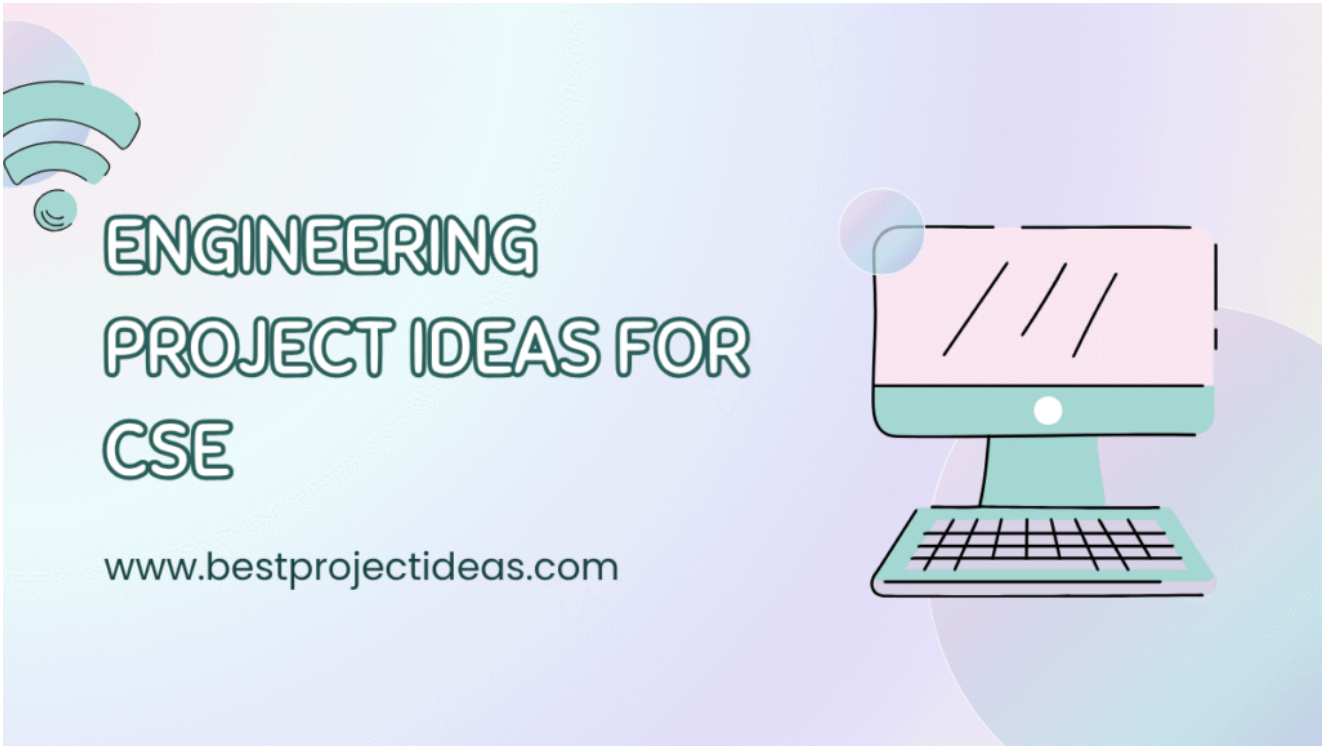
Best Project Ideas

46+ Engineering Project Ideas for CSE 2026

NOVEMBER 3, 2025 | JOHN DEAR



This article is written for computer science engineering (CSE) students who want practical, grade-worthy projects that teach important concepts and look good on resumes.

Below you'll find a clear introduction, tips for choosing a project, and 50 well-explained project ideas.

Each project includes: a short description, suggested technologies/tools, main features to implement, difficulty level, and a brief example or implementation note. You can copy-paste any section directly.

Table of Contents

Introduction

As a CSE student, building projects is the fastest way to turn theory into real skills. Projects help you understand algorithms, data structures, system design, networking, databases, security, and full-stack development. Good projects also show recruiters that you can plan, implement, test, and present a working solution.

This article lists **50 engineering project ideas for CSE** across different domains: web, mobile, AI/ML, systems, networking, security, databases, and DevOps. Each idea is practical, scalable, and adaptable — you can keep the core scope for a semester or extend it into a bigger capstone. For each idea, I provide technologies, features, difficulty, and an example implementation approach so you can get started immediately.

