EduHub

All-in-One Course Learning Platform

Implementation Plan v2.0

Vercel-Ready | GitHub-Ready | Production-Ready

A modular, scalable learning management system

optimized for engineering education

2025

Table of Contents

1. Executive Summary

EduHub is a comprehensive, modular learning platform designed specifically for electrical engineering students. Built with Next.js 15 and optimized for Vercel deployment, the platform serves as a centralized hub where students can browse, select, and complete multiple courses at their own pace. The architecture prioritizes modularity, enabling educators to add new courses with minimal configuration changes.

Platform Highlights:

- Modular Course System — Add new courses by creating a folder and registering one line of code
- Vercel-Optimized — Zero-config deployment with static generation and client-side state management
- GitHub-Ready — Clean repository structure suitable for open-source collaboration
- Rich Learning Experience — LaTeX mathematics, interactive quizzes, flashcards, and worked examples
- Progress Tracking — Cross-course statistics, daily streaks, and completion certificates
- Enhanced QoL — Keyboard shortcuts, global search, dark mode, and responsive design

Target Users:

Primary users are electrical engineering undergraduate students studying control systems, signal processing, and related technical subjects. The platform supports self-paced learning outside traditional classroom settings. Secondary users include educators who can contribute new courses and students from other engineering disciplines.

2. Platform Overview

2.1 Core Philosophy

• Modular First: Every course is self-contained. Adding content never requires touching core application code.

• Student-Centric: Every feature is designed to reduce friction and maximize learning time.

• Performance Obsessed: Fast loading, smooth interactions, and optimized bundle sizes.

• Accessible Everywhere: Works on desktop, tablet, and mobile with consistent experience.

• Open by Default: GitHub-ready structure with clear documentation for contributors.

2.2 Platform Architecture

EduHub uses a single-page application architecture where course content is statically generated at build time, and user progress is managed entirely client-side via LocalStorage. This architecture eliminates the need for a backend server while still providing rich interactive features.

• Zero server costs — 100% static hosting on Vercel free tier

• Instant page loads — content is pre-rendered at build time

• Offline-capable — Service Worker caches all content for offline study

• Privacy-first — No user data leaves the browser

• Simple deployment — Push to GitHub, Vercel auto-deploys

2.3 User Journey

Stage	Screen	User Actions
Entry	Hub Landing	View overall progress, browse courses, continue learning
Discovery	Course Catalog	Filter by category, preview course details, start course
Learning	Course Viewer	Navigate modules, read lessons, mark complete
Practice	Study Tools	Review flashcards, quick reference, take notes
Assessment	Quiz Mode	Answer questions, view explanations, track scores

Achievement	Celebration	View streak, earn badges, download certificate

3. Vercel Compatibility

EduHub is architected specifically for Vercel deployment. Every design decision considers Vercel's serverless, static-first hosting model. The result is a platform that deploys instantly, scales automatically, and costs nothing on Vercel's free tier.

3.1 Compatibility Matrix

Feature	Status	Implementation
Next.js 15 App Router	□ Native	First-class Vercel support
Static Generation	□ Native	All pages pre-rendered at build
Client-side Routing	□ Native	Fast SPA navigation
LocalStorage	□ Native	Browser-native persistence
Service Worker	□ Native	Offline support via next-pwa
Image Optimization	□ Native	Vercel Image Optimization API
Edge Functions	□ Optional	For future API needs

3.2 What We Avoid

Certain patterns are incompatible with Vercel's architecture. We deliberately avoid these to ensure seamless deployment:

- Server-side Databases (SQLite, PostgreSQL) — Vercel functions are stateless. Instead, we use LocalStorage for client-side persistence.
- File System Writes — Vercel's filesystem is read-only. All user data is stored in the browser.
- Long-running Processes — Vercel functions timeout after 10-60 seconds. All processing is client-side.
- WebSockets — Not needed for this use case. Real-time features can use polling if added later.

