**Final Project guidelines**

**Project Requirements:**

1. **Responsive Design:**Your web application must be mobile-friendly and fully responsive.
2. **User Roles (Minimum 2):**
   1. Admin
   2. At least 3 User roles (based on your project use-case)
      1. Premium, Regular, ?? (折扣, 高級產品)
   3. Login pages required for all roles with bcrypt-based password encryption
3. **Pages:**Minimum of 6 to 8 pages (as per your project functionality)
   1. home, login, sign up, shopping cart, order history, admin (CRUD)
4. **Session Management & Data Consistency:**Ensure proper session handling and consistent data throughout
5. **End-to-End Transactions & Flows**
   1. Your project must demonstrate complete transactional flows based on your domain.
   2. A transactional flow refers to a series of connected actions that a user performs from start to finish.
   3. For example, in an e-commerce application (like Amazon):
      1. A user (customer) should be able to:  
         Login → Browse Products → Add Items to Cart → Checkout → Make Payment. This entire sequence counts as one complete flow.
   4. You are expected to implement a minimum of 3 such complete flows per project, which may vary based on the roles and nature of your application.
      1. User Sign up
      2. User buy products
      3. Admin manage user accounts
   5. Each role (Admin/User/etc.) should be part of at least one meaningful transaction or activity flow.
6. **Frontend Stack:**
7. **HTML, CSS, Bootstrap** (mandatory)
8. Any additional UI libraries allowed
9. **Framework**: React or Angular only
10. **Backend Stack:**
    1. Node.js + Express.js
11. **APIs:**
    1. Use of External APIs is encouraged where relevant
    2. Payment Gateway API is required for e-commerce-type projects
       1. Stripe api in node.js+express
12. **Hosting (Bonus Points):**

Deploy your app on a platform like Heroku, Render, or any server of your choice.

1. **Git & Canvas Submission:**
2. Upload on **both GitHub and Canvas**
3. Ensure **no major timestamp gap** between submissions
4. **CRUD Functionality:**Required if your project includes persistent data
5. **Architecture:**Project must follow REST or MVC architecture
6. **Git Best Practices:**
7. Use **branching**
8. Commit per functionality (not bulk uploads)
9. Include **README.md** and **.gitignore** (especially for node\_modules)
10. Follow proper **code documentation** and **naming conventions**
11. **Database:**
    1. Preferred: **MongoDB**
    2. **SQL** is allowed **only with strong justification**