Also Read: 99+ Famous Profile Sekai Project Ideas For Students

How to choose the right project

1. **Match your course requirements.** If your program requires a report, choose a project with measurable experiments.
2. **Balance scope and time.** Start small: a minimum viable product (MVP) first, then add features.
3. **Focus on learning outcomes.** Pick projects that teach skills you want (e.g., ML, system design, security).
4. **Reuse libraries and APIs.** Use existing tools (frameworks, pre-trained models) to save time.
5. **Document everything.** Write a plan, architecture diagram, test cases, and usage instructions.

50 Engineering Project Ideas for CSE 2026

1. Student Management System (Web)

Description: A full-stack web application to manage student data — registration, attendance, grades, reports.
Technologies: React or Angular, Node.js/Express, MongoDB/Postgres, JWT authentication.
Main features: CRUD for students/courses, role-based access (admin/teacher/student), attendance tracker, gradebook, CSV import/export.
Difficulty: Beginner → Intermediate.
Example: Implement attendance marking with date-wise records and generate PDF mark sheets.

2. Online Examination Portal

Description: A secure online test platform with timed exams and automated evaluation.
Technologies: Django/Flask or Spring Boot, Vue/React, PostgreSQL, WebSocket for live timer.
Main features: Question bank, multiple question types, proctoring options (basic), timer, auto-grading and analytics.
Difficulty: Intermediate.
Example: Add an algorithm to randomize question order and prevent reattempts after submission.

3. E-commerce Website (Minimal)

Description: A shopping website with product catalog, cart, orders, and payments (sandbox).
Technologies: MERN stack (MongoDB, Express, React, Node), Stripe/PayPal sandbox.
Main features: Product listing, search and filter, shopping cart, checkout, order history, admin dashboard.
Difficulty: Intermediate.
Example: Implement search autocomplete and a recommendation based on user history.

4. Chat Application with Real-time Messaging

Description: A real-time chat app supporting one-to-one and group chats.
Technologies: Socket.IO with Node.js, React Native for mobile or React for web, Redis for pub/sub.
Main features: Message persistence, typing indicators, read receipts, media sharing.
Difficulty: Intermediate.
Example: Implement offline message sync and message delivery acknowledgment.

5. Library Management System (Desktop/Web)

Description: Manage book catalog, issue/return, fines, and inventory.
Technologies: JavaFX or .NET for desktop, or web stack with Django and PostgreSQL.
Main features: Barcode scanning, book reservation, member management, reporting.
Difficulty: Beginner → Intermediate.
Example: Auto-calculate late fees and email reminders for due dates.

6. Personal Finance Tracker

Description: Help users track income, expenses, budgets, and create visual reports.
Technologies: React Native or Flutter, Firebase or SQLite, Chart.js or D3.js.
Main features: Transaction categories, budget alerts, monthly graphs, CSV export.
Difficulty: Beginner.
Example: Add a predictive monthly expense estimator using moving averages.

7. Traffic Sign Recognition (Computer Vision)

Description: Detect and classify traffic signs from images/video (useful for autonomous vehicles).
Technologies: Python, OpenCV, TensorFlow/Keras, pre-trained CNN models (transfer learning).
Main features: Image preprocessing, model training, real-time detection with webcam feed.
Difficulty: Intermediate → Advanced.

Example: Fine-tune a MobileNet model on a traffic sign dataset and run live detection.

8. Handwritten Digit Recognition (ML)

Description: Classic MNIST classification problem extended with a GUI.
Technologies: Python, TensorFlow/PyTorch, Flask for web UI or Tkinter for desktop.
Main features: Training pipeline, accuracy visualization, live digit drawing and prediction.
Difficulty: Beginner.
Example: Build a web interface where users draw a digit and the model predicts the number.

9. Gesture-controlled Media Player

Description: Control playback (play/pause/next) with hand gestures via webcam.
Technologies: OpenCV, MediaPlayer libs, pretrained hand keypoint detectors (MediaPipe).
Main features: Gesture recognition, mapping gestures to controls, calibration screen.
Difficulty: Intermediate.
Example: Use MediaPipe to detect open/closed palm and toggle play/pause.

10. Social Media Sentiment Analyzer

Description: Analyze sentiments of tweets or posts about a topic and present analytics.
Technologies: Python, Tweepy (or social API), NLTK/spaCy, scikit-learn or transformers, Dash/Streamlit.
Main features: Data collection, sentiment classification, trending topics, charts.
Difficulty: Intermediate.
Example: Collect tweets about a new phone and show sentiment over time with word clouds.