3.3 Future Scalability Options

If multi-device sync or user accounts become necessary in the future, these Vercel-compatible options are available:

Option	Effort	Use Case

Vercel KV (Redis)	Low	Session storage, rate limiting
Vercel Postgres	Medium	User accounts, cross-device sync
Vercel Blob	Low	User-uploaded files, exports
Supabase	Medium	Full auth + database solution

4. User Experience Design

4.1 Hub Landing Page

The hub is the command center for all learning activities. It provides immediate value by showing progress, enabling quick access to courses, and encouraging continued learning through gamification elements.

- Hero Section: Welcome message with user's name (if set), primary CTA buttons for 'Continue Learning' and 'Browse Courses', and a motivational tagline.
- Quick Stats Bar: Three-column display showing total lessons completed, courses in progress, and current daily streak with fire animation.
- Continue Learning Widget: Prominent card showing the last accessed course with one-click resume. Shows progress percentage and next lesson title.
- Course Catalog: Category filter tabs (All, Control Systems, Signals, etc.) above a responsive grid. Each card displays course icon, title, difficulty, lesson count, and progress bar.
- Achievements Preview: Mini showcase of recently earned badges and available achievements to unlock.
- Footer: Keyboard shortcuts reference, theme toggle, and links to help/documentation.

4.2 Course Viewer Layout

The course viewer uses a proven two-column layout optimized for focused learning. The sidebar provides navigation while the main content area delivers the learning material.

- Fixed Header: Back-to-hub button, course title, overall progress bar, search icon, and theme toggle. Thin progress indicator shows lesson completion percentage.
- Collapsible Sidebar: Module list with expand/collapse for lessons. Completion checkboxes, active lesson highlight, quiz buttons, and bookmarks section. Auto-collapses on mobile.
- Main Content Area: Lesson title, reading time estimate, formatted content with LaTeX math, collapsible example boxes, key points summary, and navigation buttons.
- Floating Action Bar: Fixed-position bar with Quick Reference (Q), Flashcards (F), Notes (N), and Jump to Top (^). Appears on scroll for easy access.
- Progress Indicator: Subtle progress bar at top of content area showing position within module.

4.3 Mobile Experience

Mobile users receive a tailored experience that maximizes screen real estate while maintaining all functionality:

- Hamburger menu replaces sidebar — full-screen navigation when opened
- Touch-friendly button sizes (minimum 44px tap targets)
- Swipe gestures for lesson navigation (left/right)
- Bottom-anchored action bar for thumb accessibility
- Pull-to-refresh for progress sync (future: cloud sync)
- Reduced animations for battery optimization
- Portrait-optimized content layout with single-column design

4.4 Responsive Breakpoints

Name	Width	Layout Adaptations
Mobile S	< 480px	Compact header, bottom nav, full-width cards
Mobile L	480-767px	Side drawer nav, optimized touch targets
Tablet	768-1023px	Sidebar auto-hides, two-column grid
Desktop	1024-1439px	Full sidebar visible, three-column grid
Desktop L	≥ 1440px	Max-width container, optimal reading width

