Planning

Sustainable 11-Week Development Plan

This plan prioritizes steady learning progression over ambitious deliverables, with built-in flexibility to prevent burnout while ensuring project completion.

Phase Overview

- **Phase 1 (Weeks 1-3):** Foundation & Planning
- **Phase 2 (Weeks 4-7):** Core Backend Development
- **Phase 3 (Weeks 8-10):** Frontend & Integration
- **Phase 4 (Week 11):** Polish & Documentation

Week 1: Requirements & Project Foundation

Daily Task Breakdown (10 hours total)

Monday (2 hours):

- Study requirements engineering concepts (30 minutes)
- Create 3 detailed user personas (Student, Faculty, Admin) (90 minutes)

Tuesday (1.5 hours):

- Research existing academic management systems for inspiration (45 minutes)
- Document pain points and user needs for each persona (45 minutes)

Wednesday (2 hours):

- Write 10-12 core user stories with acceptance criteria (2 hours)

- Focus on authentication, course browsing, basic enrollment

Thursday (1.5 hours):

- Write 8-10 additional user stories for faculty and admin features (90 minutes)

Friday (1.5 hours):

- Create requirements prioritization matrix (MoSCoW method) (45 minutes)
- Document non-functional requirements (performance, security) (45 minutes)

Weekend (1.5 hours):

- Review and refine all requirements documentation (45 minutes)
- Create project scope statement and constraints document (45 minutes)

Week 1 Objectives:

- Master professional requirements documentation
- Understand user-centered design thinking
- Learn scope management techniques

Week 1 Outcomes:

- 20-25 well-defined user stories with acceptance criteria
- Complete stakeholder analysis document
- Requirements traceability matrix
- Project scope and constraints clearly defined

Learning Achievements:

- Requirements elicitation techniques
- User story writing standards
- Stakeholder analysis methods
- Project scoping strategies

Progress: 8% complete

Week 2: System Design & Architecture

Daily Task Breakdown (10 hours total)

Monday (2.5 hours):

- Study database design principles and normalization (45 minutes)
- Design initial database schema with 5-6 core tables (105 minutes)

Tuesday (2 hours):

- Create entity-relationship diagram (90 minutes)
- Define table relationships and constraints (30 minutes)

Wednesday (2 hours):

- Study REST API design principles (30 minutes)
- Design API endpoints for user and course management (90 minutes)

Thursday (1.5 hours):

- Create system architecture diagram (90 minutes)

Friday (1 hour):

- Design basic UI wireframes for login and dashboard screens (60 minutes)

Weekend (1 hour):

- Review and document all design decisions (60 minutes)

Week 2 Objectives:

- Learn database design fundamentals
- Understand REST API principles
- Master system architecture documentation

Week 2 Outcomes:

- Complete normalized database schema (6-8 tables)
- System architecture diagram with component relationships
- API endpoint specifications for core features
- Basic UI wireframes for main screens

Learning Achievements:

- Database normalization and relationship design
- REST API design patterns
- System documentation standards
- Architecture thinking and diagramming

Progress: 18% complete

Week 3: Development Environment & Basic Setup

Daily Task Breakdown (10 hours total)

Monday (2.5 hours):

- Install IntelliJ IDEA, PostgreSQL, Node.js (45 minutes)
- Configure development environment and tools (105 minutes)

Tuesday (2 hours):

- Create Spring Boot project structure (60 minutes)
- Set up Git repository with proper .gitignore (30 minutes)
- Configure PostgreSQL database connection (30 minutes)

Wednesday (2 hours):

- Create User entity class with JPA annotations (90 minutes)
- Test database connection and basic entity creation (30 minutes)

Thursday (2 hours):

- Set up basic Spring Security configuration (90 minutes)
- Create simple user registration endpoint (30 minutes)

Friday (1 hour):

- Test user creation and basic authentication (60 minutes)

Weekend (0.5 hours):

- Document setup process and configuration (30 minutes)

Week 3 Objectives:

- Master development environment setup
- Learn Spring Boot project structure
- Understand basic JPA entity creation

Week 3 Outcomes:

- Fully configured development environment
- Spring Boot project with database connectivity
- Basic User entity with database persistence
- Simple authentication framework working

Learning Achievements:

- Professional development environment setup
- Spring Boot configuration basics
- JPA entity creation and mapping
- Basic Spring Security concepts

