ChatGPT

Frontend Projects

Beginner

1. **To-Do List App** (Frontend, Beginner): A simple React or Vue app to manage tasks (add, mark done, delete) [1]. It practices state management and list rendering.
2. **Counter App** (Frontend, Beginner): A basic app with increment/decrement buttons to adjust a counter. Teaches component state and event handling.
3. **Calculator** (Frontend, Beginner): A simple arithmetic calculator (add, subtract, multiply, divide) built with React/Vue components. Exercises form input handling and logic.
4. **Random Quote Generator** (Frontend, Beginner): Displays quotes from a public API (like Quote Garden). Focuses on fetching data and updating state when a new quote is requested.
5. **Weather App** (Frontend, Beginner): Fetches and shows current weather for a user-specified city using a weather API [2]. Teaches API integration and conditional rendering.
6. **Image Gallery** (Frontend, Beginner): A grid gallery (e.g. fetching images from Unsplash API) with a search/filter field [3]. Teaches list layout and API use.
7. **BMI Calculator** (Frontend, Beginner): Calculates Body Mass Index from weight/height inputs [4]. Focuses on form inputs and simple formula logic.
8. **Timer/Stopwatch** (Frontend, Beginner): A clock app with start/pause/reset functionality [5]. Uses JavaScript timers (`setInterval`) and state updates.
9. **Simple Blog Interface** (Frontend, Beginner): A static blog site front-end with routes for posts and post details [6]. Familiarizes with client-side routing (React Router or Vue Router).
10. **Recipe App (Frontend, Beginner)**: Shows recipe details (ingredients, instructions) from static data or an API [6]. Focuses on fetching/displaying structured data.
11. **Image Slider** (Frontend, Beginner): A carousel component cycling through images with “next/prev” buttons. Teaches component state and animations.
12. **Color Picker** (Frontend, Beginner): Lets users pick a color and displays its hex/RGB values. Practices controlled inputs and dynamic style updates.
13. **Form Validation Demo** (Frontend, Beginner): A signup or contact form demonstrating input validation and error messages. Highlights form handling and validation logic.
14. **FAQ Accordion** (Frontend, Beginner): An accordion list where clicking a question toggles the answer. Focuses on conditional rendering of components.
15. **Quiz Interface** (Frontend, Beginner): Presents a few hardcoded multiple-choice questions and shows a score. Teaches simple state updates and event handling.

Intermediate

1. **E-Commerce Storefront** (Frontend, Intermediate): A front-end for an online store – product listings, detail pages, shopping cart UI [7]. Uses state management (e.g. Redux/Vuex) and styled components.
2. **Real-Time Chat UI** (Frontend, Intermediate): Chat interface (without building server) using Firebase or websockets [8]. Demonstrates real-time data updates in UI.