5. Architecture Design

5.1 Directory Structure

The project follows Next.js 15 App Router conventions with clear separation of concerns:

```
edu-hub/
├── src/
│   ├── app/
│   │   ├── page.tsx              # Hub landing page
│   │   ├── layout.tsx            # Root layout with providers
│   │   ├── globals.css           # Global styles + Tailwind
│   │   ├── courses/
│   │   │   └── [slug]/
│   │   │       └── page.tsx      # Dynamic course viewer
│   │   └── api/                  # Optional Edge Functions
│   │
│   ├── components/
│   │   ├── ui/                   # shadcn/ui primitives
│   │   ├── hub/                  # Hub-specific components
│   │   │   ├── CourseCard.tsx
│   │   │   ├── CategoryFilter.tsx
│   │   │   ├── ProgressDashboard.tsx
│   │   │   ├── StreakCounter.tsx
│   │   │   ├── ContinueLearning.tsx
│   │   │   └── AchievementBadge.tsx
│   │   ├── course/               # Course viewer components
│   │   │   ├── CourseViewer.tsx
│   │   │   ├── SidebarNav.tsx
│   │   │   ├── LessonContent.tsx
│   │   │   ├── ExampleBox.tsx
│   │   │   ├── Quiz.tsx
│   │   │   ├── Flashcard.tsx
│   │   │   ├── QuickReference.tsx
│   │   │   ├── Notes.tsx
│   │   │   ├── ProgressBar.tsx
│   │   │   ├── Celebration.tsx
│   │   │   └── Certificate.tsx
│   │   └── shared/               # Shared components
│   │       ├── Header.tsx
│   │       ├── SearchOverlay.tsx
│   │       ├── ThemeToggle.tsx
│   │       └── KeyboardShortcuts.tsx
│   │
│   ├── lib/
│   │   ├── courses/              # Course registry
│   │   │   ├── index.ts          # Central registry
│   │   │   ├── types.ts          # Type definitions
```

```
│   │   │   └── utils.ts           # Helper functions
│   │   │
│   │   ├── content/               # Course content
│   │   │   ├── control-systems/
│   │   │   │   ├── index.ts
│   │   │   │   ├── modules.ts
│   │   │   │   ├── examples.ts
│   │   │   │   ├── exercises.ts
│   │   │   │   ├── quizzes.ts
│   │   │   │   └── reference.ts
│   │   │   └── transformation-methods/
│   │   │
│   │   ├── storage/               # Storage utilities
│   │   │   ├── index.ts           # LocalStorage wrapper
│   │   │   ├── progress.ts        # Progress management
│   │   │   └── settings.ts        # User settings
│   │   │
│   │   └── utils.ts               # General utilities
│   │
│   ├── hooks/
│   │   ├── useProgress.ts         # Progress state
│   │   ├── useStreak.ts           # Streak calculation
│   │   ├── useCourse.ts           # Course data
│   │   ├── useSearch.ts           # Search functionality
│   │   └── useKeyboardShortcuts.ts  # Shortcut handling
│   │
│   ├── contexts/
│   │   ├── AppContext.tsx         # Global state
│   │   └── ThemeContext.tsx       # Theme management
│   │
│   └── types/
│       └── index.ts               # Global types
│
├── public/
│   ├── icons/                     # Course icons
│   ├── images/                    # Static images
│   └── manifest.json              # PWA manifest
│
├── tailwind.config.ts
├── next.config.ts
├── package.json
└── README.md
```

5.2 Course Registry System

The course registry enables true modularity. Courses are self-contained and registered in a single file, allowing addition without touching application code:

```
// lib/courses/index.ts
import { Course } from './types';
import { controlSystems } from '@/lib/content/control-systems';
import { transformationMethods } from '@/lib/content/transformation-methods';

// Central registry - add new courses here
export const courses: Course[] = [
  controlSystems,
  transformationMethods,
  // Add more courses below...
];

// Helper functions
export function getCourseBySlug(slug: string): Course | undefined {
  return courses.find(c => c.slug === slug);
}

export function getCoursesByCategory(category: CourseCategory): Course[] {
  return courses.filter(c => c.category === category);
}

export function getAllCategories(): CourseCategory[] {
  return [...new Set(courses.map(c => c.category))];
}
```

5.3 Component Hierarchy

Layer	Purpose	Examples
UI Primitives	Base components from shadcn/ui	Button, Card, Dialog, Input, Tabs
Shared Components	Cross-feature reusable components	Header, SearchOverlay, ThemeToggle
Hub Components	Hub landing page elements	CourseCard, ProgressDashboard, StreakCounter
Course Components	Course viewer elements	LessonContent, Quiz, Flashcard, SidebarNav
Page Components	Full page layouts	HubPage, CoursePage

6. Data Models

6.1 Course Model

Field	Type	Description
id	string	Unique identifier (e.g., "control-systems")
slug	string	URL-friendly identifier
title	string	Display name
subtitle	string	Tagline shown on card
description	string	Full description for course page
category	CourseCategory	Category enum for filtering
icon	string	Lucide icon name
color	string	Theme color (hex)
difficulty	enum	beginner | intermediate | advanced
estimatedHours	number	Estimated completion time
prerequisites	string[]	Required prior knowledge
learningOutcomes	string[]	What students will learn
modules	Module[]	Array of course modules

6.2 Module & Lesson Models

Field	Type	Description
id	string	Unique lesson identifier
title	string	Lesson title
content	string	Markdown content with LaTeX
objectives	string[]	Learning objectives
keyPoints	string[]	Key takeaways

readingTime	number	Estimated reading time (minutes)
examples	Example[]	Worked examples with solutions
exercises	Exercise[]	Practice problems
quizQuestions	QuizQuestion[]	Optional quiz (per module)

6.3 User Progress Model (LocalStorage)

Field	Type	Description
totalStudyTime	number	Cumulative seconds across all courses
currentStreak	number	Consecutive active days
longestStreak	number	Best streak achieved
lastActiveDate	string	ISO date of last activity
courses	Record	Per-course progress data
achievements	string[]	Unlocked achievement IDs
settings	UserSettings	Theme, preferences, etc.

