MERN Stack + DevOps + Cloud Computing + Cloud Deployment of Project

Topic	Hours	Module	Total
HTML / HTML5 – Semantic markup & standards-compliant UI structuring	4	UI / Frontend	28
CSS / CSS3 – Advanced styling, layout systems & responsive design	6		
Git / GitHub – Distributed version control & collaborative code management	**4**		
Javascript (Intermediate) - Core language fundamentals & execution model	8		
Project 1: Javascript application development using pre-integrated APIs	**6**		
NodeJS – Server-Side JavaScript runtime & event-driven architecture	8	Backend	48
Project 2: NodeJS based server-side rendering implementation	**8**		
ExpressJS Framework – MVC-Oriented backend architecture	8		
MongoDB / Mongoose – NoSQL database design with ODM abstractions	8		
Project 3: ExpressJS REST API development with authentication & authorization	**10**		
Backend Deployment – Cloud hosting using render & railway	4		
-- buffer / student bug fixing --	2		
React – Client-Side rendering & application state management	12	Frontend	28
Project 4: Frontend – Frontend application development with modern react paradigms	**12**		
Frontend Deployment & Testing – Production hosting via vercel & cloudflare	2		
-- buffer / student bug fixing --	2		
Frontend Advanced Concepts – Context API & redux-based state management	12	Full Stack	34
Project 5: Full Stack end-to-end full stack application development	**18**		
Production Deployment & Observability – Full stack integration & monitoring	2		
-- buffer / student bug fixing --	2		
Cloud Computing Fundamentals – AWS cloud, EC2, S3, Loadbalancers & Servers	**6**	Cloud	30
Cloud Computing Fundamentals – ECS, Lambda, Networking & Security essentials	6		
DevOps Basics & Deployment - GitHub workflows, CI/CD	8		
Devops - Jenkins, Docker, Kubernetes, EKS	**8**		
-- buffer / student bug fixing --	2		
Project 6: Full Stack application extension using AI / LLM APIs	**12**	Misc	24
NextJS – Full-Stack React Framework with hybrid rendering	8		
SQL Integration in MERN Apps – RDBMS in full stack	4		
	192		**192**

UI / Frontend

Lec No.	Topic	Pointers to Cover	Time (hr)
1	HTML Basics	- Tags - Semantics - Block elements - Inline elements	1
2	List, Tables	- Unordered list - Order list - Description list - Table - Table properties	1
3	HTML Forms	- Basic HTML Forms - Forms events - Submit URLs and query params	1
4	More Tags & Entities	- Create HTML landing page - Misceleneous tags - HTML Entities	1
5	Selectors, Box Model	- CSS Selectors, compounding - Box Model - Element dimensions, display property	1
6	Important concepts	- Cascading - Specificity - Inheritance - !important keyword - Text formatting	1
7	Units & Combinators	- CSS units - Combinators	1
8	Pseudo Selectors	- Pseudo selectors - Pseudo classes - Pseudo elements	1
9	Flex-box & Grid	- Flex box - Limitations of flex box - Grid - Flexbox froggy game	1
10	Media query & position property	- Standard screensizes - Design approach (mobile first) - Media query - Position property	1
11	First Repo	- Explain Git and GitHub seperately. - Git vs GitHub - Creating account - Creating repo on Github - Explain: git init git status git add <> git add . git pull git push (and push related errors and fixes)	1
12	Cloning the Repo (Students practice in pairs)	- Create new public repo & connect it to local folder - Add a html and css file and push it - Share the URL of your repo with your friend - Clone the github repo using `git clone` in terminal - Now, the cloner adds few more css styles and push it - The owner pulls the code and runs it to see changes	1