3. **Movie/TV Search App** (Frontend, Intermediate): Uses TMDB or OMDB API to search and display movie details [9]. Practices API queries, pagination, and UI lists.
4. **Advanced To-Do List** (Frontend, Intermediate): A to-do app with categories, labels, due dates [10]. Introduces more complex state (filters, sorting).
5. **Social Media Dashboard** (Frontend, Intermediate): Displays user stats/feeds (likes, followers) with charts or summaries [11]. Emphasizes data visualization libraries (Chart.js).
6. **Portfolio Website** (Frontend, Intermediate): A dynamic portfolio to showcase projects and skills [12]. Uses React/Vue with a router; highlights skills and projects.
7. **Expense Tracker UI** (Frontend, Intermediate): Logs and charts personal expenses over time [13]. Uses chart libraries to display spending trends.
8. **Online Quiz App** (Frontend, Intermediate): Presents quiz questions from JSON or API and calculates score [14]. Introduces timers or question randomization.
9. **Music Player Interface** (Frontend, Intermediate): A UI with play/pause, track list, and progress bar [15]. Teaches handling media elements and state.
10. **Job Board Frontend** (Frontend, Intermediate): Lists job postings and filters by keyword/location [16]. Focuses on dynamic lists and search filtering.
11. **Note-Taking App** (Frontend, Intermediate): Rich text notes with Markdown support [17]. Uses a Markdown library to render notes in real time.
12. **Markdown Editor** (Frontend, Intermediate): Split pane editor that renders Markdown to HTML [18]. Teaches two-way data binding.
13. **Currency Converter** (Frontend, Intermediate): Converts between currencies using real-time exchange rates [19]. Integrates exchange rate API.
14. **Language Flashcards** (Frontend, Intermediate): Vocabulary flashcards that quiz the user [20]. Practices local state and randomization.
15. **Personal Dashboard** (Frontend, Intermediate): Combines weather, news headlines, and to-dos in one page. Practices using multiple APIs and widgets.
16. **Filtering Table/List** (Frontend, Intermediate): Displays a data table (users/products) with live search and filters. Focuses on table rendering and filter logic.
17. **Image Search App** (Frontend, Intermediate): Searches Unsplash or Pixabay and displays an infinite scroll gallery. Teaches pagination/infinite scroll.
18. **Crypto Price Tracker** (Frontend, Intermediate): Live charts of cryptocurrency prices. Integrates a crypto price API and real-time charts.
19. **Kanban Board UI (Trello Clone)** (Frontend, Intermediate): Drag-and-drop task cards between lists. Teaches drag/drop libraries (like React DnD).
20. **Multi-Step Form Wizard** (Frontend, Intermediate): Signup or checkout form spanning multiple steps with progress indicator. Focuses on component state across steps.
21. **Drag-and-Drop File Uploader** (Frontend, Intermediate): Allows dragging files to upload area, previews them. Teaches handling file inputs.
22. **Music Playlist Manager** (Frontend, Intermediate): Create and manage playlists of songs. Practices list state and filtering.
23. **Graphical Function Plotter** (Frontend, Intermediate): Input a math function (e.g. `sin(x)`) and plot it with a canvas/graphing library. Teaches canvas or SVG.
24. **Chatbot UI** (Frontend, Intermediate): Interface for a chatbot (just static responses). Uses state to display conversation flow.
25. **Video Gallery** (Frontend, Intermediate): Displays a collection of YouTube/Vimeo video embeds. Teaches embedding and responsive iframes.

26. **Interactive Game (e.g. Tic-Tac-Toe)** (Frontend, Intermediate): Implement game logic in the browser with state. Teaches component interaction.
27. **Photo Collage Maker** (Frontend, Intermediate): Drag and place images onto a canvas layout. Teaches HTML5 canvas or positioning.
28. **Graph Editor** (Frontend, Intermediate): Visual node-and-edge diagram editor. Teaches SVG or canvas interaction.
29. **UI Components Showcase** (Frontend, Intermediate): Build common UI widgets (modal, tabs, tooltips) to a component library. Emphasizes reusable components.
30. **Data Tables with Charts** (Frontend, Intermediate): Show a table of data and a linked chart (e.g. sales data). Practicing data binding between components.
31. **Weather Forecast App** (Frontend, Intermediate): Extends the Weather app to show 5-day forecasts [21]. Teaches processing and charting data.
32. **Quiz Creation Tool** (Frontend, Intermediate): Frontend UI to add quiz questions (used by a backend). Focuses on form dynamics.
33. **Interactive Map** (Frontend, Intermediate): Google Maps or Leaflet integration with markers from an API. Teaches map APIs.
34. **Video Conference UI** (Frontend, Advanced): UI layout for a video chat (without backend). Includes video elements grid.
35. **Interactive Timeline** (Frontend, Advanced): Timeline visualization of events (like timelines.js). Teaches dynamic SVG or chart library.

