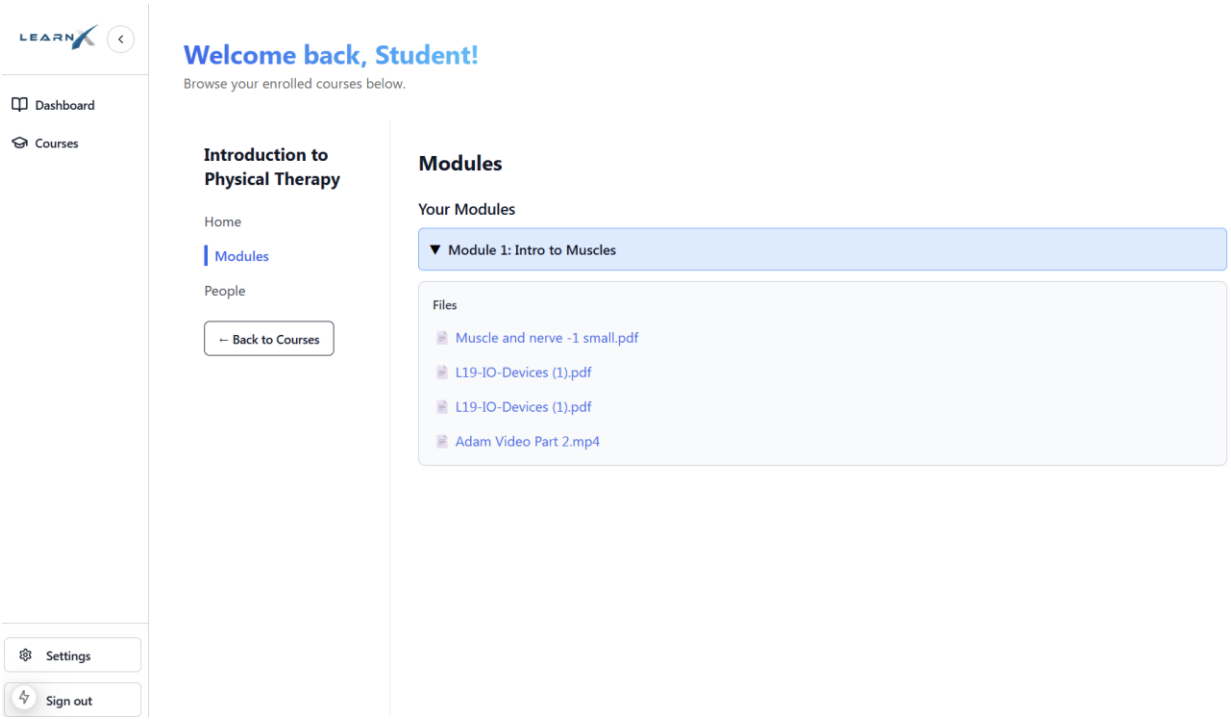
Screen 7: Professor Create Course



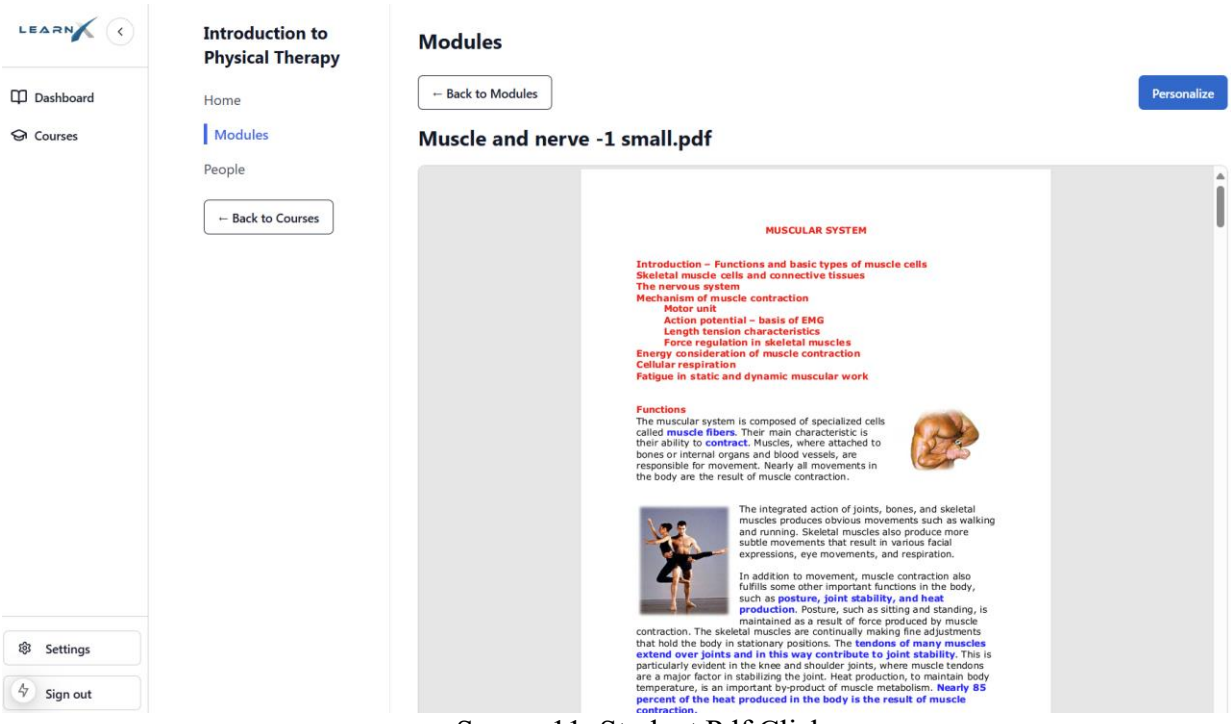
Screen 8: Student Dashboard



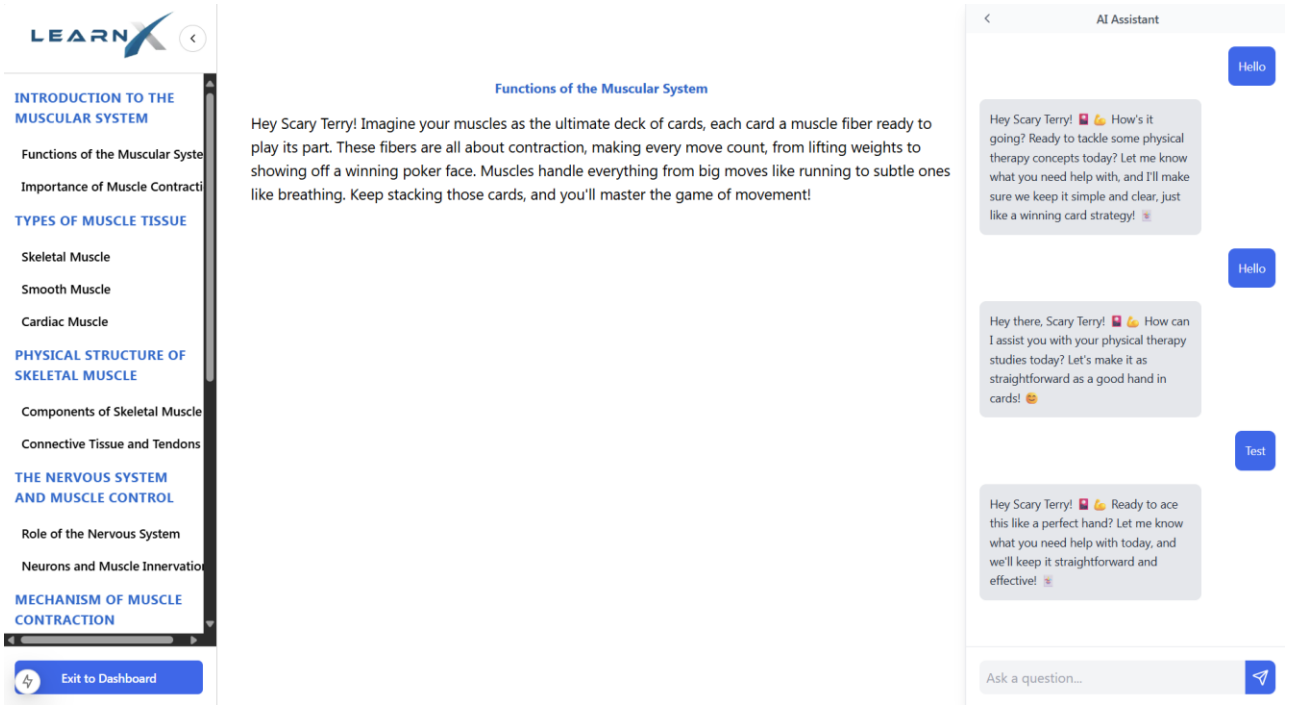
Screen 9: Student Courses Page



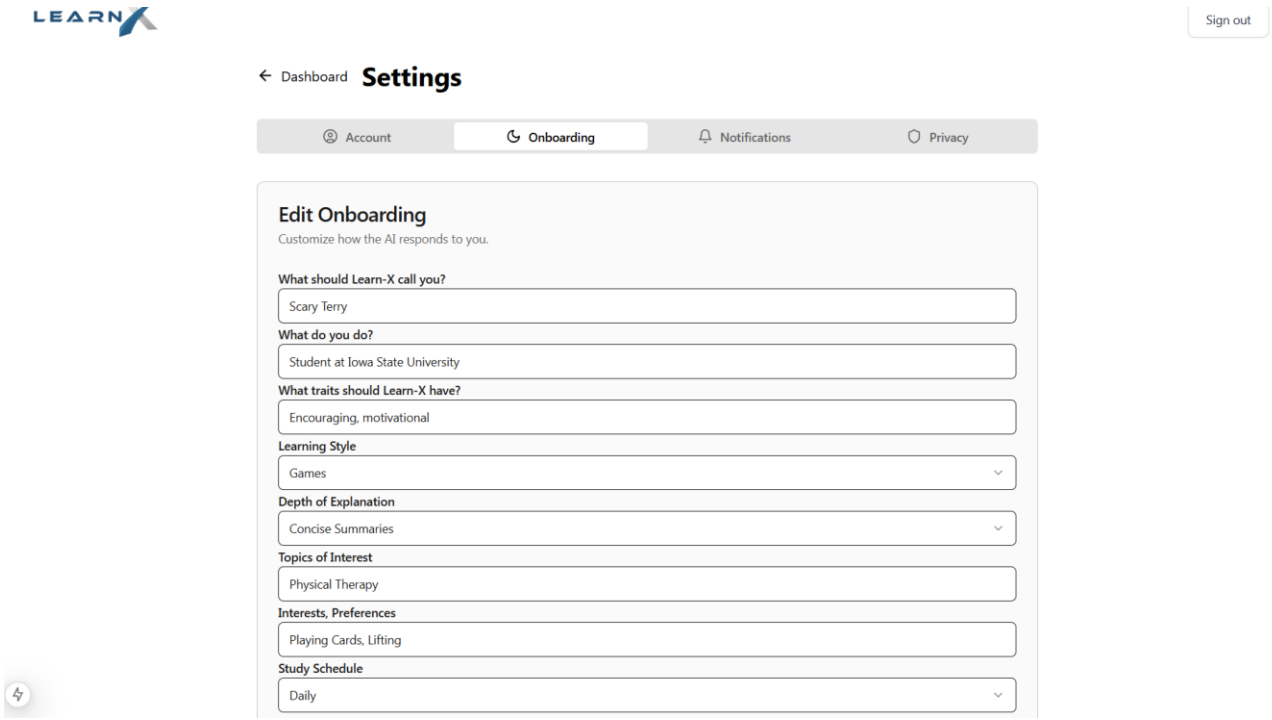
Screen 10: Student Modules Tab



Screen 11: Student Pdf Click

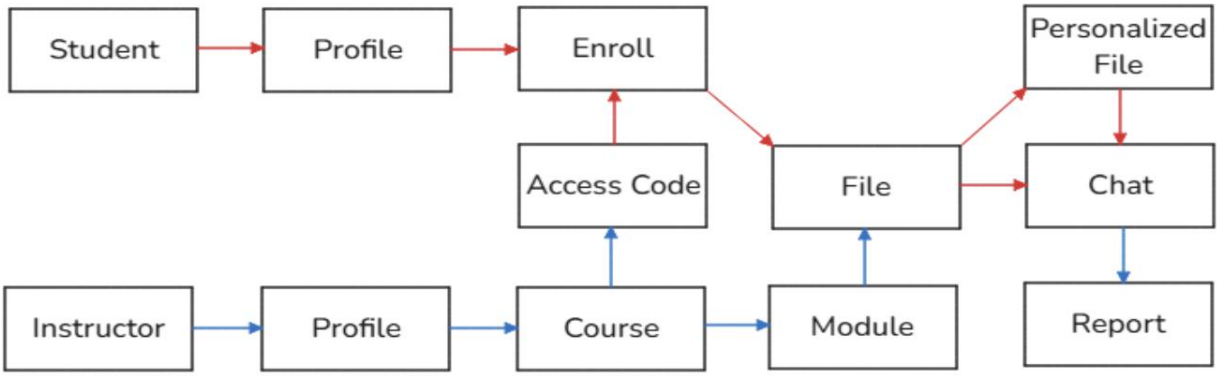


Screen 12: Student Learning Page



Screen 13: Settings Page

2.7 USER FLOW



2.8 USER STORIES

Student User Stories

- **As a student**, I want to register and log in so that I can access my personalized dashboard.
- **As a student**, I want to join a course using an access code so that I can see course materials.
- **As a student**, I want to complete an onboarding quiz so that the system can tailor content to my learning preferences.
- **As a student**, I want to view PDFs and audio files uploaded by my instructor so that I can study effectively.
- **As a student**, I want to receive a personalized version of the course files so that the content aligns with my interests.
- **As a student**, I want to interact with an AI chatbot so that I can get help with course content.
- **As a student**, I want to see previous chatbot conversations for each module so that I can review past questions.