13	Using Git with VS Code	- Using Visual Studio Code UI to view 'diff'. Use the ui for git steps 'add', 'commit', 'pull' & 'push' - Ignore files using `.gitignore` - Understanding Commits, Hashes & HEAD	1
14	GitHub concepts	- Git Architecture: Working Tree, Staging Area - Branching Concepts & Use Cases - Branch Operations using commands like git branch git checkout git switch - Merging Strategies Fast-forward Three-way Merge - Merge Conflicts: Identification & Resolution - Rebasing: Concept, Pros & Cons	1
15	Language fundamentals	- Variables - Datatypes - Numbers - Strings - Template literals	1
16	Scope specifiers	- Var - Let - Const - Loose & strict equality - Loops - Basics of functions	1
17	Functions in JS	- Function declaration - Function assignment - Anonymous function assignment - Arrow functions - IIFE - Revision of scopes	1
18	Objects & arrays	- Objects - Arrays - Loops - Memory allocation	1
19	Events in JS	- Browser object model - Document object model - Events - Form Events	1
20	Advanced events	- Form events - Event bubbling and event capturing - Event delegation	1
21	Callbacks	- Definition of callbacks - Higher order functions - Array methods - setTimeout, setInterval	1
22	Promises	- Promise handling (fetch API) - Promises vs callbacks - Creating new promise	1
23	Project - Ecommerce Platform	- Create a new github repo and connect it to local folder - Landing page - API results fetching and rendering results (use dummyjson.com/products for it) - Dynamic rendering using create element, append child - Push the code to GitHub	1
24	Search Feature	- Implement search logic and redirecting use query params - Read query on search results page and render the search list - Add card interactions using CSS - Push the changes to GitHub	1
25	Save search history	- Use localStorage to save search history - Create a search suggestions box for search bar - Filter the search history suggestions as the user type in	1
26	Watch history tracking	- Store the user watch history array in the localStorage - Create a seperate page to show user's search history in reverse chronological order - Render the preview by redirecting to asset details page created earlier - Push the changes to GitHub	1
27	Pagination	- Brainstorm pagination logic. - Implement pagination on search page - Handle pagination edge cases - Push the changes to GitHub	1
28	GitHub Pages	- Create deployment file - Use github based routing - Access project publicly	1

Backend

Lec No.	Topic	Pointers to Cover	Time (hr)
1	Introduction	- Client Server Model - Frontend Server vs Backend Server - Latency, Response Time - NodeJS theory intro - NodeJS installation - Node REPL	1
2	NodeJS Modules	- Developer defined modules - Commonjs vs ESModules - export-require - indepth working of require function	1
3	Javascript runtime	- Blocking vs non-blocking - sync vs async - fs sync API - fs callback API - fs promise API - console.time vs performance.now	1
4	Blocking and non-blocking operations	- Handson on blocking vs non-blocking operations - using os modules - using crypto module (pbkdf2)	1
5	Event driven architecture	- Custom Events - HTTP server	1

6	Backend server	- HTTP server to send responses - text response - html response - json response - route based responses - REST API architecture	1	
7	Serving static webpages	- Send direct html template code - Limitations - code quality, repeatation, syntax highlighting - Save templates in seperate folder - Send based on route	1	
8	Serving dynamic webpages	- Keep the HTML templates in template folder with identifiers - Use inline / internal CSS - Replace the identifiers with data to hydrate HTML - Send based on route	1	
9	Creating a proxy on dummyjson.com/products	- Keep the HTML templates in template folder with identifiers - Get the data from dummyjson.com/products or dummyjson.com/recipes and hydrate the html - Send CSS files seperately - Send favicon	1	
10	Rendering same webpage using plain javascript	- Fetch the data from dummyjson.com/products or dummyjson.com/recipes and render it in the ui using plain html, css, js - Identify the relation between api response time and the cards loading time - Use slow 3G internet from developer tools	1	
11	Client side rendering vs Server side rendering	- Identify the relation between api response time and the cards loading time for CSR - Identify the relation between api response time and the cards loading time for SSR - Use slow 3G internet from developer tools - Discuss SEO impact - Discuss pros and cons of SSR and CSR	1	
12	Deployment using render	- Create account on render - Connect the repository and create render project - Deploy the backend and test it	1	
13	External modules	- Chalk - Figlet - ExpressJS - ExpressJS basic server - Route not found middleware - Status codes revision	1	
14	Routers and controllers	- Express app level routing - Express router - app.use(router) vs app.use('/api/users', router) - controllers - try catch for error handling	1	
15	GET and POST	- GET method - JSON response - POST method - Body parser - Content-type	1	
16	PATCH and DELETE	- PUT vs PATCH method - Status codes - Validation - DELETE method	1	
17	MVC architecture	- Model - Views / Discuss why views are ignored - Controllers - Services - DALs - DTOs - utils - constants - config	1	
18	Middlewares and DTOs	- App level middleware - Router level middleware - Middleware chaining - DTOs for valdiation	1	
19	Code refactoring	- Refactor old code to use MVC architecture - Split the code into DTOs, controllers and services - put the utility logic inside controllers in seperate files inside utility folder	1	
20	Zod validator	- Understand zod as DTO - Zod Basics - Validation Rules - Parsing & Error Handling	1	
21	Introduction to MongoDB	- Introduction to databases - Database management systems - SQL vs No-SQL - Discuss MongoDB vs relational databases - Database, Collection, Document, BSON vs JSON, _id field and ObjectId - ACID in MongoDB (modern versions) - Single-document atomicity	1	
22	MongoDB query and filtering (UI)	- Filters & operators ($eq, $gt, $in) - Projections - Sorting - Pagination (limit, skip)	1	
23	Introduction to Mongoose	- Configure database user - Configure network access - ODM vs MongoDB Driver - Schema, Model, Document - Data Types & Validations - Connection & Config - Strict Mode & Timestamps	1	
24	CRUD operations using Mongoose	- Create (create, save) - Read (find, findOne, findById) - Update (updateOne, findByIdAndUpdate) - Delete (deleteOne, findByIdAndDelete) - Error Handling	1	