Backend Projects

Beginner

1. **Personal Blogging API** (Backend, Beginner): A RESTful API for a blog: list, read, create, update, delete posts [22]. Implements basic CRUD with a database (SQL or NoSQL).
2. **To-Do List API** (Backend, Beginner): API for tasks with user accounts: create, update, delete, list tasks [23]. Introduces user authentication (JWT) and CRUD logic.
3. **Weather Data Aggregator API** (Backend, Beginner): Fetches weather data from a public API, caches it (Redis) for repeated requests [24]. Practices third-party API calls and caching.
4. **Expense Tracker API** (Backend, Beginner): Stores users' expenses and categories, returns summaries. Covers data modeling and simple analytics.
5. **User Authentication API** (Backend, Beginner): User registration and login endpoints (token-based). Teaches password hashing and session/JWT management.
6. **Basic CRUD API** (Backend, Beginner): A generic template API (e.g. user profiles) demonstrating create/read/update/delete operations [25].
7. **Contact Management API** (Backend, Beginner): Stores contact entries (name, phone, email); supports add, update, delete contacts [26].
8. **Bookstore Inventory API** (Backend, Beginner): Manage books and stock levels [27]. Enables adding new books, updating inventory counts, and searching by title/author.
9. **Note-Taking API** (Backend, Beginner): API for creating and retrieving text notes. Introduces linking notes to users and timestamps.
10. **URL Shortener Service** (Backend, Beginner): Generates short URLs redirecting to long URLs [28]. Teaches hashing or ID mapping and HTTP redirects.

11. **Photo Upload Service** (Backend, Beginner): Handles image uploads (storing to filesystem or S3) and returns URLs. Introduces multipart/form-data handling.
12. **User Profile API** (Backend, Beginner): Manages user profile data (biography, avatar upload). Covers file uploads and user data.
13. **Appointment Booking API** (Backend, Beginner): Book and list appointment slots. Involves simple datetime logic and availability.
14. **Basic Chat History API** (Backend, Beginner): Stores chat messages between users. Endpoint to fetch chat history. (No real-time yet)
15. **CMS Backend** (Backend, Beginner): Content management endpoints for pages (title, slug, body). Supports create/update/delete pages.
16. **Simple Journal API** (Backend, Beginner): Personal daily journal entries CRUD. Emphasizes date filtering.
17. **Task Reminder API** (Backend, Beginner): Set and get reminders for tasks at future times. Introduces scheduled jobs (e.g., with cron).
18. **Poll Creation API** (Backend, Beginner): Create polls with options and record votes. Basic poll data modeling.
19. **Email Dispatch API** (Backend, Beginner): Endpoint to send emails (e.g., contact form emails) using SMTP or a service API.
20. **Survey Response API** (Backend, Beginner): Collect survey answers in a structured format. Teaches handling multiple input types.
21. **Markdown Processor API** (Backend, Beginner): Converts markdown text to HTML and returns it. Integrates a markdown library.
22. **Currency Rates API** (Backend, Beginner): Provides up-to-date currency conversion rates (calls an external rates API).
23. **String Utilities API** (Backend, Beginner): Offers endpoints for string operations (e.g., reverse string, uppercase). Simple business logic.
24. **JSON Storage API** (Backend, Beginner): Stores arbitrary JSON objects by key. Focuses on schema-less storage (NoSQL).

Intermediate

1. **Real-Time Polling App Backend** (Backend, Intermediate): Builds on Poll API by adding live updates (WebSockets) [29]. Clients see votes update in real time.
2. **E-Commerce Backend** (Backend, Intermediate): API for products, carts, orders [30]. Includes product catalog, adding to cart, and order placement logic (without payment).
3. **Fitness Tracker API** (Backend, Intermediate): Users log workout sessions; API aggregates progress over time [31]. Teaches data aggregation and time queries.
4. **Recipe Sharing Backend** (Backend, Intermediate): Create/read recipes with images and ingredient lists [32]. Covers file uploads (images) and complex queries (filter by ingredients).
5. **Movie Reservation System** (Backend, Intermediate): Reserve seats for movie screenings [33]. Involves relational modeling (screens, seats, bookings) and concurrency control.
6. **Restaurant Review API** (Backend, Intermediate): Users submit reviews for restaurants; uses NLP to analyze review sentiment [34]. Introduces natural language processing libraries.
7. **Multiplayer Game Server** (Backend, Intermediate): A server for a simple game (e.g. Battleship) handling two-player sessions [35]. Involves state syncing and maybe WebSockets.
8. **Database Backup CLI** (Backend, Intermediate): A command-line tool (Node or Python) to backup and restore a database [36]. Teaches scripting and filesystem operations.