- **As a student**, I want to update my onboarding responses so that my learning preferences can be revised as needed.

Instructor User Stories

- **As an instructor**, I want to create courses and modules so that I can organize my curriculum.
- **As an instructor**, I want to upload PDFs and audio files to a course so that students can access learning materials.
- **As an instructor**, I want to manage enrolled students so that I can track participation.
- **As an instructor**, I want to preview how content will be personalized so that I understand what students will see.

- **As an instructor**, I want to delete outdated files or modules so that my course stays current.

System User Stories

- **As the system**, I want to personalize uploaded content using student onboarding data so that each student gets a tailored learning experience.
- **As the system**, I want to restrict access to instructor-only features so that students cannot modify courses.
- **As the system**, I want to store chat history per student/module so that each conversation is persistent and relevant.

Admin User Stories

- **As an admin**, I want to have full access to all instructor and student functionality so that I can manage and test the system from any user perspective.

3 Specific Requirements

3.1 FEATURES

FEATURE-1: Course Management

- Description: Instructors can create, edit, and delete courses and modules.
- Functional Detail: Courses include title, description, and modules. Each module can hold multiple files.
- Non-Functional Requirement: Interface should load within 2 seconds when navigating between courses. Course data should persist across sessions.

FEATURE-2: File Upload & Preview

- Description: Instructors upload PDFs or audio files; students can preview them in-browser.
- Functional Detail: Files are displayed with embedded viewers and auto-labeled by filename/type.
- Non-Functional Requirement: Files must be under 100MB. System must prevent unsupported file types.

FEATURE-3: AI Personalization

- Description: Uploaded files are converted into personalized content per student.
- Functional Detail: On upload, files are chunked, embedded (FAISS), and personalized based on onboarding data using OpenAI and RAG.
- Non-Functional Requirement: Personalized content generation must complete within 10 seconds.

FEATURE-4: Student Onboarding

- Description: New students fill out a quiz to guide personalization.
- Functional Detail: Stores preferences like learning style, interests, prior experience.
- Non-Functional Requirement: Data must be editable and securely stored. Usability goal: form completion < 2 minutes.