11. URL Shortener with Analytics

Description: A service that shortens URLs and tracks click stats and sources.
Technologies: Node.js/Express, Redis for caching, MongoDB/Postgres, Nginx.
Main features: URL creation, redirection, click counters, location/time analytics, custom aliases.
Difficulty: Beginner → Intermediate.
Example: Provide an admin page showing top referrers and device types.

12. Plagiarism Detection System

Description: Compare documents for similarity using text preprocessing and algorithms.
Technologies: Python, TF-IDF, cosine similarity, shingling (n-gram), Flask/React for UI.
Main features: File upload, similarity percentage, highlighted matching passages, report generation.
Difficulty: Intermediate.
Example: Use 3-gram shingles and compute cosine similarity between documents.

13. IoT-based Smart Home Controller

Description: Control lights, fans, and sensors via a web/mobile app connecting to IoT devices.
Technologies: Raspberry Pi/Arduino, MQTT, Firebase/Node-RED, React Native.
Main features: Device discovery, scheduling, sensor data charts (temperature/humidity).
Difficulty: Intermediate → Advanced.
Example: Use MQTT broker to toggle a relay connected to a Raspberry Pi GPIO pin.

14. Resume Builder with Templates

Description: An online resume builder with multiple templates, PDF export, and tips.
Technologies: React, Node.js, jsPDF or Puppeteer for PDF generation, MongoDB for storage.
Main features: Template selection, drag-and-drop sections, export to PDF, profile sharing link.
Difficulty: Beginner.

Example: Implement live preview of the resume and export as a downloadable PDF.

15. Expense Splitter App (Group Expense)

Description: Split bills among friends, track balances, and simplify settlements.
Technologies: Flutter for cross-platform mobile, Firebase Auth and Firestore.
Main features: Create groups, add expenses, balances per person, push notifications.
Difficulty: Beginner → Intermediate.
Example: Show net owing for each member and propose minimal transactions to settle debts.

16. Image Caption Generator

Description: Generate natural-language captions for images using encoder-decoder models.
Technologies: Python, TensorFlow/Keras, CNN (encoder) + LSTM/Transformer (decoder).
Main features: Image preprocessing, beam search for caption generation, evaluation metrics (BLEU).
Difficulty: Advanced.
Example: Use InceptionV3 as encoder and an LSTM decoder trained on MS-COCO subset.

17. Smart Parking System

Description: Manage parking slots in real time, showing available slots and automatic billing.
Technologies: IoT sensors or camera-based detection (OpenCV), Node.js, PostgreSQL.
Main features: Slot occupancy detection, reservation, payment integration, admin analytics.
Difficulty: Intermediate → Advanced.
Example: Use ultrasonic sensors to detect slot occupancy and update a web dashboard.

18. E-Learning Platform with Quiz Builder

Description: A platform for creating and taking courses with integrated quizzes and progress tracking.
Technologies: Laravel or Django, React, PostgreSQL, AWS S3 for content.
Main features: Course authoring, video lectures, quizzes with auto-grading, progress analytics.
Difficulty: Intermediate.
Example: Implement adaptive quizzes that change difficulty based on past performance.

19. Voice Assistant (Basic)

Description: Build a voice assistant that can answer simple queries and perform system tasks.
Technologies: Python, SpeechRecognition library, pyttsx3 for TTS, APIs for weather/news.
Main features: Wake word, voice commands, simple task execution (open apps, search web).
Difficulty: Intermediate.
Example: Recognize “open notepad” and execute a system call to open an application.

20. URL/Classified Ads Scraper with Dashboard

Description: Scrape classified websites for certain items and show aggregated results.
Technologies: Python, BeautifulSoup/Scrapy, PostgreSQL, Dash or Flask for dashboard.
Main features: Scheduler for scraping, deduplication, price trend graphs, alerts for matches.
Difficulty: Intermediate.
Example: Scrape listings for used laptops and alert when price < threshold.

21. Online Food Ordering System (with Delivery Tracking)

Description: Customers order food, restaurants update status, and delivery is tracked in real time.
Technologies: React Native (mobile), Node.js backend, MongoDB, Google Maps API for tracking.
Main features: Menu management, order lifecycle updates, delivery routing, ratings.
Difficulty: Intermediate.
Example: Show driver location on map and compute ETA using directions API.