Progress: 30% complete

Week 4: User Management Foundation

Daily Task Breakdown (10 hours total)

Monday (2.5 hours):

- Study Spring Data JPA repository patterns (30 minutes)
- Create UserRepository interface (30 minutes)
- Implement basic CRUD operations (90 minutes)

Tuesday (2 hours):

- Learn service layer architecture patterns (30 minutes)
- Create UserService with business logic (90 minutes)

Wednesday (2 hours):

- Study REST controller patterns (30 minutes)
- Create UserController with basic endpoints (90 minutes)

Thursday (2 hours):

- Learn JUnit testing basics (30 minutes)
- Write 5-6 unit tests for UserService (90 minutes)

Friday (1 hour):

- Test all user management endpoints with Postman (60 minutes)

Weekend (0.5 hours):

- Refactor and improve code quality (30 minutes)

Week 4 Objectives:

- Master repository pattern implementation
- Learn service layer design principles

- Understand basic REST controller creation

Week 4 Outcomes:

- Complete user management system (CRUD operations)
- UserRepository, UserService, UserController implemented
- 5-6 unit tests covering core functionality
- Working REST endpoints for user operations

Learning Achievements:

- Spring Data JPA repository patterns
- Service layer architecture
- REST controller implementation
- Basic unit testing with JUnit

Progress: 45% complete

Week 5: Course Management System

Daily Task Breakdown (10 hours total)

Monday (2.5 hours):

- Create Course entity with proper relationships (90 minutes)
- Set up Course-User relationships (instructor assignment) (60 minutes)

Tuesday (2 hours):

- Create CourseRepository and CourseService (90 minutes)
- Implement course creation and basic validation (30 minutes)

Wednesday (2 hours):

- Create CourseController with REST endpoints (90 minutes)
- Add course search and filtering capabilities (30 minutes)

Thursday (2 hours):

- Study enrollment concepts and design Enrollment entity (60 minutes)
- Create basic enrollment relationships (60 minutes)

Friday (2 hours):

- Write unit tests for course management functionality (90 minutes)
- Test course endpoints thoroughly (30 minutes)

Weekend (1.5 hours):

- Implement basic enrollment creation (no complex validation yet) (90 minutes)

Week 5 Objectives:

- Learn entity relationship mapping
- Master complex business logic implementation
- Understand enrollment system design

Week 5 Outcomes:

- Complete course management system
- Course entity with instructor relationships
- Basic enrollment entity and creation
- Course search and filtering functionality

Learning Achievements:

- JPA relationship mapping (@OneToMany, @ManyToOne)
- Complex entity design patterns
- Business logic validation

- Search and filtering implementation

Progress: 62% complete

Week 6: Enrollment Business Logic

Daily Task Breakdown (10 hours total)

Monday (2 hours):

- Study business rule implementation patterns (30 minutes)
- Implement prerequisite validation logic (90 minutes)

Tuesday (2 hours):

- Create schedule conflict detection algorithms (2 hours)

Wednesday (2 hours):

- Implement enrollment capacity management (90 minutes)
- Add basic waitlist functionality (30 minutes)

Thursday (2 hours):

- Create EnrollmentService with complete business rules (2 hours)

Friday (1.5 hours):

- Add enrollment REST endpoints (90 minutes)

Weekend (0.5 hours):

- Write unit tests for enrollment business logic (30 minutes)

Week 6 Objectives:

- Master complex business rule implementation
- Learn algorithm design for academic systems
- Understand validation frameworks

Week 6 Outcomes:

- Complete enrollment system with validation
- Prerequisite checking algorithms
- Schedule conflict detection
- Enrollment capacity management

Learning Achievements:

- Complex business rule implementation
- Algorithm design and validation
- Advanced service layer patterns
- Business logic testing strategies

Progress: 75% complete

Week 7: Grade Management & API Completion

Daily Task Breakdown (10 hours total)

Monday (2 hours):

- Create Grade entity and relationships (90 minutes)
- Design GPA calculation algorithms (30 minutes)

Tuesday (2 hours):

- Implement GradeService with calculation logic (2 hours)

Wednesday (2 hours):

- Create grade management REST endpoints (90 minutes)
- Add JWT token authentication to all endpoints (30 minutes)

Thursday (2 hours):

- Study API documentation standards (30 minutes)
- Create comprehensive API documentation (90 minutes)

Friday (1.5 hours):