25	Mongoose queries (Miscellaneous)	- Query Chaining - Operators ($gt, $in, $or) - Sorting, Limit, Skip - Population (populate) - Middleware Hooks	1	
26	MongoDB filters vs Mongoose queries	- Native MongoDB Filters - Mongoose Query Abstraction - Query Translation - Performance Tradeoffs - When to Use Which	1	
27	List API, search and pagination	- List API Structure - Pagination (limit, skip) - Cursor vs Page Pagination - Search (Regex / Text Index) - Indexing for Performance	1	
28	.env setup, deployment using render	- Mongoose setup to always apply validation on updates, return new document - Setup .env properly to not leak any secrets - Add .env to .gitignore - Push the code github and create new project for this repo on render - Setup environment variables in the project - Deploy the project	1	
29	Project: Content management system	- Create a new repository and connect it to local folder - MVC architecure folder structure and .env setup - Setup cors, body parser limit, morgan - User schema, OTP schema, Artifact schema - Test endpoint "/" - Push the code to GitHub - Mongoose setup to always apply validation on updates, return new document	1	
30	OTP verification	- Discuss security measures for user email access validation - OTP sending and verification - Measures to store OTPs properly - Bcrypt in depth (hashing algorithm, strategy, salt, rounds, password collision) - Mongoose pre-save middleware for otp schema	1	
31	User Signup	- After user email access validation, discuss security measures for password storing - Bcrypt revision - Mongoose pre-save middleware for user schema - Push the changes to GitHub	1	
32	User Login	- Password validation - Bcrypt in depth for password comparision - Hashing vs Encryption - Discuss ways to remeber user - How does token solve the validation problem - How to store token safely - JWT token - Tampering of JWT tokens - Push the code to GitHub	1	
33	Protected APIs (Authentication)	- Discuss security measures for storing token on frontend - Test login and sigup APIs using postman - Send token in cookies - Create and add auth middleware as app level middleware - Create POST api for artifact - Test protected API with token using postman - Push the code to GitHub	1	
34	Role based access controls (Authorization)	- Allow admin users to view all artifacts - Implement role based access control in auth middleware - Push the code to GitHub	1	
35	CRUD on artifacts	- Multer middleware to store file on server - use Cloudinary to store the artifacts - POST api to create artifact in DB - GET api to read artifact info - PATCH and DELETE on artifact - Push the code to GitHub	1	
36	Like and Comment API	- Discuss the Schema changes - Implement GET & POST for like & comment - Push the code to GitHub	1	
37	Security enhancements	- Rate limiting - Helmet middleware - XSS, CSRF basics	1	
38	Advanced Concepts	- Webhooks - CRON jobs - NGROK for development API testing	1	
39	Chatting Setup	- Chat Schema, Thread Schema - GET and POST api for Chats - Introduction to websockets - Websockets vs HTTP	1	
40	Live chat setup	- Implement basic websocket using socket.io - Use local map to store online users - Send realtime messages to online user	1	
41	Production level code	- Request logging - Error logging - HTTP status code standards - Idompotency - Pipe, streams	1	
42	Error handling middleware	- app.on method to handle runtime and process level unhandled errors - error handling middleware - how to pass errors using next() function - async handler architecture	1	
43	Deployment using Render	- Create account on railway - Connect the repository and create railway project - Deploy the backend and test it	1	
44	SMTP issue fix	- Use Resend SDK to fix the SMTP issue - Redploy and test OTP related APIs	1	
45	Deployment using Railway	- Connect the repository and create render project - Deploy the backend and test it	1	
46	Monitoring and logs	- Use the request logs and errors logs to debug production app - Understand the pain points in debugging backend apps - Load testing of the backend	1	
47	-- buffer / student bug fixing --		1	

