[X02] Final Project

Framework Programming - 2025/2026

Moch. Nafkhan Alzamzami, S.T., M.T.
Department of Informatics
Institut Teknologi Sepuluh Nopember

1. Project Description

This is an individual project. Create a complete web application that demonstrates your mastery of framework programming concepts. Your project should showcase modern development practices and solve a real-world problem. Choose one of the project specifications in Section 4.

If you have any questions about the project requirements or need clarification, feel free to ask the lecturer.

1.1. Framework Technologies

- For frontend, use:
  → a modern frontend framework, like:
    - React
    - Vue
    - Svelte
    - Angular
  → or a template engine, like:
    - EJS
    - Handlebars
- For backend, use NestJS.

1.2. Resources & Learning

You are welcome and encouraged to use any resources to complete this project:

- Online documentation, tutorials, and guides
- Stack Overflow, GitHub, and other developer communities
- AI assistants (ChatGPT, Claude, GitHub Copilot, etc.)
- Code examples and open-source projects as references
- Discussion with peers (but write your own code)

What matters is your understanding: You must be able to explain (almost) every line of code in your project during the presentation.

1.3. Deliverables

1. **Source Code**
2. **Presentation**
   - As a video, containing:
     → Live demonstration for each user story.
     → Thorough explanations of the source code.
     → Your face visible via webcam throughout the entire video (picture-in-picture or side-by-side layout).
   - Upload the video to YouTube as unlisted.
   - Recommended tools: OBS Studio, Zoom recording, or any screen recording software with webcam overlay.
3. **Additional Files**
   - `docs/main.pdf`: Complete project documentation
     → You may use any word processor application (e.g. Typst, GDocs, or MS Word).
   - `profile.txt` with format:

```txt
{NRP}
```

```
{Nama}
{YouTube Link}
```

2. Common Requirements

All projects must implement the following core features:

2.1. Authentication & Authorization

1. **User Registration**
   - Implement secure user registration with proper validation
   - Store passwords using secure hashing (bcrypt, argon2, etc.)
   - Collect relevant user information based on your project needs
   - Input validation (email format, password strength, required fields)
2. **User Login with JWT**
   - Implement JWT-based authentication
   - Generate access tokens upon successful login
   - Store JWT securely on the client side
   - Include proper token expiration
   - Implement refresh tokens for extended sessions
3. **Authorization & Access Control**
   - Implement role-based access control (RBAC) if your project requires multiple user types
   - Protect routes/endpoints based on authentication status
   - Implement authorization guards/middleware
   - Return appropriate HTTP status codes (401 Unauthorized, 403 Forbidden)
4. **Security Best Practices**
   - Never store passwords in plain text
   - Implement proper CORS configuration
   - Validate and sanitize all user inputs
   - Protect against common vulnerabilities (SQL injection, XSS)

2.2. Core Functionality

1. **File Upload**
   - Implement file upload functionality for images or documents
   - Store files in local server storage
   - Validate file types and size limits
   - Associate uploaded files with relevant entities (users, items, etc.)
2. **Search & Filter**
   - Implement search functionality for main entities
   - Provide filtering options based on relevant attributes (categories, dates, status, etc.)
   - Support pagination for list views

3. Project Choices

Important Note: You are NOT required to implement all user stories listed for your chosen project. What matters is demonstrating your mastery of framework programming concepts and skills. You may:

- Skip certain user stories if you have valid technical or time constraints
- Implement alternative features that better showcase your abilities
- Simplify or modify user stories while maintaining core functionality

The evaluation focuses on **code quality, architecture, and understanding of framework concepts** rather than checking off every requirement. Make sure to document your decisions and explain your reasoning in your project documentation.

3.1. Difficulty Levels

You have **three options** for this project:

- Choose a **pre-defined project** (Todo list or E-commerce Dashboard)
  → 🟢 **Beginner-Friendly** (Good for learning basics)
  → 🔴 **Advanced** (High complexity)
- 💡 **Custom Project** (Complexity assessed by instructor)
  → **Your Own Idea** - Create any project relevant to your interests (see Section 4 for details)

3.1.1. Extra Points Incentive

Choosing a more difficult project can earn you extra points:

- 🟢 **Beginner-Friendly**: Base scoring
- 🔴 **Advanced**: Up to +10 extra points for well-executed implementation
- 💡 **Custom Project**: Up to +5/+10 points based on assessed complexity, plus up to +5 bonus for exceptional innovation

Extra points are awarded based on:

- Successfully implementing complex features
- Clean architecture and code organization despite complexity
- Proper handling of edge cases and error scenarios
- Demonstrating deep understanding of framework concepts
- Innovation and creativity (for custom projects)

Note: A well-executed beginner project is better than a poorly executed advanced project. Choose wisely based on your current skill level and available time.