7. Course Categories

Categories help students discover relevant content and provide visual organization in the hub interface. Each category has a distinct color that is used consistently throughout the platform.

Category ID	Display Name	Color	Topics
control-systems	Control Systems	#3B82F6	Feedback control, stability, PID, root locus
signals-systems	Signals & Systems	#8B5CF6	Signal processing, transforms, filtering
electronics	Electronics	#10B981	Circuit analysis, analog, digital electronics
power-systems	Power Systems	#F59E0B	Power generation, distribution, machines
mathematics	Mathematics	#EC4899	Engineering math, transforms, linear algebra
programming	Programming	#06B6D4	DSP, embedded systems, MATLAB, Python
communications	Communications	#F97316	Wireless, modulation, information theory

Categories are defined as a TypeScript enum for type safety and can be extended by adding new values to the enum and the category information registry. The hub filter component automatically reflects available categories.

8. Quality of Life Features

Quality of Life features distinguish exceptional learning platforms. These features reduce friction, improve accessibility, and enhance the overall learning experience. Features are prioritized by impact on student productivity.

8.1 Critical QoL (Must Have)

Feature	Benefit	Implementation
Global Search	Find any content instantly across all courses	Fuse.js for fuzzy search, indexed at build time
Keyboard Shortcuts	Navigate 3x faster, power-user efficiency	Global event listener with shortcut registry
Auto-Save Progress	Never lose work, seamless experience	Debounced LocalStorage writes
Continue Learning	One-click resume, zero friction restart	Track last visited course/lesson
Reading Time	Plan study sessions, set expectations	Word count / 200 WPM calculation
Dark Mode	Reduce eye strain, study at night	Tailwind dark: variant + LocalStorage persistence
Mobile Responsive	Study anywhere on any device	Tailwind responsive utilities
Fast Page Loads	Instant navigation, better UX	Static generation + client-side routing

8.2 High-Impact QoL (Should Have)

Feature	Benefit	Implementation
Daily Goals	Build consistent study habits	Goal setting widget with daily tracking
Formula Quick Reference	Instant access to key formulas	Per-course reference sheets, keyboard shortcut
Personal Notes	Active reading improves retention	Per-lesson notes with Markdown support

Lesson Bookmarks	Quick access to important lessons	Star icon, dedicated bookmarks view
Progress Export	Backup progress, share achievements	JSON export/import functionality
Print-Friendly View	Paper revision for exams	CSS @media print styles
Achievement System	Motivation through gamification	Unlockable badges for milestones
Study Statistics	Insight into learning patterns	Charts showing time, completion rates

8.3 Enhanced QoL (Nice to Have)

Feature	Benefit	Implementation
Spaced Repetition	Science-backed memory retention	SM-2 algorithm for flashcard scheduling
Error Reporting	Crowdsource content corrections	GitHub issue template link
Lesson Highlighting	Mark important passages	Text selection + highlight storage
Unit Converter	Quick engineering calculations	Floating converter widget
Audio Summary	Review on the go (podcast style)	Text-to-speech integration
Offline Mode (PWA)	Study without internet	Service Worker caching
Custom Themes	Personalization, accessibility	Theme builder with presets
Focus Mode	Distraction-free studying	Hide sidebar, full-width content

8.4 QoL Design Principles

- One-Click Actions: Common tasks (continue learning, mark complete) require single clicks
- Instant Feedback: Every action has immediate visual confirmation
- Progressive Disclosure: Advanced features are hidden until needed
- Consistent Patterns: Similar actions work the same way everywhere
- Graceful Degradation: Core features work even when optional features fail
- Performance Budget: Each feature must not significantly impact load time

9. Interactive Study Tools

Study tools enhance learning by providing active engagement with course material. Each tool is designed to complement reading and reinforce understanding through different cognitive mechanisms.