48	-- buffer / student bug fixing --		1

Frontend

Lec No.	Topic	Pointers to Cover	Time (hr)
1	Using react as an extension to javascript's dynamic rendering	- Story behind DOM re-rendering optimization - React CDNs - React.createElement - ReactDOM.createRoot - React Virtual DOM - React Element - React Element Object representation - $$typeof as a security measure	1
2	Introducing JSX to reduce syntax burden	- JSX definition - Babel as transcompiler - Babel CDN to parse JSX - Functions analogy for React Component - React Component vs Element - Functional arguments analogy for props - Undertanding how components promote re-useability	1
3	Making UI using React Components	- Component nesting - forEach vs map - Component looping using map - passing dynamic props - using Dummy Data - React Keys for UI rendering - Arguments analogy for props passwed to component - Default values and error handling - Delayed destructuring vs on-the-fly destructuring	1
4	React App using Parcel (zero-config) bundler	- Understanding why bundlers are required in modern javascript tools - `npm init` and use npmjs and install react, react-dom and parcel - View package.json, pakage-lock.json, node_modules and create .gitignore - Commonjs modules vs es modules - Use ESModules in the code to import react - bundle the project using parcel bundler - Import / export custom components in react - Named export vs default export	1
5	Internal details React App	- Development build vs Production build - dist folder - css behaviour in react - Global stylesheet vs component specific stylesheets - Inline styling with style object - .module.css in react	1
6	React App using Vite Bundler	- Understanding boilerplate code - ESLint - .jsx extension vs .js extension - vite development vs production build - Styling revision - global stylsheet, component level stylsheets, inline, .module.css - Landing page design	1
7	React events and component states	(To add contact me form on the landing page) - Events in react (onKeyUp, onKeyDown, onChange, onClick, onMouseUp, ...) - Using those events to show alerts / console logs - Introduce variable using 'let' and then show how re-rendering is required - Introduce useState to manage the state - Explain component mounting / unmounting, scheduling, re-rendering	1
8	useState hook in React	(Explore react using contact me form on the landing page) - "One way data-binding". Edge cases that introduce bugs. - Use "value" attribute to avoid bugs. Thus, explain "two way data-binding". - Show updates-chaining bugs. Explain updates-chaining fixes. - Callback function in setter function.	1
9	useState in depth in React	(Add interactable items on the landing page) - Callback function for default state. useState(() => {...}); - State persistence using localStorage. Revise ternary operator. - Controlled components. - Uncontrolled components. - Differences and usecases of controlled vs uncontrolled components.	1
10	Managing Arrays and Objects	(To reduce state variables in the form, add dynamic list to page) - Storing Object as a state - Explain how react compares state using Object.is - Correct and incorrect ways to manipulate object state of react component - Shallow vs deep copy - Store "Array" as a state - Use methods like push and pop - Differentiate and emphasise on the array methods which return new array vs the array methods which manipulate original array. Final note: guide students to always use array methods which return new array to reduce the chances of introducing bugs.	1
11	useEffect hook in React	(Create a product listing page) - Explain component lifecycle methods - Introduce useEffect - Show useEffect working by implmenting useEffect in parent and child component. Understand the flow of trigger. - Do api call to dummyjson.com/products to get the data and show it on the page. - Explain .then / .catch to handle the api call. - Explain async / await and try / catch to handle the api call. - Track api call and component lifecycle using console logs. (Understand re-rendering) - An example usecase of useEffect with one dependency. - An example usecase of useEffect with two dependencies.	1
12	Mini Project - Stopwatch App	(Build a working app - Stopwatch app) - This exercise will let students reinforce whatever they have learnt till now. It will revise useState, useEffect, re-rederning, state management and complex UI logic. - Focus on multiple dependencies on useEffect - Focus on useEffect's cleaning function	1

