Study Pal Development Roadmap - Complete Instruction Guide

📋 Project Overview & Success Criteria

Goal: Build a full-stack productivity web application with user authentication, task management, Pomodoro timer, and deployment to production.

Success Metrics:

- ✅ Users can register, login, and manage profiles
- ✅ Complete task CRUD with priority and deadline management
- ✅ Functional Kanban board with drag-and-drop
- ✅ Working Pomodoro timer with session tracking
- ✅ Daily planning algorithm with task assignment
- ✅ Personal diary with performance ratings
- ✅ Dashboard showing all key metrics
- ✅ Deployed and accessible online

PHASE 1: Backend Foundation (Days 1-10)

Week 1: Core Backend Setup

Day 1-2: Complete Task Management System

Objective: Enable full CRUD operations for tasks

Tasks to Complete:

1. Create Task model class with all required fields (id, userId, title, description, priority, deadline, estimatedHours, status, timestamps)
2. Create TaskRepository interface with custom query methods
3. Create TaskService class with business logic methods
4. Create TaskController with REST endpoints
5. Test all task endpoints in Postman

Testing Checklist:

- [ ] POST /api/tasks - Create new task
- [ ] GET /api/tasks/user/{userId} - Get all user tasks
- [ ] GET /api/tasks/{id} - Get specific task
- [ ] PUT /api/tasks/{id} - Update task
- [ ] PUT /api/tasks/{id}/status - Update task status
- [ ] DELETE /api/tasks/{id} - Delete task
- [ ] GET /api/tasks/user/{userId}/today - Get today's tasks

Success Criteria: All task endpoints return expected status codes and data

Day 3-4: JWT Authentication System

Objective: Replace basic auth with proper JWT token system

Tasks to Complete:

1. Add Spring Security JWT dependencies to pom.xml
2. Create JWT utility class for token generation and validation
3. Create AuthController with login/register endpoints
4. Create JwtAuthenticationFilter for request filtering
5. Update SecurityConfig for JWT-based authentication
6. Add password encryption with BCrypt
7. Create AuthService for authentication logic

Testing Checklist:

- [ ] POST /api/auth/register - User registration
- [ ] POST /api/auth/login - User login returns JWT token
- [ ] Protected endpoints require valid JWT token
- [ ] Invalid/expired tokens return 401 Unauthorized
- [ ] Password encryption works correctly

Success Criteria: Users can register, login, and access protected endpoints with JWT tokens

Day 5-6: Advanced Task Features

Objective: Add business logic for task management

Tasks to Complete:

1. Create methods for overdue task detection
2. Add task rollover functionality (incomplete tasks move to next day)
3. Implement priority-based task sorting
4. Add task search and filtering capabilities
5. Create bulk task operations (mark multiple as complete)
6. Add task statistics calculations

Testing Checklist:

- [ ] GET /api/tasks/user/{userId}/overdue - Get overdue tasks
- [ ] GET /api/tasks/user/{userId}/upcoming - Get upcoming tasks
- [ ] Task rollover works at midnight (test manually)
- [ ] Priority sorting functions correctly
- [ ] Statistics calculations are accurate

Success Criteria: Advanced task features work reliably with edge cases handled

Day 7-8: Data Validation & Error Handling

Objective: Make backend robust and production-ready

Tasks to Complete:

1. Add input validation annotations to all models
2. Create global exception handler class
3. Add custom exception classes for business logic errors
4. Implement proper error response formatting
5. Add logging throughout the application
6. Create health check endpoint

Testing Checklist:

- [ ] Invalid input returns 400 Bad Request with clear messages
- [ ] Missing required fields are properly validated
- [ ] Database errors are handled gracefully
- [ ] All endpoints log requests and responses
- [ ] GET /actuator/health returns application status

Success Criteria: Backend handles all error scenarios gracefully with meaningful messages

Day 9-10: Backend Performance & Database Optimization

Objective: Optimize backend for production performance

Tasks to Complete:

1. Add database indexes for frequently queried fields
2. Implement pagination for task lists
3. Add caching for frequently accessed data
4. Optimize database queries
5. Add request rate limiting
6. Configure application properties for different environments