9.1 Quick Reference Sheets

Quick reference sheets provide instant access to essential formulas and concepts without leaving the lesson view. Each course has curated reference content organized by module.

- Keyboard shortcut (Q) opens overlay instantly
- Searchable formula database with fuzzy matching
- LaTeX-rendered equations with copy-to-clipboard
- Module-filtered view for focused reference
- Print-friendly format for exam cheat sheets
- Per-course customization by content authors

9.2 Flashcard System

Flashcards use active recall to strengthen memory. The system supports basic flip cards and can be extended with spaced repetition for optimized learning schedules.

- Keyboard shortcut (F) for instant access
- Flip animation revealing answer on click/spacebar
- Navigation with arrow keys or swipe gestures
- Progress tracking (cards reviewed, correct rate)
- Per-module card decks for focused study
- Optional spaced repetition scheduling (SM-2 algorithm)
- Custom card creation (future: user-created cards)

9.3 Personal Notes

Personal notes allow students to annotate lessons with their own understanding. Notes persist across sessions and can be exported for external use.

- Per-lesson note storage with auto-save
- Markdown support for rich formatting
- Keyboard shortcut (N) for quick access

- Search across all notes
- Export all notes as Markdown or PDF
- Note indicator on lesson sidebar

9.4 Quiz System

Quizzes provide assessment and reinforcement. Each module can have an associated quiz that tests understanding of key concepts. Quiz results are tracked and best scores are saved.

- Multiple choice questions with single correct answer
- Immediate feedback with explanations
- Score tracking and best score display
- Retry functionality with randomized question order
- Time-optional mode (future: timed quizzes)
- Pass threshold for module completion badges

9.5 Worked Examples

Worked examples demonstrate problem-solving methodology step-by-step. They are embedded in lessons and can be expanded/collapsed to avoid overwhelming the reader.

- Collapsible boxes to reduce visual clutter
- Step-by-step solution with explanations
- LaTeX rendering for mathematical content
- Diagram support for visual explanations
- Related exercise links for practice

10. Gamification System

Gamification elements increase engagement and motivation by providing tangible progress indicators and rewards. The system is designed to encourage consistent study habits without becoming distracting.

10.1 Daily Streak System

Element	Behavior	Reward
Current Streak	Consecutive days with at least 1 lesson completed	Fire icon animation, count display
Longest Streak	Historical best streak	Badge at milestones (7, 30, 100 days)
Streak Freeze	One missed day forgiveness per week (future)	Preserves streak, no bonus
Streak Recovery	Notification if streak at risk (future)	Reminder to study today

10.2 Achievement Badges

Badge	Requirement	Tier
First Steps	Complete first lesson	Bronze
Module Master	Complete a module	Bronze
Course Graduate	Complete a course	Silver
Quiz Ace	Score 100% on any quiz	Bronze
Week Warrior	7-day streak	Silver
Month Master	30-day streak	Gold
Polyglot	Complete 3+ courses	Gold
Perfect Score	100% on all quizzes in a course	Platinum
Century	Complete 100 lessons	Platinum

10.3 Celebration Animations

Celebrations provide positive feedback for achievements, making learning feel rewarding:

- Lesson Complete: Subtle checkmark animation, progress bar update
- Module Complete: Modal popup with confetti, achievement card preview
- Course Complete: Full-screen celebration, certificate generation prompt
- Streak Milestone: Fire animation intensification, milestone badge
- Quiz Perfect Score: Star burst animation, score highlight
- New Achievement: Slide-in notification, badge showcase

10.4 Certificates

Certificates provide formal recognition of course completion. Students can download or share their achievements:

- Generated on course completion
- Includes: Course name, completion date, student name (optional)
- PDF download with branded template
- Shareable link (future: unique verification URL)
- Stored in browser for re-download

11. Technical Stack

The technical stack is chosen for Vercel compatibility, developer experience, and long-term maintainability. All technologies are well-documented, actively maintained, and have strong community support.