22. Blockchain-based Voting System

Description: Secure voting using blockchain for immutable, verifiable records.
Technologies: Ethereum (Solidity), Web3.js, React, Ganache for testing.
Main features: Voter registration, secure ballot casting, immutable ledger, results verification.
Difficulty: Advanced.
Example: Implement smart contracts to store votes and provide a front-end to read events.

23. Fitness Tracker with Workout Plans

Description: Track workouts, calories, and suggest plans based on goals.
Technologies: Flutter or React Native, SQLite or Firebase, charts for progress.
Main features: Workout logging, personalized plans, progress charts, reminders.
Difficulty: Beginner.
Example: Recommend a 4-week strength plan based on user input (beginner/intermediate).

24. Real-time Stock Market Dashboard

Description: Live visualization of stock prices, portfolio tracking, and alerts.
Technologies: React, WebSocket for live data, APIs for stock data, D3/Chart.js for graphs.
Main features: Watchlists, historical charts, technical indicators, price alerts.
Difficulty: Intermediate.
Example: Show moving average crossover signals and notify users about events.

25. Secure File Storage with Encryption

Description: Cloud-like file storage that encrypts files on client side before upload.
Technologies: Node.js, AWS S3 or Firebase Storage, crypto libraries (AES/RSA).
Main features: Client-side encryption, secure key management, shareable encrypted links.
Difficulty: Advanced.
Example: Use RSA to encrypt AES keys and AES for file encryption; store encrypted keys on server.

26. Recommendation System for Movies

Description: Build content-based and collaborative filtering recommendation engine.
Technologies: Python, scikit-learn, Surprise library, Flask for API, React dashboard.
Main features: User-based and item-based recommendations, hybrid approach, test metrics.
Difficulty: Intermediate → Advanced.
Example: Train on MovieLens dataset and serve top-N recommendations for a user.

27. Optical Character Recognition (OCR) System

Description: Convert images of documents into editable text; support multiple languages.
Technologies: Python, Tesseract OCR, OpenCV for preprocessing, Flask + React UI.
Main features: Image enhancement, text extraction, downloadable text/PDF.
Difficulty: Intermediate.
Example: Preprocess scanned images with thresholding to improve OCR accuracy.

28. Attendance Using Face Recognition

Description: Automatic attendance system that recognizes faces and logs presence.
Technologies: Python, OpenCV, face_recognition library (dlib), SQLite.
Main features: Enroll students, real-time recognition, logs, export attendance sheets.

Difficulty: Intermediate.
Example: Use HOG or CNN-based face encodings to match classroom faces.

29. Malware Analysis Sandbox (Educational)

Description: Safe environment to run and analyze suspicious files (with strict safety).
Technologies: Virtual machines (VirtualBox), Python, Cuckoo Sandbox concepts, logging and analysis.
Main features: File upload (offline), behavior monitoring, network capture (isolated), report builder.
Difficulty: Advanced (security concerns).
Example: Create a VM snapshot, run sample in VM, collect process and network logs for analysis.
Note: Implement with caution and do not run real malware on public or home networks.

30. Voice-to-Text Meeting Summarizer

Description: Transcribe meeting audio and generate concise summaries.
Technologies: Speech-to-text APIs (or open-source models), Python, NLP summarization models (transformers).
Main features: Speaker diarization, transcript export, key-point summary, search within transcripts.
Difficulty: Advanced.
Example: Record meeting, transcribe, then run an abstractive summarizer to create minutes.

31. Online Grocery Store with Inventory Forecasting

Description: E-commerce for groceries with demand forecasting for inventory.
Technologies: MERN stack, basic ML (ARIMA/Prophet), PostgreSQL, Redis cache.
Main features: Product listing, order management, stock prediction, low-stock alerts.
Difficulty: Intermediate → Advanced.
Example: Forecast next month's demand for a product and automatically suggest reorder quantities.

32. Password Manager (Local/Cloud)

Description: Securely store and retrieve user passwords with encryption and master password.
Technologies: Electron for desktop app or Flutter mobile, AES encryption, hashed master password.
Main features: Add/edit passwords, password generator, auto-lock after idle, backup/export encrypted.
Difficulty: Intermediate.
Example: Generate site-specific strong passwords and store them encrypted on disk.

33. News Aggregator with Topic Clustering

Description: Aggregate news from multiple sources and cluster similar stories into topics.
Technologies: Python, RSS/News APIs, TF-IDF embeddings or transformers, clustering algorithms.
Main features: Source aggregation, topic clusters, summarization, search.
Difficulty: Intermediate.
Example: Cluster articles about a single event from different outlets and show unified timeline.