- Test all API endpoints thoroughly (90 minutes)

Weekend (0.5 hours):

- Set up CORS configuration for frontend integration (30 minutes)

Week 7 Objectives:

- Learn academic calculation algorithms
- Master API security implementation
- Understand comprehensive testing approaches

Week 7 Outcomes:

- Complete grade management system
- GPA calculation algorithms working
- All REST endpoints secured with JWT
- Comprehensive API documentation

Learning Achievements:

- Academic algorithm implementation
- JWT authentication and security
- API documentation standards

- Comprehensive backend testing

Progress: 85% complete

Week 8: Frontend Foundation

Daily Task Breakdown (10 hours total)

Monday (2.5 hours):

- Study Vue.js basics and component concepts (60 minutes)
- Set up Vue.js project structure (90 minutes)

Tuesday (2 hours):

- Learn Vue.js routing and create basic navigation (2 hours)

Wednesday (2 hours):

- Create login and registration components (2 hours)

Thursday (2 hours):

- Implement authentication state management (90 minutes)
- Connect authentication to backend API (30 minutes)

Friday (1 hour):

- Create basic dashboard layout for different user roles (60 minutes)

Weekend (0.5 hours):

- Test authentication flow end-to-end (30 minutes)

Week 8 Objectives:

- Learn Vue.js fundamentals
- Master component-based architecture
- Understand frontend-backend integration

Week 8 Outcomes:

- Vue.js project with proper structure
- Working authentication system
- Basic navigation and routing
- Role-based dashboard layouts

Learning Achievements:

- Vue.js component architecture
- Frontend routing and navigation
- State management patterns
- API integration from frontend

Progress: 90% complete

Week 9: Core UI Implementation

Daily Task Breakdown (10 hours total)

Monday (2.5 hours):

- Create course browsing interface (2.5 hours)

Tuesday (2 hours):

- Implement student enrollment forms (2 hours)

Wednesday (2 hours):

- Create faculty course management interface (2 hours)

Thursday (2 hours):

- Implement grade entry forms for faculty (2 hours)

Friday (1 hour):

- Add basic styling and responsive design (60 minutes)

Weekend (0.5 hours):

- Test all user workflows end-to-end (30 minutes)

Week 9 Objectives:

- Master form handling in Vue.js
- Learn responsive design principles
- Understand complete workflow implementation

Week 9 Outcomes:

- Complete student interface with course browsing and enrollment
- Faculty interface for course and grade management
- Basic admin panel for user management
- Responsive design working on mobile devices

Learning Achievements:

- Advanced Vue.js form handling
- Responsive design implementation
- Complete workflow development
- User experience design principles

Progress: 95% complete

Week 10: Integration & Testing

Daily Task Breakdown (10 hours total)

Monday (2.5 hours):

- Perform comprehensive integration testing (2.5 hours)

Tuesday (2 hours):

- Fix integration bugs and issues (2 hours)

Wednesday (2 hours):

- Add error handling throughout application (2 hours)

Thursday (2 hours):

- Implement loading states and user feedback (2 hours)

Friday (1 hour):

- Performance testing and optimization (60 minutes)

Weekend (0.5 hours):

- Final bug fixes and polish (30 minutes)

Week 10 Objectives:

- Master integration testing approaches
- Learn error handling patterns
- Understand performance optimization

Week 10 Outcomes:

- Fully tested and integrated system
- Comprehensive error handling
- Performance optimizations applied
- Production-ready application

Learning Achievements:

- Integration testing strategies
- Error handling best practices
- Performance optimization techniques
- Production readiness assessment

Progress: 98% complete

Week 11: Documentation & Presentation

Daily Task Breakdown (10 hours total)

Monday (3 hours):

- Create comprehensive technical documentation (3 hours)

Tuesday (2 hours):

- Write user manuals for each user type (2 hours)

Wednesday (2 hours):

- Prepare deployment documentation (2 hours)

Thursday (2 hours):

- Create project presentation materials (2 hours)

Friday (1 hour):

- Final review and project submission preparation (60 minutes)

Week 11 Objectives:

- Master technical documentation standards
- Learn presentation preparation techniques
- Understand project delivery processes

Week 11 Outcomes:

- Complete technical documentation
- User manuals for all user types
- Deployment and installation guides
- Professional project presentation

Learning Achievements:

- Technical writing and documentation
- Project presentation skills
- Professional delivery standards
- Portfolio development