13	Major Project - Media Platform (Real world Youtube Clone with media streaming capability - Videos and Images)	(Routing setup and dummy components) - Create a new GitHub repo and attach to the current folder - Introduce routing concepts - Use React router (v7) for routing setup - Differentiate between Link and Anchor tag - useNavigate to navigate dynamically using functions - useEffect cleanup function example - setup routing for the project and dummy components - Push the code to GitHub	1
14	Show trending videos RapidAPI platform Add Tailwind for styling	- Tailwind setup - create landing page - Show the API results (use RapidAPI platform for backend support - https://rapidapi.com/omarmhaimdat/api/youtube-v2/playground/apiendpoint_fd65e7da-b0cd-4740-9857-d74dc13d75b2) - Debug Youtube UI to see how Shimmer UI works. Add shimmer UI to landing page of the project - Use dummy data for intial making of the page, then replace it with live api to reduce the chances of hitting rate-limit - Explain layout shift in the UI and create a layout that does not do layout shift. - Push the changes to GitHub	1
15	Searching & Listing	- Add Search box on the landing page - Handle user search query - Navigate the user to search results page - Persist search query using query params - Render the result cards - Homework: add shimmer UI for search results - Push the changes to GitHub	1
16	View particular image / play video	- Create seperate page for playing a particular video / show a particular image - Use iframe for the video player - Persist the asset id in query param - Call an api to show the asset details - Link this page to search results and the trending videos - Push the changes to GitHub	1
17	Feature: Pagination	- Brainstorm to create pagination component. Discuss edge cases. - Create a pagination component with props and state. - Import pagination component on search page - Handle pagination persist using query params - Push the changes to GitHub	1
18	Feature: Project structuring	- This lecture gives breathing time to students and allow mentor to debug the issues that students are facing. - Check every student's project for naming conventions, code writing style and completion status - Allow students to explore ways to add more features - Push the changes to GitHub	1
19	Feature: Search history tracking	- Store the search in the localStorage - Brainstorm for suggestion box component - Create suggestion box component using props and state - Integrate suggestion box in the search box on the home page. Pass the searchHistory as a prop. - Integrate suggestion box in the search box on the search results page. Pass the searchHistory as a prop. - Homework: Filter out the suggestions as the user type - Push the changes to GitHub	1
20	Feature: Watch history tracking	- Store the user watch history array in the localStorage - Create a seperate page to show user's search history in reverse chronological order - Render the preview by redirecting to asset details page created earlier - Push the changes to GitHub	1
21	Industry level code: Custom Hooks	- Explain how custom hooks are used in react. - Refactor the code to create custom hook. - Start with the custom hook for view trending videos API. - Use that hook on the homepage. - Create a custom hook for get video details API. - Push the changes to GitHub	1
22	Industry level code: Constants and utils	- Create a constants file to store all the constats in the app - Create a utils folder to store all the utilities - Create and use localStorageUtil to store the logic for storing and retrieving data from localStorage - Create and use searchHistoryUtil, watchHistoryUtil build on top of localStorageUtil - Push the changes to GitHub	1
23	Industry level code: .env, config	- Create a .env file to store the environment variables - Note: students should understand that .env in the frontend is public and putting any confidential information in it is risky. - Note: students should not put .env in the .gitignore - Create a config.js file to use environment variables and provide default values - Push the changes to GitHub (Avoid putting .env in the .gitignore)	1
24	Run project to final preview	- Run the project using npm run build. - Fix the build issues / errors / warnings. - Preview the development build locally. - Test for any bugs or issues. - Push the code to github	1
25	Deployment on Vercel	- Create account on vercel - Attach the github account - View public repos and connect to the project repo - Setup the project - Preview the build - Homework: Interested students should arrange credit card to buy domain tomorrow	1
26	Domain Connection	- Create account on spaceship / godaddy / hostinger - Buy a cheap domain - Explore DNS settings on different platforms - Follow steps mentioned on vercel to connect to purchased domain - Create a different project and connect different repo - Create a subdomain on the purchased domain and connect it to this project	1
27	-- buffer / student bug fixing --		1