Testing Checklist:

- [ ] Large task lists load quickly with pagination
- [ ] Database queries are optimized (check logs)
- [ ] Rate limiting prevents abuse
- [ ] Different environment configs work correctly

Success Criteria: Backend performs well under load and is production-ready

---

PHASE 2: Frontend Development (Days 11-20)

Week 2: React Application Setup

Day 11-12: Project Setup & Authentication

Objective: Create React app with login/register functionality

Tasks to Complete:

1. Create new React application using Create React App
2. Set up project structure (components, pages, services, utils)
3. Install required dependencies (axios, react-router-dom, etc.)
4. Create authentication service for API calls
5. Build login and register forms
6. Implement JWT token storage and management
7. Create protected route wrapper component

Testing Checklist:

- [ ] Users can register new accounts
- [ ] Users can login with valid credentials
- [ ] JWT tokens are stored securely
- [ ] Protected routes redirect to login when unauthenticated
- [ ] Logout functionality clears tokens

Success Criteria: Complete authentication flow works from frontend to backend

Day 13-14: Dashboard & User Profile

Objective: Create main dashboard and profile management

Tasks to Complete:

1. Create dashboard layout with navigation
2. Build user profile page with edit functionality
3. Display upcoming tasks widget on dashboard
4. Add task completion statistics
5. Create responsive design for mobile devices
6. Implement theme/styling system

Testing Checklist:

- [ ] Dashboard shows key user metrics
- [ ] Profile editing saves changes to backend
- [ ] Navigation works across all pages
- [ ] Mobile responsive design functions properly
- [ ] Visual design is consistent and professional

Success Criteria: Professional-looking dashboard that displays relevant user information

Day 15-16: Task Management Interface

Objective: Build complete task management UI

Tasks to Complete:

1. Create task creation form with validation

2. Build task list view with sorting and filtering
3. Implement task editing functionality
4. Add task deletion with confirmation
5. Create priority and status indicator components
6. Build task search functionality

Testing Checklist:

- [ ] Users can create tasks with all required fields
- [ ] Task list displays properly formatted information
- [ ] Editing tasks updates backend correctly
- [ ] Search and filtering work accurately
- [ ] UI feedback for all user actions

Success Criteria: Users can manage all aspects of their tasks through the interface

Day 17-18: Kanban Board

Objective: Create interactive Kanban board for task management

Tasks to Complete:

1. Build three-column Kanban layout (Todo, In Progress, Completed)
2. Implement drag-and-drop functionality
3. Connect drag-and-drop to backend status updates
4. Add visual indicators for task priorities and deadlines
5. Make Kanban board responsive for mobile
6. Add animations and smooth transitions

Testing Checklist:

- [ ] Tasks can be dragged between columns
- [ ] Status updates save to backend immediately
- [ ] Visual feedback during drag operations
- [ ] Mobile touch interactions work properly
- [ ] Overdue tasks are visually highlighted

Success Criteria: Intuitive Kanban board that updates task status through drag-and-drop

Day 19-20: Frontend Polish & Integration

Objective: Complete frontend integration and polish

Tasks to Complete:

1. Add loading states and error messages throughout app
2. Implement form validation with user feedback
3. Create confirmation dialogs for destructive actions
4. Add keyboard shortcuts for power users
5. Optimize performance and bundle size
6. Test all user flows end-to-end

Testing Checklist:

- [ ] All forms validate input and show helpful errors
- [ ] Loading states prevent user confusion
- [ ] Error messages are clear and actionable
- [ ] App performs well on slower devices
- [ ] All features work together seamlessly

Success Criteria: Polished, professional frontend with excellent user experience

---

PHASE 3: Advanced Features (Days 21-30)

Week 3: Core Productivity Features

Day 21-22: Pomodoro Timer

Objective: Build fully functional Pomodoro timer system

Tasks to Complete:

1. Create backend models for Pomodoro sessions and settings
2. Build timer component with start/pause/reset functionality
3. Implement customizable work/break intervals
4. Add session tracking and statistics
5. Connect timer to task completion
6. Create timer notifications and alerts