Progress: 100% complete

Built-in Wellness Features

Daily Structure:

- Maximum 2.5 hours per day to prevent burnout

- 15-minute breaks every hour during coding sessions
- Weekend work limited to 1 hour maximum

Flexibility Mechanisms:

- Each week has 0.5-1 hour buffer for unexpected challenges
- Tasks can be moved between days within the same week
- Scope reduction options identified for each phase

Progress Tracking:

- Weekly self-assessment of energy levels and motivation
- Adjustment triggers if falling behind schedule
- Celebration of weekly achievements to maintain motivation

This plan ensures steady progress while maintaining your wellbeing and allowing for the natural learning curve of software development.

Back up plan: JFX

JavaFX Fallback Strategy - Realistic Assessment

This is a smart contingency plan. The architecture you've built so far actually supports this transition well.

Feasibility of the Switch

When to make the decision: Week 8 is the optimal decision point. By then you'll have:

- Complete backend with all APIs working

- Clear understanding of Vue.js complexity
- Realistic assessment of remaining time

Technical transition effort: 2-3 weeks to build JavaFX frontend consuming your existing APIs.

What You Keep vs. What Changes

Preserved Investment:

- Entire backend system (Weeks 3-7 of work)
- All business logic and data models
- Database design and implementation
- REST API endpoints
- Authentication system

Nothing is wasted - your backend becomes an API server that JavaFX calls instead of Vue.js.

What Changes:

- Frontend technology only
- API consumption method (HTTP calls instead of direct Vue.js integration)
- User interface implementation approach

JavaFX Implementation Timeline

Week 8: JavaFX Foundation (instead of Vue.js)

- Set up JavaFX project structure
- Create main application window and navigation
- Implement login forms connecting to your existing auth API

Week 9: Core UI Implementation

- Student dashboard consuming course and enrollment APIs
- Faculty interface for course management

- Admin panel for user management
- All connected to your existing backend APIs

Week 10-11: Polish and completion

- Error handling and user feedback
- UI improvements and testing
- Documentation and presentation

Advantages of JavaFX Fallback

Simpler integration: Direct method calls to HTTP client rather than Vue.js framework complexity.

Familiar environment: Single IDE (IntelliJ) for entire project instead of switching between development environments.

Reduced debugging complexity: No CORS issues, no frontend build process complications, no separate server management.

Still impressive: Desktop applications demonstrate solid programming skills and complete system architecture.

Architecture After Switch

```
JavaFX Desktop App → HTTP Client → Your Spring Boot API → PostgreSQL
```

Your Spring Boot backend continues running as an API server. JavaFX makes HTTP requests to it just like Vue.js would have.

Implementation Example

```java
// JavaFX calls your existing API
public class CourseService {
    private final HttpClient httpClient = HttpClient.newHttpClient();
    private final String API_BASE = "http://localhost:8080/api";
    public List<Course> getAllCourses() {
        HttpRequest request = HttpRequest.newBuilder()
            .uri(URI.create(API_BASE + "/courses"))
            .GET()
            .build();

        // Parse JSON response and return Course objects
        // Your existing API handles all the business logic
    }
}
```

Strategic Timing

Week 8 decision criteria:

- Can you create a basic Vue.js component that displays data from your API within 2-3 hours?
- Are you comfortable with Vue.js routing and state management concepts?
- Do you have energy for learning frontend framework patterns?

If any answer is "no" or you're feeling overwhelmed: Switch to JavaFX without guilt or hesitation.

Portfolio Considerations

JavaFX advantages for academic projects:

- Demonstrates complete system architecture understanding
- Shows proficiency with enterprise Java development patterns
- Proves ability to build functional, user-friendly applications
- Less common skill that can differentiate you from web-only developers

Web development advantages:

- More common in current job market
- Demonstrates modern development practices
- Shows frontend-backend integration skills

Both approaches have professional value. The key is completing a working system that demonstrates your software engineering competency.

Recommendation

Plan to make this decision at Week 8, not before. You might surprise yourself with how quickly Vue.js concepts click once you have a solid backend foundation.

However, having this fallback option reduces pressure and anxiety, which actually increases your chances of success with either approach. Knowing you have a viable alternative allows you to take appropriate risks with the Vue.js implementation without fear of project failure.

The most important outcome is completing a functional academic management system that demonstrates professional software development practices. The specific frontend technology is secondary to that goal.