28	-- buffer / student bug fixing --		1

Full Stack

Lec No.	Topic	Pointers to Cover	Time (hr)
1	State management using Context API	- Checkout new branch "development" - Explain prop drilling - Use context API from scratch to share basic states - Share this context to whole app - Push the changes to "development" branch	1
2	Implement code for Context API	- Import useContext to use the shared states - Use context states in shared components to replace the props - Refactor code to reduce prop drilling - Push the changes to "development" branch	1
3	Production level Context API	- Create custom hook to encapsulate context API and useContext - Create custom component to provide the context provider along with values - Refactor code to use the simplified version - Push the changes to "development" branch	1
4	Limitiation & Cons of Context API	- Unnecessary re-rendering on any state change - Only put centralized states in central level context API - Create other context to encapsulate search - This will showcase use of multiple contexts in a single app - Push the changes to "development" branch	1
5	Use Axios Instance	- Explain axios library - Import axios and create a instance - Use the instance in all the hooks to centralize API calls - Refactor code to remove manual json conversion - Push the changes to "development" branch	1
6	Toasts for success and errors	- Import toast library and create a util to use it - Have utilities for info, success and error toasts - Use error toast util to handle catch behaviour in the hooks as well as validations - Use success toast in the appropriate api hooks where required - Add info toast before starting the api call - Push the changes to "development" branch	1
7	Pending states for API hooks	- Add pending states for API calls in try block and settle it in finally block - Use these pending states to disable buttons, inputs and show loaders / shimmers - Push the changes to "development" branch	1
8	Redeploy to verify changes on Production	- Creating a staging environment using "development" branch - Deploy the project to staging and see the changes - Merge the code with Main branch - Students should showcase their project with all the changes re-deployed	1
9	Redux	- Revisit pros and cons of context api - Explain Redux - Understand redux architecture	1
10	Redux toolkit	- Install redux toolkit - Create a store and slice - Integrate slice to replace the responisibility of app context - Refactor code to derive states from redux and dispatch the actions instead of context api states - Push the changes to "development" branch	1
11	Use Redux for shared states	- Use redux slices for searching component - Refactor code to use redux states - Push the changes to "development" branch	1
12	Client Side Rendering Single Page Application	- Explore how code is shared from server to client - Understand how browser processed html, css and javascript - Explain how SPA is different from normal web applications - Measures to improve SEO score for React Apps	1
13	Frontend Backend Connection	- Understand CORS issue - Configure .env in frontend and backend - Understand debugging flow - Create Login and Signup Form on frontend - Debugging network requests (DevTools)	1
14	Integrate Authentication & Authorization	- Protected routes in frontend - API call to initialize app state - Integrate Login and Signup Forms with respective APIs - Store the tokens in cookies with http only true, Frontend cannot access them.	1
15	Frontend styling	- Use tailwind css to make the UI better - Create custom components for buttons with optional variations - Add styling for loading state, disabled state and hover state - Fix the chat pop-up widget to bottom-right corner	1
16	Integrate List API from backend	- Implement list API to search assets - Add hard filters in the frontend and pass it as query to the API - Show the results on the search page	1
17	Pagination	- Pass the results from List API to pagination component - Handle the values to show page options and navigation buttons - On change, trigger the page change and refetch List API with different page number	1
18	Asset Details API	- For the asset details page, integrate asset details API - Depending on the asset type, show video component or image component respectively	1
19	Search history tracking	- Create API to store user specific search history on backend - Integrate it on frontend	1
20	Watch history tracking	- Create API to store user specific watch history on backend - Integrate it on frontend	1
21	Payment Flow	- Add a validation of token count for chat page on frontend - Add a validation of token count for all chat APIs on backend as a middleware - Create payment details page on frontend	1
22	Transaction	- Create payment / orders schema on backend - Create Order API in backend - Add "transaction" property to order creation and token assignment	1