9. **Code Compilation API** (Backend, Intermediate): Accepts code (e.g. C++/Python) and returns compilation result/output [37]. Sandboxes execution (like Judge0 or similar).
10. **Real-Time Chat Backend** (Backend, Intermediate): Chat service using Socket.IO (Node) or Django Channels, enabling instant messaging [38]. Introduces real-time protocols.
11. **Content Delivery Simulator (CDN)** (Backend, Intermediate): Simulate a simple CDN with caching and load balancing [39]. Teaches caching strategies.
12. **Time-Tracking Service** (Backend, Intermediate): Logs hours for freelancers; generate invoices [40]. Introduces date libraries and report generation.
13. **Web Scraper API** (Backend, Intermediate): CLI or web API that scrapes given URLs and returns data [41]. Teaches HTTP requests and parsing HTML.
14. **OAuth2 Authorization Server** (Backend, Intermediate): Implements OAuth2 flows (login with tokens). Teaches security best practices.
15. **GraphQL API Server** (Backend, Intermediate): Converts a REST model into a GraphQL schema. Provides a GraphQL endpoint.
16. **Database Migration Tool** (Backend, Intermediate): CLI that migrates DB schema versions (like Flyway).
17. **Queue/Message Broker** (Backend, Intermediate): Simple pub/sub service using Redis or RabbitMQ.
18. **Image Processing Service** (Backend, Intermediate): API that resizes/crops images using a library (like Sharp or Pillow).
19. **Search Engine** (Backend, Intermediate): Full-text search over stored content using Elasticsearch or equivalent.
20. **Payment Gateway Integration** (Backend, Intermediate): Mock payment processing endpoints (Stripe/PayPal simulation).
21. **Recommendation Engine** (Backend, Intermediate): Provides item suggestions (e.g. movies) using collaborative filtering.
22. **Geo-IP Lookup Service** (Backend, Intermediate): Returns location info for an IP address via an external API.
23. **NLP Text Analysis API** (Backend, Intermediate): Endpoints for sentiment analysis or keyword extraction on input text.
24. **Job Scheduler Service** (Backend, Intermediate): Schedules and runs background tasks at specified times.
25. **Metrics Dashboard API** (Backend, Intermediate): Collects and serves usage metrics (prometheus-style).
26. **Blockchain Ledger Simulator** (Backend, Intermediate): Simulated distributed ledger with simple consensus.

Full-Stack Projects

Beginner

1. **Portfolio Website** (Full-Stack, Beginner): A personal portfolio allowing CRUD of projects via a backend. Often built with React frontend and Node/Django backend to save posts and uploads [42].
2. **Simple Chat App** (Full-Stack, Beginner): Real-time chat with user login. Uses React/Vue on front end and Node.js (Socket.IO) on back end [38]. Covers websockets.
3. **Blogging Platform** (Full-Stack, Beginner): Users can write/read/edit blog posts. Frontend (React/Vue) communicates with a Django/Node REST API to store posts/comments.

