Web Development Final Year Project Ideas PDF

A good web development final year project is a full stack app with login and roles, a real database, at least one complex feature such as payments, real-time chat, search or file uploads, and is deployed online. A learning management system, hostel management portal, job portal and e-commerce site with test payments are dependable choices.

Key takeaways

- Final-year web projects need roles, a database and one hard feature, not only pages.
- MERN, Next.js with PostgreSQL, or Django are all accepted stacks.
- Deploy it: a live URL is the best viva demo.
- Use test modes of payment gateways; never handle real card data.
- The PDF gives plans and steps, not finished source code.

Who this PDF is for

This PDF is for final-year B.Tech, BCA and MCA students who want a full stack web project. Each plan lists the main users and roles, the stack, modules a team of 3 or 4 can split (front end, back end, database, deployment and testing), and what examiners check. Stacks are flexible: use React or Next.js with Node.js, or Django or Spring Boot if your course teaches those.

How to choose your project

- **Real users help.** A project your hostel, club or department can use is easier to explain.
- **List roles first**: student, admin, teacher, seller.
- **Pick one hard feature** and do it well.
- **Set up Git from day one** with branches for each member.

Standard level projects

1. Learning Management System

- **Problem:** Teachers upload courses and quizzes; students learn and track progress.
- **Data and services:** Your own course content; video hosting on a free service.
- **Tech stack:** React, Node.js, Express, MongoDB, JWT, Cloudinary or S3.
- **Main modules:** Auth and roles, course builder, quiz engine, progress tracking, certificates.
- **Steps:** Design the schema, build auth, add course and quiz APIs, build student and teacher dashboards.

- **Show in the viva:** Role-based access checks on the server, not just the UI.
- **Extensions:** Add live classes with a video SDK.

2. Hostel Management Portal

- **Problem:** Manage room allotment, fees, complaints and visitors.
- **Data and services:** Sample student and room data you create.
- **Tech stack:** Django, PostgreSQL, Bootstrap or Tailwind.
- **Main modules:** Room allotment, fee tracking, complaint tickets, visitor log, reports.
- **Steps:** Model rooms and students, build allotment rules, add complaint workflow and reports.
- **Show in the viva:** How allotment rules prevent double booking.
- **Extensions:** Add QR-based visitor passes.

3. Job Portal with Resume Upload

- **Problem:** Connect recruiters and job seekers.
- **Data and services:** Sample jobs and resumes you create.
- **Tech stack:** Next.js, PostgreSQL, Prisma, file storage.
- **Main modules:** Profiles, job posting, search and filters, applications, recruiter dashboard.
- **Steps:** Build auth, job CRUD, search with filters and pagination, application tracking.
- **Show in the viva:** Database indexes for search speed.
- **Extensions:** Add AI-based resume matching.

4. Event Booking System

- **Problem:** Book seats for college events with QR tickets.
- **Data and services:** Sample events you create.
- **Tech stack:** React, Node.js, MySQL, Razorpay or Stripe in test mode.
- **Main modules:** Event listing, seat selection, payment, QR ticket, check-in scanner.
- **Steps:** Build listings, lock seats during payment, confirm via webhook, email a QR ticket.
- **Show in the viva:** How you stop two users booking the same seat.
- **Extensions:** Add waitlists.

Advanced level projects

5. Multi-vendor E-commerce Marketplace

- **Problem:** Sellers list products; buyers order and pay.
- **Data and services:** Sample product data, or a public product API for testing.
- **Tech stack:** MERN or Next.js, Redis, Stripe or Razorpay test mode, Docker.
- **Main modules:** Seller onboarding, catalogue, cart, checkout, orders, reviews, admin panel.
- **Steps:** Design the order flow, add payments via webhooks, cache product pages, containerise.
- **Show in the viva:** The order state machine and webhook handling.
- **Extensions:** Add search with Elasticsearch or Meilisearch.

6. Real-time Collaborative Whiteboard

- **Problem:** Several users draw and chat on the same board.

- **Data and services:** No external data.
- **Tech stack:** React, Node.js, Socket.IO, MongoDB, HTML canvas.
- **Main modules:** Rooms, drawing sync, chat, board save and load, permissions.
- **Steps:** Broadcast draw events through Socket.IO, handle late joiners, save board state.
- **Show in the viva:** How events are synced and conflicts handled.
- **Extensions:** Add shapes and undo across users.