Layer	Technology	Purpose
Framework	Next.js 15	React framework with App Router, static generation
Language	TypeScript 5	Type safety, better IDE support, fewer bugs
Styling	Tailwind CSS 3	Utility-first CSS for rapid UI development
UI Components	shadcn/ui	Accessible, customizable React components
Math Rendering	KaTeX	Fast LaTeX rendering for equations
Search	Fuse.js	Client-side fuzzy search library
State	React Context	Lightweight global state management
Storage	LocalStorage	Browser-native persistence, no server needed
Icons	Lucide React	Beautiful, consistent SVG icons
Animation	Framer Motion	Smooth animations and transitions
PWA	next-pwa	Service worker for offline support

11.1 Package Dependencies

- next@15.x — Application framework
- react@19.x, react-dom@19.x — UI library
- typescript@5.x — Type system
- tailwindcss@3.x — Styling
- @radix-ui/react-* — UI primitives (via shadcn/ui)
- katex@0.16.x — LaTeX rendering
- fuse.js@7.x — Fuzzy search
- lucide-react@latest — Icons
- framer-motion@11.x — Animations

• next-pwa@5.x — PWA support (optional)
• clsx, tailwind-merge — Utility functions

12. Implementation Phases

Development is organized into five phases over approximately six weeks. Each phase delivers a usable increment, allowing for early testing and iterative improvement.

Phase 1: Foundation (Days 1-7)

- Initialize Next.js 15 project with TypeScript
- Configure Tailwind CSS with custom theme
- Install and configure shadcn/ui components
- Create course types and interfaces
- Implement course registry system
- Build hub landing page layout
- Create course card component
- Set up ESLint, Prettier, and VS Code settings
- Initialize Git repository with .gitignore

Phase 2: Course Viewer (Days 8-14)

- Build CourseViewer page with dynamic routing
- Create collapsible SidebarNav component
- Implement LessonContent with KaTeX rendering
- Add progress tracking with useProgress hook
- Create ProgressBar and completion indicators
- Implement lesson navigation (prev/next)
- Add reading time calculation
- Build mark complete functionality
- Create bookmark system

Phase 3: Study Tools (Days 15-21)

- Build QuickReference overlay component
- Create Flashcard system with flip animation
- Implement Notes component with auto-save
- Build Quiz system with scoring
- Add keyboard shortcuts system

- Create floating action buttons
- Implement celebration animations
- Build search functionality with Fuse.js

Phase 4: Content Integration (Days 22-35)

- Migrate Transformation Methods course
- Create Control Systems course (9 modules)
- Add worked examples from Ogata textbook
- Create practice exercises
- Build quiz question banks
- Create quick reference sheets
- Prepare flashcard decks
- Test all content for accuracy

Phase 5: Polish & Deploy (Days 36-42)

- Implement streak tracking system
- Add achievement badges
- Create certificate generator
- Optimize mobile responsiveness
- Add dark mode with persistence
- Implement PWA for offline support
- Performance optimization
- Deploy to Vercel
- Write documentation and README

13. Adding New Courses

Adding a new course requires minimal configuration. Follow these five steps:

Step 1: Create Course Directory

```
mkdir src/lib/content/my-new-course
```

Step 2: Create Required Files

- index.ts — Course metadata
- modules.ts — Module and lesson content
- examples.ts — Worked examples
- exercises.ts — Practice problems
- quizzes.ts — Quiz questions
- reference.ts — Quick reference content

Step 3: Define Course Metadata

```
// index.ts
import { Course } from '@/lib/courses/types';
import { modules } from './modules';

export const myNewCourse: Course = {
id: 'my-new-course',
slug: 'my-new-course',
title: 'My New Course',
subtitle: 'Course tagline',
description: 'Full description...',
category: 'electronics',
icon: 'Circuit',
color: '#10B981',
difficulty: 'intermediate',
estimatedHours: 20,
modules,
// ... other fields
};
```

Step 4: Register in Course Registry

```
// lib/courses/index.ts
import { myNewCourse } from '@/lib/content/my-new-course';

export const courses: Course[] = [
controlSystems,
transformationMethods,
```

```
myNewCourse, // Add here
];
```

Step 5: Verify

Your course automatically appears in the hub. No routing changes needed — the dynamic [slug] route handles everything.