34. Bug Tracker / Issue Management System

Description: Tool for managing project issues, sprints, and bug reports.
Technologies: Django or Rails, React frontend, PostgreSQL, JWT.
Main features: Issue creation, tagging, status workflow, assignees, comments, reporting.
Difficulty: Beginner → Intermediate.
Example: Add a Kanban board UI and auto-notify assignees on status change.

35. Data Visualization Dashboard for Open Data

Description: Visualize government or public datasets and allow interactive exploration.
Technologies: Python (Pandas), Flask, D3.js/Chart.js, PostgreSQL.

Main features: Dataset importer, filtering, interactive charts, download options.
Difficulty: Intermediate.
Example: Visualize population vs literacy rate across regions with heatmaps and line charts.

36. Auction Website (Real-time Bidding)

Description: Platform for timed auctions with real-time bidding and winner determination.
Technologies: Node.js, WebSockets, React, PostgreSQL, payment gateway integration.
Main features: Live bid updates, auto-bid, auction timers, seller dashboards.
Difficulty: Intermediate → Advanced.
Example: Implement highest-bid-wins logic and anti-sniping (extend auction if last-second bid).

37. QR Code-based Event Check-in System

Description: Generate QR codes for attendees and scan them at entry for quick check-in.
Technologies: React Native for scanner app, Node.js backend, QR code libraries.
Main features: Ticket generation, QR scanning, attendance logs, check-in reports.
Difficulty: Beginner.
Example: Generate unique encrypted token in QR code and validate on scan with server verification.

38. Task Automation Bot (Web Scraping + Actions)

Description: A bot that automates routine web tasks, like data entry or form submissions.
Technologies: Python, Selenium/Playwright, scheduling with Celery or cron.
Main features: Scriptable tasks, retry logic, error reporting, credential management.
Difficulty: Intermediate.
Example: Automate daily login and report download from an intranet site (with permission).

39. Multi-tenant SaaS Boilerplate

Description: Skeleton application supporting multiple organizations (tenants) with isolated data.
Technologies: Django/Flask with SQLAlchemy, PostgreSQL schemas, React frontend.
Main features: Tenant onboarding, role-based access, billing placeholder, single codebase multi-tenant support.
Difficulty: Advanced.
Example: Implement schema-based isolation and admin-level tenant management.

40. CAPTCHA Solver (Research Project)

Description: Analyze and attempt to solve CAPTCHA variants; focus on limitations and defenses.
Technologies: Python, OpenCV, ML models for OCR, ethical use and research-only controls.
Main features: Preprocessing pipeline, classifier, report on success rates, recommendations to strengthen CAPTCHAs.
Difficulty: Advanced.
Example: Use image segmentation then feed to an OCR model; document ethical implications and defensive suggestions.
Note: Use only for research; do not use to bypass protections illegally.

41. Medical Diagnostic Assistant (Symptom Checker)

Description: System that suggests possible conditions from symptoms using rule-based and ML approaches.
Technologies: Python, knowledge base (medical datasets), NLP for symptom parsing, web UI.
Main features: Symptom input, probable diagnoses with confidence scores, triage advice.
Difficulty: Advanced.
Example: Combine decision trees for common conditions and ML model for rare pattern detection; include disclaimers.

42. Real-time Collaborative Editor

Description: A Google Docs-like collaborative text editor with operational transformation or CRDT.
Technologies: Node.js, WebSocket, operational transformation (OT) libraries or CRDT implementations, React.
Main features: Real-time cursor position of collaborators, conflict resolution, version history.
Difficulty: Advanced.
Example: Use Yjs (CRDT) or ShareDB (OT) to handle concurrent edits and sync clients.

43. Sentiment-based Music Playlist Generator

Description: Generate playlists matching the user's mood from streaming metadata.
Technologies: Python, Spotify API, sentiment analysis on user input or recent texts.
Main features: Mood detection, song feature matching (tempo, valence), playlist creation.
Difficulty: Intermediate.
Example: Detect user mood as "energetic" and create a playlist emphasizing high-tempo tracks.

44. Dynamic Quiz Generator with Adaptive Difficulty

Description: Generates quizzes that adapt question difficulty based on user performance.
Technologies: Django or Node.js, React, a small ML model or heuristic for adaptation.
Main features: Question bank tagging by difficulty, adaptive selection, performance analytics.
Difficulty: Intermediate.
Example: If a student answers three consecutive questions correctly, increase difficulty level.