23	Integrate Payments	- Create account of cashfree - Integrate cashfree sdk on frontend and backend as well for payment verification - Test the payments to buy tokens for chatting - Add token decrease logic on every chat	1
24	Test payments	- Identify and fix the edge cases and bugs in the current flow - Integrate GET and POST chat APIs in the Chat UI on the frontend	1
25	Create New Asset	- Create a page to upload new asset on the backend - Integrate GET asset details API to view user specific assets - Modify schema to have the asset as "public" or "private" for search and public visibility	1
26	Likes and comments	- Integrate GET Likes API - Integrate GET Comments API - Integrate POST Likes API - Integrate POST Comments API	1
27	Integrate Live Chatting	- Implement websocket connection - Handle message sent logic on the frontend - Handle message sent logic on the backend and notify other user if online	1
28	Websocket debugging in frontend	- Use developer tools to debug webscokets on the frontend - Understand events fired and the payload recieved during the socket push - Handle mark user as online (backend) on socket connection - Handle mark user as offline (backend) on socket connection - Backend will fire event to all sockets for every connection / disconnection	1
29	Live chat handling	- Handle message recieved logic on the frontend - Re-render the Ui to show the message and a notification badge for new added message	1
30	Unread Messages	- Create unread count API on backend - Integrate unread count API on frontend - Create mark all read API on backend - Integrate mark all read API on frontend	1
31	Deploy the frontend using Vercel	- Login to vercel account - View public repos and connect to the project repo - Setup the project - Preview the build	1
32	Deploy the backend using Railway	- Since the backend already is connected to a railway project, just redeploy the backend to view the latest changes - Use the backend url from railway in the environment variables of the frontend on vercel - Connect the custom domain to frontend vercel project - Add this domain as CORS accessor in backend .env	1
33	-- buffer / student bug fixing --		1
34	-- buffer / student bug fixing --		1

Cloud

Lec No.	Topic	Pointers to Cover	Time (hr)
1	Cloud Computing & AWS Basics	- What is Cloud Computing? - IaaS, PaaS, SaaS - AWS global infrastructure (Regions, AZs) - Shared Responsibility Model	1
2	EC2 Fundamentals	- What is EC2 - Instance types & use cases - AMIs - Key pairs & security groups - Launching an EC2 instance	1
3	EC2 Hands-on (Deploy Code from GitHub)	- SSH into EC2 - Installing runtime (Node / Python) - Cloning GitHub repo - Installing dependencies - Running application - Opening required ports	1
4	Storage & S3 Basics	- Object storage concepts - Buckets & objects - Storage classes - Bucket policies & permissions - Versioning	1
5	Load Balancers & High Availability	- Why load balancers - ALB vs NLB - Target groups - Health checks - High availability concepts	1
6	Servers, DNS & Domain Mapping	- Public IP vs Elastic IP - What is DNS - A records - Connecting domain (GoDaddy / Hostinger) - DNS propagation & common issues	1
7	Containers & ECS Fundamentals	- Containers vs VMs - ECS overview - Clusters, tasks, services - ECS use cases	1
8	ECS Deployment & Scaling	- Task definitions - Fargate vs EC2 - Service scaling - Load balancer integration	1
9	Serverless with AWS Lambda	- What is serverless - Lambda execution model - Triggers (API Gateway, S3) - Cold starts (concept)	1
10	Networking Fundamentals	- VPC basics - Public vs private subnets - Internet gateway - NAT gateway - Route tables	1

11	Security Essentials	- IAM users, roles, policies - Security groups vs NACLs - Least privilege - Secrets & credentials	1
12	Cloud Architecture & Cost Awareness	- Secure architecture patterns - Monitoring basics (CloudWatch) - Cost optimization basics - Common beginner mistakes	1
13	DevOps Fundamentals	- What is DevOps - CI vs CD - DevOps lifecycle - Automation mindset	1
14	Git & GitHub Workflow	- Git basics - Branching strategies - Pull requests - Code reviews	1
15	GitHub Actions Basics	- GitHub Actions overview - Workflow structure - Jobs & steps - Runners	1
16	CI Pipeline Design	- Build pipelines - Running tests - Linting & checks - Artifacts	1
17	CD Pipelines & Environments	- Deployment strategies - Environment separation - Secrets management - Rollbacks	1
18	Manual Deployment to Automation	- Manual EC2 deployment recap - SSH-based deployments - Restarting services - Zero downtime concept	1
19	CI/CD to EC2 (Hands-on)	- GitHub Actions → EC2 flow - SSH keys - Secrets in GitHub - Auto deployment - Service restart	1
20	Monitoring & Debugging Pipelines	- Logs & debugging - Failure recovery - Notifications - Best practices	1
21	Docker Fundamentals	- Docker overview - Images vs containers - Docker architecture - CLI basics	1
22	Docker Deep Dive	- Dockerfile - Image layers - Volumes - Networking basics	1
23	Jenkins Fundamentals	- Jenkins overview - Architecture - Jobs vs pipelines - Jenkinsfile intro	1
24	Jenkins CI/CD Pipelines	- Declarative pipelines - Stages & steps - GitHub integration - Best practices	1
25	Kubernetes Fundamentals	- Why Kubernetes - Cluster architecture - Pods, nodes, services - kubectl basics	1
26	Kubernetes Core Objects	- Deployments - ReplicaSets - ConfigMaps - Secrets - Scaling & rolling updates	1
27	AWS EKS	- What is EKS - EKS architecture - Deploying apps - Load balancer integration	1
28	Production DevOps Architecture	- CI/CD + Docker + K8s flow - Observability basics - Security best practices - Real-world architecture	1
29	-- buffer / student bug fixing --		1
30	-- buffer / student bug fixing --		1