14. GitHub & Vercel Deployment

14.1 Repository Setup

- Create GitHub repository (public or private)
- Initialize local Git: git init
- Add remote: git remote add origin
- Create .gitignore with Next.js defaults
- Create comprehensive README.md
- Add LICENSE file (MIT recommended)
- Push to main branch

14.2 Vercel Deployment

- Log in to Vercel (vercel.com)
- Click 'Import Project' → Import Git Repository
- Select your GitHub repository
- Configure: Framework Preset = Next.js
- Environment: No environment variables needed
- Click 'Deploy'
- Done! Auto-deploys on every push to main

14.3 Recommended .gitignore

```
# Dependencies
node_modules/
.pnp
.pnp.js

# Build
.next/
out/
build/
dist/

# Testing
coverage/

# Misc
```

```
.DS_Store
*.pem

# Debug
npm-debug.log*
yarn-debug.log*
yarn-error.log*

# Local env files
.env*.local

# Vercel
.vercel

# TypeScript
*.tsbuildinfo
next-env.d.ts
```

14.4 Vercel Configuration

```
// vercel.json (optional)
{
"framework": "nextjs",
"buildCommand": "npm run build",
"outputDirectory": ".next"
}
```

15. Keyboard Shortcuts

Keyboard shortcuts enable power users to navigate efficiently. All shortcuts work globally unless specified.

Key	Action	Context
ESC	Go back / Close modal	Global
/	Focus search	Hub
Q	Open Quick Reference	Course
F	Open Flashcards	Course
N	Open Notes	Course
←	Previous lesson	Course
→	Next lesson	Course
M	Toggle mark complete	Lesson
B	Toggle bookmark	Lesson
T	Toggle dark mode	Global
?	Show shortcuts help	Global

Implementation Notes:

- Shortcuts are disabled when typing in input fields
- Shortcuts display in footer help section and '?' modal
- Mobile: visible buttons replace shortcuts with touch targets
- Consider customizable shortcuts for accessibility (future)

16. Color Scheme

16.1 Category Colors

Category	Primary	Light Variant
Control Systems	#3B82F6	#DBEAFE
Signals & Systems	#8B5CF6	#EDE9FE
Electronics	#10B981	#D1FAE5
Power Systems	#F59E0B	#FEF3C7
Mathematics	#EC4899	#FCE7F3
Programming	#06B6D4	#CFFAFE
Communications	#F97316	#FFEDD5

16.2 UI Colors

- Primary: #1F4E79 — Headers, primary buttons, links
- Secondary: #2E75B6 — Secondary actions, accents
- Success: #10B981 — Completions, positive indicators
- Warning: #F59E0B — Streaks, attention items
- Error: #EF4444 — Errors, destructive actions
- Background Light: #FFFFFF — Main background
- Background Dark: #0F172A — Dark mode background
- Surface Light: #F8FAFC — Cards, elevated surfaces
- Surface Dark: #1E293B — Dark mode surfaces
- Text Primary: #1F2937 — Main text
- Text Secondary: #6B7280 — Muted text, descriptions
- Border: #E5E7EB — Dividers, borders

17. Future Roadmap

Version 2.0 — Enhanced Features

- User accounts with Vercel Postgres or Supabase
- Cross-device progress sync
- Offline PWA with full content caching
- Additional courses: Electric Circuits, DSP, Power Electronics
- Enhanced analytics dashboard
- Parent/teacher monitoring view
- Discussion forums per lesson

Version 3.0 — Advanced Features

- AI-powered Q&A; assistant (Vercel AI SDK)
- Adaptive learning paths based on performance
- Video content integration
- Interactive simulations and labs
- Mobile app via React Native or Expo
- LMS integration (Canvas, Moodle)
- Multi-language support (i18n)

Long-Term Vision

- Complete EE undergraduate curriculum
- Graduate-level content
- Industry certification programs
- Virtual lab partnerships
- Research integration
- Global community of learners

This implementation plan provides a complete blueprint for building EduHub. The Vercel-optimized architecture ensures seamless deployment and zero server costs, while the modular course system enables easy content expansion. Regular iteration based on student feedback will drive continuous improvement.