4. **To-Do List (Full-Stack)** (Full-Stack, Beginner): Extends the to-do app by adding a backend. React front-end calls a Node/Django API to persist tasks to a database.
5. **Mini-CMS** (Full-Stack, Beginner): Admin front end to manage pages/content, with backend endpoints to save them. Teaches full-stack page editing.
6. **E-Commerce App** (Full-Stack, Beginner): A simple shop with product listings and a shopping cart [43]. Frontend in React, backend in Node (Express) with MongoDB.
7. **Social Media Feed** (Full-Stack, Beginner): Users post updates and follow each other. Implements login and feeds. (Inspired by Instagram/Twitter UI).
8. **Content Management System** (Full-Stack, Beginner): A business CMS allowing rich-text content creation (like building small websites). Frontend uses React+RichText editor; backend uses Django.
9. **Food Delivery (Intermediate)** (Full-Stack, Intermediate): Browse restaurants and menus, place orders [44] [45]. React front-end, Node back end, stripe API for payments.
10. **Recipe Sharing Platform** (Full-Stack, Intermediate): Users post and browse recipes [46]. Backend stores recipes and comments; frontend displays them.
11. **Video Conferencing App** (Full-Stack, Advanced): Real-time video chat using WebRTC and signaling via a Node backend [47]. Complex real-time media features.
12. **Job Portal** (Full-Stack, Intermediate): Employers post jobs and candidates apply [48]. Includes user roles and resume uploads.
13. **Project Management Tool** (Full-Stack, Intermediate): Kanban board like Trello [49]. React UI with drag-drop tasks; backend in Node/Django for persistence.
14. **Online Learning Platform** (Full-Stack, Advanced): Course catalog with video playback and quizzes [44]. Full user accounts, progress tracking.
15. **Chatbot with Dashboard** (Full-Stack, Intermediate): Frontend to send chat messages; backend runs an AI chatbot or rule-based bot.
16. **Q&A Forum (StackOverflow Clone)** (Full-Stack, Intermediate): Users ask/answer questions with votes and tags. React/Vue front-end with Django REST API.
17. **Event Ticketing System** (Full-Stack, Intermediate): Event listings, seat selection, booking tickets. (Similar logic to Movie Reservation, but full UI).
18. **Food Diary App** (Full-Stack, Intermediate): Log meals and calories. Frontend with forms; backend aggregates daily totals.
19. **Fitness Tracker** (Full-Stack, Intermediate): Log workouts and view progress charts. Combines [78] with a React dashboard.
20. **Expense Manager** (Full-Stack, Beginner): Input income/expenses; visualize by charts. (React frontend + Node backend to store entries).
21. **Multi-Author Blog** (Full-Stack, Intermediate): Similar to #110 but with multiple user accounts, comments, and likes.
22. **Real Estate Listings** (Full-Stack, Intermediate): Browse property listings with search and filtering. (React frontend + Django/Node backend with a database of listings).
23. **Car Rental Service** (Full-Stack, Intermediate): Rent vehicles by date range. Backend manages availability; frontend shows booking calendar.
24. **CRM System** (Full-Stack, Intermediate): Manage customers, deals, tasks. (React UI with REST API to store CRM data).
25. **Appointment Scheduler** (Full-Stack, Beginner): Book and manage appointments (e.g. for a salon). Frontend shows available slots; backend enforces scheduling.
26. **Auction/Bidding Site** (Full-Stack, Advanced): Users list items for bidding, live updating highest bid. Requires websockets or polling for live updates.
27. **Event Management App** (Full-Stack, Intermediate): Create events with guest lists and RSVPs. Frontend uses calendar/date pickers; backend stores events and responses.