Testing Checklist:

- [ ] Timer counts down accurately
- [ ] Work and break sessions alternate correctly
- [ ] User can customize time intervals
- [ ] Sessions are saved to database
- [ ] Timer integrates with daily task list

Success Criteria: Reliable Pomodoro timer that helps users track study time

Day 23-24: Planning Algorithm

Objective: Build intelligent daily task assignment system

Tasks to Complete:

1. Create backend service for daily task calculation
2. Implement priority-based task assignment algorithm
3. Build planning interface for user input (available hours, activities)
4. Add conflict resolution when too many high-priority tasks
5. Create automatic task rollover system
6. Add planning statistics and insights

Testing Checklist:

- [ ] Algorithm assigns appropriate tasks based on available time
- [ ] High-priority conflicts are presented to user for resolution
- [ ] Incomplete tasks roll over to next day automatically
- [ ] Planning recommendations are helpful and accurate

Success Criteria: Smart planning system that optimizes daily task assignments

Day 25-26: Personal Diary System

Objective: Create reflection and performance tracking system

Tasks to Complete:

1. Create diary entry model with rating calculations
2. Build diary entry form with structured questions
3. Implement automatic performance scoring based on completed tasks and study time
4. Create diary history view and insights
5. Add daily reflection prompts
6. Display previous day's reflections on dashboard

Testing Checklist:

- [ ] Users can create daily diary entries
- [ ] Performance ratings calculate automatically and accurately
- [ ] Diary history is easily browsable
- [ ] Previous day's insights show on dashboard
- [ ] Reflection prompts encourage thoughtful entries

Success Criteria: Meaningful diary system that provides insights into productivity patterns

Day 27-28: Dashboard Integration

Objective: Create comprehensive dashboard showing all key metrics

Tasks to Complete:

1. Build dashboard widgets for all major features
2. Add data visualization for productivity trends
3. Create quick action buttons for common tasks
4. Implement dashboard customization options
5. Add motivational elements and progress tracking
6. Ensure real-time updates across all components

Testing Checklist:

- [ ] Dashboard provides complete overview of user's productivity
- [ ] Data visualizations are clear and helpful
- [ ] Quick actions work from dashboard
- [ ] Real-time updates reflect changes immediately

- [ ] Dashboard motivates continued use

Success Criteria: Central hub that gives users complete control and insight into their productivity

Day 29-30: Feature Integration & Testing

Objective: Ensure all features work together seamlessly

Tasks to Complete:

1. Test all feature interactions and data flow
2. Fix any integration bugs or conflicts
3. Optimize performance across all features
4. Add comprehensive error handling
5. Create user onboarding flow
6. Write user documentation

Testing Checklist:

- [ ] All features work independently and together
- [ ] Data flows correctly between components
- [ ] Performance is acceptable under realistic usage
- [ ] New users can understand and use all features
- [ ] Edge cases are handled appropriately

Success Criteria: Fully integrated application ready for production deployment

---

PHASE 4: Quality Assurance & Testing (Days 31-35)

Day 31-32: Automated Testing

Objective: Implement comprehensive testing suite

Tasks to Complete:

1. Write unit tests for all backend services
2. Create integration tests for API endpoints
3. Add frontend component testing
4. Implement end-to-end testing for critical user flows
5. Set up continuous integration pipeline
6. Add code quality checks and linting

Success Criteria: Automated tests catch bugs and ensure code quality

Day 33-34: Manual Testing & Bug Fixes

Objective: Thoroughly test application from user perspective

Tasks to Complete:

1. Create comprehensive test scenarios for all features
2. Test on multiple browsers and devices
3. Perform stress testing with large amounts of data
4. Test edge cases and error scenarios
5. Fix all discovered bugs and issues
6. Validate accessibility compliance

Success Criteria: Application works flawlessly across all supported environments

Day 35: Performance Optimization

Objective: Optimize application for production performance

Tasks to Complete:

1. Analyze and optimize frontend bundle size
2. Implement lazy loading for non-critical components
3. Optimize database queries and add indexes
4. Add caching layers where appropriate
5. Configure compression and optimization
6. Test performance under realistic load