45. CAPTCHA Generator (Security Study)

Description: Create robust CAPTCHAs exploring trade-offs between usability and security.
Technologies: Python, image libraries, audio CAPTCHA support, accessibility options.
Main features: Image/text/audio CAPTCHAs, distortion algorithms, rate limiting.
Difficulty: Intermediate → Advanced.
Example: Generate image CAPTCHAs using stylized fonts and background noise; test solvability.

46. Smart To-Do App with NLP Input

Description: Take natural language input ("tomorrow 9 AM meeting") and add structured tasks with reminders.
Technologies: Flutter/React Native, NLU (spaCy/transformers), local notifications.
Main features: Natural language parsing to extract date/time/tags, recurring tasks, priority sorting.
Difficulty: Intermediate.
Example: Parse "Pay electricity bill every month" and create a recurring task with monthly reminders.

47. Containerized Microservices Demo

Description: Build a sample microservices application with multiple services and orchestrate them with Docker Compose/Kubernetes.
Technologies: Docker, Kubernetes (minikube), REST/gRPC, Redis, PostgreSQL.
Main features: Service discovery, health checks, load balancing, CI/CD pipeline example.
Difficulty: Advanced.
Example: Split an e-commerce app into catalog, orders, and user services and deploy with minikube.

48. CAPTCHA-resistant Login System

Description: Design login flows minimizing bot access using rate limiting, device fingerprinting, and progressive challenges.
Technologies: Node.js, Redis, fingerprintJS, reCAPTCHA integration options.
Main features: Brute-force protection, progressive challenge escalation, audit logs.

Difficulty: Advanced.
Example: Lockout policies combined with incremental challenge difficulty for suspicious IPs.

49. Language Learning App with Spaced Repetition

Description: Mobile/web app that schedules vocabulary reviews using spaced repetition algorithms (SRS).
Technologies: React Native or Flutter, local storage or Firebase, SM-2 algorithm implementation.
Main features: Flashcards, SRS scheduling, progress statistics, audio pronunciation.
Difficulty: Intermediate.
Example: Implement SM-2 to schedule next review based on user recall rating.

50. Automated Code Grading Platform

Description: A platform that runs student code against test cases with sandboxing and reports results.
Technologies: Docker sandboxing, Node.js/Python backend, language runners, database for submissions.
Main features: Support multiple languages, test case management, plagiarism checks, leaderboard.
Difficulty: Advanced.
Example: Use Docker containers to compile/run code safely and compare outputs with expected results.

Must Read: 150 8D Project Ideas 2025-26

Tips to implement these projects successfully

1. **Start with an MVP:** Implement core functionality first (e.g., login + main feature), then extend.
2. **Version control:** Use Git and push frequently with clear commits.
3. **Testing:** Add unit tests and integration tests for key modules.

4. **Documentation:** Maintain README, installation steps, and usage examples—important for grading interviews.
5. **UI/UX:** Even a simple, clean interface elevates presentation.
6. **Deployment:** Deploy on Heroku, Vercel, Netlify, or a VPS to show live demos.
7. **Security & Privacy:** Handle user data responsibly—hash passwords and validate input to avoid vulnerabilities.
8. **Use templates & libraries:** Don't reinvent the wheel—use established libraries (Bootstrap/Tailwind, Auth libraries).
9. **Time management:** Allocate weeks for planning, implementation, testing, and documentation.

Conclusion

These **50 engineering project ideas for CSE** span beginner to advanced levels and cover web/mobile development, machine learning, computer vision, IoT, blockchain, security, and system design.

Choose a project that aligns with your learning goals and available time.

Start with a clear plan, implement an MVP, and iterate.

Good documentation and a hosted demo significantly increase the impact of your project when applying for internships or showcasing in your portfolio.

Blog, Project Ideas

JOHN DEAR

I am a creative professional with over 5 years of experience in coming up with project ideas. I'm great at brainstorming, doing market research, and analyzing what's possible to develop innovative and impactful projects. I also excel in collaborating with teams, managing project timelines, and ensuring that every idea turns into a successful outcome. Let's work together to make your next project a success!

49+ Birthday Project Ideas — Creative Projects for Student

Best Project Ideas

Are you ready to make your big ideas happen? Let's connect and discuss how we can bring your vision to life. Together, we can create amazing results and turn your dreams into reality.

Top Pages

Terms And Conditions

Disclaimer

Privacy Policy

Follow Us