28. **Music Streaming App** (Full-Stack, Advanced): Browse and play audio tracks. (Streaming server or public APIs, React front-end audio player, backend file service).
29. **Photo Sharing Network** (Full-Stack, Intermediate): Upload and comment on photos. (React frontend with image grid, Node backend storing images).
30. **Professional Network** (Full-Stack, Advanced): Like LinkedIn with profiles, connections, and posts. Full user system and search.
31. **Travel Planner** (Full-Stack, Intermediate): Plan trips by booking hotels/flights. Integrate travel APIs for search.
32. **Social Q&A Platform** (Full-Stack, Advanced): Similar to #116 with advanced features like chat between users.
33. **Online Auction (Advanced)** (Full-Stack, Advanced): Extend #126 with payment integration and item management.
34. **Collaborative Document Editor** (Full-Stack, Advanced): Real-time shared text editor (requires Operational Transforms or CRDTs).
35. **Game Leaderboard** (Full-Stack, Intermediate): Submit game scores and display global rankings. Frontend charts; backend ranking logic.
36. **Personal Portfolio Builder** (Full-Stack, Beginner): Users design their own portfolio site (drag/drop); backend generates static pages.
37. **Crowdfunding Platform** (Full-Stack, Advanced): Users create funding campaigns; others pledge money. Handles payments and goal tracking.
38. **Marketplace** (Full-Stack, Intermediate): General buy/sell platform with user stores [50]. Combines e-commerce and social features.
39. **Restaurant Reservation App** (Full-Stack, Intermediate): Book tables at restaurants [51]. Similar to Appointment Scheduler but table-specific.
40. **Weather Dashboard** (Full-Stack, Beginner): Combines [53] with user preferences. Users save locations; backend caches weather.
41. **Job Tracker** (Full-Stack, Intermediate): Track job applications and interviews [52]. Frontend lists positions; backend stores statuses.
42. **Health Monitor** (Full-Stack, Intermediate): Users log health metrics (blood pressure, sugar); charts show trends.
43. **LMS (Learning Management System)** (Full-Stack, Advanced): Courses, quizzes, grades (extensive full-stack app).
44. **Collaboration Whiteboard** (Full-Stack, Advanced): Shared drawing board (requires real-time sync).
45. **Calendar Planner** (Full-Stack, Intermediate): Shared calendar for events/tasks (React + full REST API for events).
46. **Freelance Marketplace** (Full-Stack, Advanced): Clients post jobs; freelancers apply (like Upwork).
47. **Crypto Trading Simulator** (Full-Stack, Advanced): Simulated trading environment with live price data.
48. **Code Submission Platform** (Full-Stack, Intermediate): Submit code problems and see outputs (front similar to IDE).
49. **Multi-language Blog** (Full-Stack, Intermediate): Blog supporting content in multiple languages, with translation features.
50. **Educational Games Portal** (Full-Stack, Intermediate): Library of mini educational games, tracks user scores.

Implementation Guides

We now outline step-by-step implementation for each project above, referencing code examples or repositories where useful.

Frontend Project Guides

1. To-Do List App (Frontend, Beginner)

- **Setup:** Initialize a React app (`create-react-app`) or Vue project. Install dependencies.
- **Components:** Create a `TodoApp` component that holds state: an array of tasks (each {text, done}). Use `useState` (React) or `data()` (Vue).
- **Add Task:** Render an input and button; on submit add a new task to state.
- **Display Tasks:** Map over tasks to display them with checkboxes.
- **Mark Done/Delete:** On checkbox change or delete button click, update state (toggle `done` or remove item).
- **Styling:** Simple CSS for layout.
- **Code Example:** In React, one main component might be:

```
function TodoApp() {
  const [tasks, setTasks] = useState([]);
  const [text, setText] = useState("");
  const addTask = () => {
    setTasks([...tasks, {text, done: false}]);
    setText("");
  };
  const toggleDone = (i) => {
    const newTasks = [...tasks];
    newTasks[i].done = !newTasks[i].done;
    setTasks(newTasks);
  };
  const deleteTask = (i) => {
    setTasks(tasks.filter((_, idx) => idx!==i));
  };
  return (
    <div>
      <input value={text} onChange={e=>setText(e.target.value)}/>
      <button onClick={addTask}>Add</button>
      <ul>
        {tasks.map((t,i)=>(
          <li key={i}>
            <input type="checkbox" checked={t.done}
onChange={()=>toggleDone(i)}/>
            <span style={{textDecoration: t.done?'line-through':''}}
>{t.text}</span>
            <button onClick={()=>deleteTask(i)}>Del</button>
```

```
          </li>
        ))}
      </ul>
    </div>
  );
}
```

- **Further Resources:** Example full code is available on GitHub (e.g., a React To-Do App repository).