Success Criteria: Application loads quickly and performs well under normal usage

---

PHASE 5: Production Configuration (Days 36-40)

Day 36-37: Environment Configuration

Objective: Prepare application for different deployment environments

Tasks to Complete:

1. Create separate configuration files for development, staging, and production
2. Implement environment variable management
3. Configure logging for production monitoring
4. Set up database connection pooling
5. Add health checks and monitoring endpoints
6. Configure security headers and HTTPS settings

Success Criteria: Application can be deployed to any environment with proper configuration

Day 38-39: Security Hardening

Objective: Secure application for production deployment

Tasks to Complete:

1. Review and update all security configurations
2. Implement proper CORS settings

3. Add input sanitization and SQL injection prevention
4. Configure secure session management
5. Add rate limiting and DDoS protection
6. Perform security audit and vulnerability testing

Success Criteria: Application meets security best practices for production use

Day 40: Documentation & Deployment Preparation

Objective: Prepare all deployment documentation and scripts

Tasks to Complete:

1. Create comprehensive deployment documentation
2. Write API documentation for future maintenance
3. Create database migration scripts
4. Prepare monitoring and alerting setup
5. Create backup and disaster recovery procedures
6. Test deployment process in staging environment

Success Criteria: Complete documentation and tested deployment process

---

PHASE 6: Cloud Deployment (Days 41-45)

Day 41-42: Database Deployment

Objective: Set up production database

Tasks to Complete:

1. Choose cloud database provider (MongoDB Atlas recommended)
2. Create production database cluster
3. Configure database security and access controls
4. Set up database monitoring and alerting
5. Create database backup procedures
6. Test database connectivity from application

Success Criteria: Production database is secure, monitored, and accessible

Day 43-44: Application Deployment

Objective: Deploy application to cloud platform

Tasks to Complete:

1. Choose deployment platform (Heroku, AWS, or DigitalOcean recommended)
2. Configure application for cloud deployment
3. Set up continuous deployment pipeline
4. Configure domain name and SSL certificate

5. Set up load balancing if needed
6. Test deployed application thoroughly

Success Criteria: Application is live and accessible at custom domain

Day 45: Monitoring & Launch

Objective: Set up monitoring and officially launch application

Tasks to Complete:

1. Configure application monitoring and alerting
2. Set up error tracking and logging
3. Test all features in production environment
4. Create user accounts and test complete workflows
5. Set up backup and maintenance procedures
6. Document post-deployment maintenance tasks

Success Criteria: Fully deployed, monitored, and operational Study Pal application

---

Post-Deployment Maintenance Plan

Daily Tasks

- Monitor application health and performance
- Review error logs and fix critical issues
- Check database performance and backups

Weekly Tasks

- Review user feedback and feature requests
- Update dependencies and security patches
- Analyze usage patterns and optimize accordingly

Monthly Tasks

- Comprehensive security review
- Performance optimization based on real usage data
- Plan and implement new features based on user needs

Success Validation Checklist

Technical Requirements ☑

- [ ] Application loads in under 3 seconds
- [ ] All CRUD operations work correctly
- [ ] Authentication and authorization function properly

- [ ] Mobile responsive design works on all devices
- [ ] No security vulnerabilities in production
- [ ] Database performance is optimized
- [ ] Error handling covers all edge cases

User Experience Requirements ☑

- [ ] New users can complete onboarding without confusion
- [ ] All features are intuitive and easy to use
- [ ] Application provides helpful feedback for all actions
- [ ] Users can accomplish all intended tasks efficiently
- [ ] Application motivates continued use

Business Requirements ☑

- [ ] Application meets all original specifications
- [ ] Deployment is stable and reliable
- [ ] Maintenance procedures are documented
- [ ] Application can scale with increased usage
- [ ] Future feature additions are possible

Final Goal: A professional, fully-functional Study Pal application that demonstrates your full-stack development skills and provides real value to users.

This roadmap will take approximately 45 days of focused development. Follow each phase systematically, complete all testing checklists, and don't move to the next phase until all success criteria are met. This approach ensures you build a robust, professional application suitable for your portfolio and real-world use.