3.1.2. What makes a project more difficult?

- Complex data models with many relationships (e.g., courses with nested modules and lessons)
- Real-time features requiring state synchronization (e.g., drag-and-drop task boards)
- Advanced business logic and calculations (e.g., stock management, quiz scoring, budget tracking)
- Multiple user roles with fine-grained permissions (e.g., admin, manager, member with different access levels)
- Sophisticated algorithms (e.g., quiz randomization, ingredient-based recipe search)

4. Project Specifications

All projects must implement these core technical requirements (already included in the user stories below):

- **CRUD RESTful API**: Create, Read, Update, Delete operations via RESTful endpoints
- **Email**: Send email notifications for relevant events
- **File Upload**: Upload and store files (images, documents, receipts, etc.)
- **Authentication & Authorization**: User login/registration with JWT and access control
- **Search & Filtering**: Search functionality with filters and pagination

4.1. 🟢 Beginner-Friendly Project: Todo list

1. As a user, I want to create a task (CREATE)
   - Title, description
   - Priority (low, medium, high)
   - Due date
   - Category/tag
   - Optional: attach files (file upload - documents, images, etc.)
2. As a user, I want to view all my tasks (READ)
   - List all tasks with pagination
   - See task details
3. As a user, I want to search and filter tasks (Search & Filtering)

   - Search tasks by title
   - Filter by priority, category, status, or due date
4. As a user, I want to mark tasks as complete/incomplete (UPDATE)
   - Toggle task completion status
5. As a user, I want to edit or delete my tasks (UPDATE & DELETE)
6. As a user, I want to organize tasks into categories/tags (CREATE)
   - Create custom categories or tags
   - Assign tasks to categories
7. As a user, I want to view other users' todo lists (READ)
   - Browse and search for other users
   - View their public tasks (read-only access)
   - Cannot edit or delete other users' tasks
8. As a user, I want to receive email reminders (Email)
   - Email notification when a task deadline is approaching (1 day before due date)
   - Daily summary of pending tasks

4.2. 🔴 Advanced Project: E-commerce Dashboard

1. As a seller, I want to manage products (CREATE)
   - Product name, description, price
   - Category (e.g., Electronics, Fashion, Food)
   - Stock quantity
   - Upload product images (file upload - multiple images per product)
   - SKU (stock keeping unit) auto-generated
2. As a customer, I want to browse and search products (READ & Search & Filtering)
   - View all products with pagination
   - Search by product name or description
   - Filter by category, price range, availability
   - Sort by price, newest, popularity
3. As a customer, I want to add products to cart (CREATE)
   - Select product and quantity
   - View cart with total price
   - Update quantities or remove items from cart
4. As a customer, I want to place orders (CREATE)
   - Checkout from cart
   - Provide shipping address
   - Order confirmation with order number
   - Order status: pending, processing, shipped, delivered, cancelled
5. As a seller, I want to manage orders (UPDATE)
   - View all orders for my products
   - Update order status (mark as processing, shipped, delivered)
   - View order details (customer, products, quantities, total)
6. As a customer, I want to view my order history (READ)
   - List all my past orders with pagination
   - View order details and status
   - Filter by status or date range
7. As a seller, I want to view sales dashboard (READ)
   - Total revenue and order count (daily, weekly, monthly)
   - Sales trends over time (charts/graphs)
   - Top-selling products
   - Inventory alerts for low stock items
   - Order status breakdown (pending, shipped, etc.)

8. As a seller, I want to manage product inventory (UPDATE)
   - Update stock quantities
   - Mark products as out of stock
   - Receive alerts when stock is low (< 10 units)
9. As a user, I want to edit or delete my content (UPDATE & DELETE)
   - Sellers can edit/delete their products
   - Customers can cancel pending orders
10. As a user, I want to receive email notifications (Email)
    - Order confirmation email to customers
    - Order status updates to customers (shipped, delivered)
    - New order alerts to sellers
    - Low stock warnings to sellers
    - Weekly sales summary reports to sellers

4.3. 💡 Custom Project (Your Own Idea)

You may create your own project idea instead of choosing from the above options.

4.3.1. Requirements

1. Your project must implement all core technical requirements:
   - **CRUD RESTful API**: Full Create, Read, Update, Delete operations
   - **Email**: Email notifications for relevant events
   - **File Upload**: File upload and storage functionality
   - **Authentication & Authorization**: JWT-based auth with access control
   - **Search & Filtering**: Search with filters and pagination
2. Include at least 6 core features (user stories)
3. Document your project idea, features, and database schema in your project documentation

4.3.2. Project Ideas

Some examples to inspire you (but you can build anything):

- Social media platform with real-time messaging
- Project collaboration tool with file sharing
- Inventory management system with analytics
- Food delivery platform with order tracking
- Hotel/room booking system
- Job board with application tracking
- Music/video streaming platform
- Online quiz/exam platform
- Expense splitting app for groups
- Habit tracker with statistics

Note: Choose something personally meaningful or relevant to your interests. Your project will be evaluated based on its demonstrated complexity.