Tech stacks (Next.js, React.js, Node.js, Express, MongoDB, Razorpay).   
Here’s a detailed plan:

**1. Single Product and Multi-Product Pages for Selling**

**Frontend (Next.js + React.js)**

* **Single Product Page:**
  + Create a dynamic route in Next.js (e.g., /product/[id]).
  + Use React components to display product details like images, description, pricing, reviews, and "Add to Cart" functionality.
  + Integrate Razorpay for the payment gateway.
* **Multi-Product Page:**
  + Create a products listing page (e.g., /products), which fetches a list of products from the backend using Next.js API routes.
  + Filter, sort, and pagination options for a smooth user experience.

**Backend (Node.js + Express + MongoDB)**

* **API for Products:**
  + Create APIs for CRUD operations on products (GET for listing products, POST for creating products, PUT for updating, DELETE for removing products).
  + MongoDB will store product data (title, description, images, price, category, etc.).
* **Payment Gateway Integration:**
  + Use Razorpay APIs to handle payment transactions.
  + Backend logic will generate payment links or requests and handle payment success/failure callbacks.

**Timeline Estimate:**

* **2-3 Weeks** for full functionality, including front-end integration, product listing, and payment gateway setup.

**2. Video Content Uploads and Connections for Video Courses**

**Frontend (Next.js + React.js)**

* **Video Upload Page:**
  + Create a form using React where users can upload video files (course material).
  + Use services like AWS S3, Firebase Storage, or any CDN to store the video files.
* **Video Course Page:**
  + Create a dynamic page that fetches and plays video courses using HTML5 video player or third-party libraries like Plyr or Video.js.

**Backend (Node.js + Express + MongoDB)**

* **API for Video Uploads:**
  + Create an API to handle video uploads and store metadata (title, description, URL) in MongoDB.
  + Use AWS SDK or Firebase Admin SDK for uploading videos to cloud storage.
* **Authentication & Authorization:**
  + Ensure only authorized users (course creators or admins) can upload videos.
  + Implement access control for viewing videos (e.g., only users who purchased the course).

**Timeline Estimate:**

* **2-3 Weeks** for setting up video uploads, storage, and integration with front-end.

**3. Event Creation**

**Frontend (Next.js + React.js)**

* **Event Creation Form:**
  + Allow users to create events with fields like event name, date, description, location (if physical), and video link (if online).
* **Event Listings and Registration:**
  + Display a list of upcoming events on the front-end.
  + Allow users to register for events and save their registration info.

**Backend (Node.js + Express + MongoDB)**

* **Event Management API:**
  + Create API endpoints to add, update, list, and delete events.
  + Store event details in MongoDB.
* **Notifications and Reminders:**
  + Integrate with email or SMS APIs (SendGrid, Twilio) to send reminders to registered users.

**Timeline Estimate:**

* **2 Weeks** for event creation, display, and user registration functionality.

**4. Custom Payment Page**

**Frontend (Next.js + React.js)**

* **Custom Checkout Pages:**
  + Design custom checkout flows using React and Razorpay payment buttons.
  + Include features like multiple payment methods (cards, UPI, wallets) and support for discount codes or offers.

**Backend (Node.js + Express + MongoDB)**

* **Payment API Integration:**
  + Use Razorpay’s API to create orders, handle payment verifications, and store transaction details.
  + Handle callbacks from Razorpay to update the status of payments (success, failure, pending).

**Timeline Estimate:**

* **1-2 Weeks** for integrating Razorpay and creating custom payment flows.

**5. Telegram Group Channel Creation**

**Frontend (Next.js + React.js)**

* **Telegram Integration:**
  + Provide a simple UI for users to join Telegram groups or channels after purchasing a product or course.
  + Use Telegram’s deep link or API to allow users to be added to specific channels.

**Backend (Node.js + Express)**

* **API for Group Management:**
  + Use Telegram Bot API or Telegram's HTTP API to manage group/channel memberships.
  + Once a user purchases a product, use the backend to automatically invite them to a specific group/channel.

**Timeline Estimate:**

* **1-2 Weeks** to handle Telegram group integration.

**Complete Stack Breakdown**

* **Frontend:**
  + **Next.js:** Framework for React with SSR (Server-Side Rendering) and static site generation.
  + **React.js:** Component-based frontend library for building dynamic user interfaces.
  + **Tailwind CSS/Material UI:** (Optional) for styling the frontend.
* **Backend:**
  + **Node.js + Express:** Server-side framework to handle API requests, file uploads, authentication, and payment handling.
  + **MongoDB:** Database to store product data, user details, events, and transaction information.
  + **Razorpay:** For handling all payment-related processes.
  + **AWS S3/Firebase Storage:** For storing video content (or any other cloud-based storage solution).

**Project Timeline Overview**

* **Planning & Architecture:** 1 week
* **Single/Multi-product pages + Payment Integration:** 2-3 weeks
* **Video content upload & course handling:** 2-3 weeks
* **Event creation & management:** 2 weeks
* **Custom payment page setup:** 1-2 weeks
* **Telegram group integration:** 1-2 weeks
* **Testing and QA:** 1-2 weeks

**Total Estimated Time:**

**9-13 weeks** depending on the complexity of features and thorough testing.

**Approach:**

1. **Phase 1 (Core Features):**
   * Set up the Next.js and React.js environment.
   * Build product pages (single and multi-product) and integrate payment with Razorpay.
   * Implement the backend for handling product data, payments, and basic authentication.
2. **Phase 2 (Video and Events):**
   * Add the video content upload system.
   * Build the event creation and management module.
3. **Phase 3 (Advanced Features):**
   * Customize payment pages for specific products or services.
   * Implement Telegram group/channel integration.
4. **Phase 4 (Testing and Deployment):**
   * Conduct extensive testing (unit tests, integration tests).
   * Deploy the application (consider platforms like Vercel for frontend and AWS/DigitalOcean for backend).