FEATURE-5: Course Chatbot

- Description: AI chatbot answers questions based on course material.
- Functional Detail: Restricts answers to uploaded files using vector search relevance.
- Non-Functional Requirement: Responses returned in < 5 seconds; must reject off-topic questions clearly.

3.2 PERFORMANCE REQUIREMENTS

- Dashboard and file preview pages must load in under 2 seconds.
- AI-generated responses must return results in under 10 seconds
- File personalization must return results in under 30 seconds.

3.3 DESIGN CONSTRAINTS

- Must use: Next.js (frontend), Flask (backend), PostgreSQL (database), Firebase (auth), OpenAI (AI engine), FAISS + pgvector (embedding/search).
- Deployment must be Docker-compatible.
- Frontend styled using Tailwind CSS and responsive on common screen sizes.

3.4 SOFTWARE SYSTEM ATTRIBUTES

- Reliability: Uptime target 99.5%. Auto-reconnect to DB on failure. Firebase handles auth session expiration.
- Security: HTTPS-only cookies, Firebase-auth verified sessions, role-based endpoint protections, and ownership validation.
- Maintainability: Frontend/backend separated; modular file and API structure; environment variables stored in .env.
- Portability: Fully Dockerized. Compatible with cloud platforms like Vercel, Heroku, or AWS.