7. Hospital Appointment and Records System

- **Problem:** Book doctor appointments and keep visit records.
- **Data and services:** Synthetic patient data you generate (no real patient data).
- **Tech stack:** Spring Boot or Django, PostgreSQL, React.
- **Main modules:** Doctor schedules, booking, prescriptions, record access log, admin.
- **Steps:** Model schedules and slots, build booking with conflicts check, log every record access.
- **Show in the viva:** Why access logs and roles matter for health data.
- **Extensions:** Add SMS or email reminders.

8. Online Code Judge

- **Problem:** Students submit code that runs against test cases.
- **Data and services:** Problems and test cases you write.
- **Tech stack:** React, Node.js, Docker containers for code runs, PostgreSQL, a job queue.
- **Main modules:** Problem bank, editor, sandboxed runner, verdicts, leaderboard.
- **Steps:** Run each submission in a limited Docker container with time and memory limits, compare outputs.
- **Show in the viva:** Sandboxing and why it is needed.
- **Extensions:** Add plagiarism checks.

Research-oriented projects

9. Web Performance Optimisation Study

- **Problem:** Measure and improve a site's load speed.
- **Data and services:** One of your own apps.
- **Tech stack:** Lighthouse, WebPageTest, Next.js or React.
- **Main modules:** Baseline audit, image and code splitting fixes, caching, re-audit.
- **Steps:** Record Core Web Vitals, apply one fix at a time, measure again.
- **Show in the viva:** Before and after numbers for each fix.
- **Extensions:** Add a CDN and compare.

10. Web Accessibility Checker

- **Problem:** Scan pages for accessibility issues and suggest fixes.
- **Data and services:** Public pages you test, and your own app.
- **Tech stack:** Node.js, Puppeteer or Playwright, axe-core.
- **Main modules:** Crawler, rule engine, report, fix suggestions.
- **Steps:** Crawl pages, run axe-core, group issues by WCAG rule, export reports.

- **Show in the viva:** Three common issues and their fixes.
- **Extensions:** Add colour contrast suggestions.

11. Microservices vs Monolith Comparison

- **Problem:** Build the same app both ways and compare.
- **Data and services:** A small shop or booking app.
- **Tech stack:** Node.js or Spring Boot, Docker Compose, a load-testing tool such as k6.
- **Main modules:** Monolith version, three-service version, load tests, results.
- **Steps:** Build both, run load tests, compare latency, errors and effort.
- **Show in the viva:** The trade-off table.
- **Extensions:** Add Kubernetes deployment.

12. Progressive Web App for Offline Use

- **Problem:** A notes or attendance app that works without internet.
- **Data and services:** No external data.
- **Tech stack:** React or vanilla JavaScript, service workers, IndexedDB.
- **Main modules:** Offline cache, local storage, sync on reconnect, conflict handling.
- **Steps:** Cache the app shell, store edits locally, sync when online, resolve conflicts.
- **Show in the viva:** Service worker lifecycle.
- **Extensions:** Add push notifications.

Report structure

A final-year report is marked on clarity as much as on code. A structure most universities accept:

1. **Abstract** (150 to 250 words): problem, method, main result.
2. **Introduction:** background, problem statement, objectives and scope.
3. **Literature review:** 8 to 15 papers or existing systems, and the gap your project fills.
4. **Requirements and design:** functional and non-functional requirements, architecture diagram, the ER diagram, use case diagram, API list and page flow.
5. **Implementation:** modules, tools, key algorithms and screenshots.
6. **Testing and results:** test cases with pass/fail, screenshots of key screens, performance numbers and the live URL.
7. **Conclusion and future scope.**
8. **References** in the format your department asks for (IEEE style is common).

Follow your own university's template if it gives one; it overrides this list.

Common mistakes to avoid

- Checking user roles only in the front end.
- Storing passwords without hashing.
- Pushing secrets and .env files to GitHub.
- Not deploying, so the demo depends on one laptop.

Frequently asked questions

Which web development project is best for a final year?

A full stack app with roles, a database and one complex feature such as payments, real-time updates or a code runner. LMS, job portal and e-commerce marketplace are common, safe choices.

MERN or Django for a final year project?

Either is fine. Use the stack your team knows best. Examiners care more about design, security and a working demo.

Should I deploy my project?

Yes. A live URL shows the project works outside your laptop. Many platforms offer free tiers for student apps.

Can I use templates for the front end?

You can use a UI library, but the core features, database and APIs should be your own work.

Does this PDF include full source code?

No. It gives the problem, data, stack, modules and steps for each project. You write the code yourself, which is what examiners test in the viva.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- MDN Web Docs
- React documentation
- OWASP Top 10
- web.dev: Core Web Vitals

About this PDF

Prepared by the Uncodemy training team as practice project ideas. Datasets and tools named are public; check each dataset's licence before use. We update it every January. Corrections: info@uncodemy.com.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550