Misc

Lec No.	Topic	Pointers to Cover	Time (hr)
1	Introduction to AI & LLMs for Developers	- AI vs ML vs DL - What are LLMs - Capabilities and limitations - Real-world use cases - Where LLMs fit in full stack apps	1
2	LLM APIs Overview	- LLM API concept - Request-response flow - Tokens and context - Latency considerations - Pricing basics	1
3	Prompt Engineering Basics	- What is a prompt - System vs user prompts - Prompt structure - Prompt clarity - Deterministic vs creative prompts	1
4	Advanced Prompting Techniques	- Few-shot prompting - Role prompting - Chain-of-thought concept - Prompt iteration - Common prompt failures	1

5	Backend Integration with LLM APIs	- Calling LLM APIs from backend - API key management - Environment variables - Request validation - Error handling	1	
6	LLM-powered Features Design	- Text generation - Summarization - Classification - Extraction - Content moderation - Autocomplete use cases	1	
7	Frontend Integration of AI Features	- Triggering AI requests from UI - Loading states - Streaming responses concept - Handling partial responses	1	
8	Context Management & Memory	- Context window limits - Managing conversation history - Truncation strategies - User-specific context	1	
9	Data Validation & Guardrails	- Input sanitization - Output validation - Schema-based outputs - Preventing hallucinations - Fallback logic	1	
10	Performance & Cost Optimization	- Token optimization - Caching AI responses - Batching requests - Handling retries - Timeout strategies	1	
11	Security & Privacy with AI	- PII handling - Prompt injection risks - Rate limiting - Abuse prevention - Compliance basics	1	
12	AI Feature in Production	- End-to-end AI feature flow - Monitoring AI failures - Logging prompts and responses - Real-world case discussion	1	
13	Introduction to Next.js	- What is Next.js - CSR vs SSR vs SSG - Why Next.js over CRA - Next.js architecture overview	1	
14	Routing & Navigation	- File-based routing - Dynamic routes - Nested routes - Layouts - Navigation using Link and router	1	
15	Rendering Strategies	- Client-side rendering - Server-side rendering - Static site generation - Incremental static regeneration - Tradeoffs	1	
16	Data Fetching in Next.js	- Fetching on server vs client - Fetch API usage - Caching strategies - Revalidation concept	1	
17	Backend in Next.js	- API routes - Route handlers - Server actions concept - Request handling - Backend vs frontend boundary	1	
18	Authentication & Authorization	- Session vs token auth - Protected routes - Middleware usage - Auth flow design	1	
19	Performance & Optimization	- Image optimization - Font optimization - Code splitting - Bundle size analysis - SEO basics	1	
20	Deployment & Production	- Environment variables - Build process - Deployment platforms - Monitoring basics - Common production issues	1	
21	SQL & RDBMS Fundamentals	- What is RDBMS - Tables and rows - Primary keys - Foreign keys - Normalization basics	1	
22	Core SQL Queries	- SELECT, INSERT, UPDATE, DELETE - WHERE clause, ORDER BY, LIMIT	1	
23	Relationships & Joins	- One-to-one - One-to-many - Joins (INNER, LEFT) - Indexing basics - Query performance awareness	1	
24	SQL in Full Stack Apps	- Connecting backend to SQL database - ORM basics - Drizzle ORM - Migrations concept - MongoDB vs SQL tradeoffs	1	

Evaluation Rubriks
Biweekly Test (Combination of MCQs and Coding Questions)
Project Review
Quaterly Mock Interviews (Ratings across Communication, Project Knowledge, Orientation)