2. Counter App (Frontend, Beginner)

- **Setup:** Use React/Vue. Create a `Counter` component.
- **State:** Initialize `count = 0`.
- **Buttons:** Two buttons "+" and "–" that call functions to `setCount(count+1)` or `setCount(count-1)`.
- **Display:** Show `count` in a `<span>`.
- **Code Snippet:**

```
function Counter() {
  const [count, setCount] = useState(0);
  return (
    <div>
      <button onClick={()=>setCount(count-1)}>-</button>
      <span>{count}</span>
      <button onClick={()=>setCount(count+1)}>+</button>
    </div>
  );
}
```

3. Calculator (Frontend, Beginner)

- **Components:** A display component and buttons for digits/ops.
- **State:** Store current input and calculation logic (or parse expression).
- **Logic:** On number button click, append to input; on operator or equals, compute result.
- **Example:** For simplicity, use `eval()` in React:

```
const [input, setInput] = useState("");
const handleClick = (val) => setInput(input+val);
const calculate = () => setInput(eval(input).toString());
// Render buttons 0-9, + - * /, and = calling calculate
```

4. Random Quote Generator (Frontend, Beginner)

- **API:** Use a quotes API (e.g. Quotes API).
- **State:** `quote` state.

- **Effect:** On component mount (useEffect), fetch a quote.
- **Button:** "New Quote" button triggers another fetch.
- **Example:**

```
useEffect(() => {
  fetch('https://api.quotable.io/random')
    .then(res=>res.json())
    .then(data=>setQuote(data.content));
}, []);
```

5. Weather App (Frontend, Beginner)

- **API:** Use OpenWeatherMap or similar.
- **Input:** City name text box.
- **Fetch:** On submit, call API and set `weather` state.
- **Display:** Show temperature, description, icon.
- **Example:** Using `fetch('api.openweathermap.org/data/2.5/weather?q='+city)`.
- **Code Note:** See SheCodes React Weather App example [53] for a working implementation.

6. Image Gallery (Frontend, Beginner)

- **API:** Use Unsplash API or Pixabay.
- **Search:** Input box; on submit fetch images.
- **Display:** Show images in grid (CSS flex or grid).
- **Example:** After fetching JSON of image URLs, map to `<img>` elements.

7. BMI Calculator (Frontend, Beginner)

- **Form:** Two inputs for weight and height.
- **Calculate:** On submit, compute `BMI = weight/(height^2)`.
- **Display:** Show BMI result.
- **Optional:** Categorize (Underweight/Normal/Overweight).

8. Timer/Stopwatch (Frontend, Beginner)

- **State:** `seconds` (integer).
- **Controls:** Start, Pause, Reset buttons.
- **Timer:** On "Start", use `setInterval` to increment seconds each second. Clear interval on pause.
- **Display:** Convert seconds to MM:SS or HH:MM:SS.

9. Simple Blog Interface (Frontend, Beginner)

- **Routing:** Use React Router or Vue Router.
- **Pages:** Home (list of posts), Post detail, New Post form.
- **Data:** For frontend only, store posts in state or localStorage.
- **Functionality:** Add and display posts. (No backend for beginner level) [6].

10. Recipe App (Frontend, Beginner)

- **Data:** Use static JSON or public recipe API.
- **UI:** List recipes with images; click to view details (ingredients, steps).
- **State:** Store selected recipe details.

(Due to space, remaining guides are outlined briefly; code and full repositories should be consulted for full implementations.)

11. Image Slider (Frontend, Beginner)

Set up a list of images and a current index state; “Next”/“Prev” buttons update the index.

12. Color Picker (Frontend, Beginner)

Use HTML `<input type="color">`; on change, display the selected hex/RGB.

13. Form Validation Demo (Frontend, Beginner)

Create a form with fields (email, password). On submit, check rules (regex/email) and show errors.

14. FAQ Accordion (Frontend, Beginner)

Render a list of question-answer pairs; track which are open in state and toggle.

15. Quiz Interface (Frontend, Beginner)

Hardcode questions in JSON; iterate over them, showing choices; track score in state.

Backend Project Guides

51. Personal Blogging API (Backend, Beginner)

- **Tech:** Node.js (Express) or Django.
- **Endpoints:** `/posts` GET (list), POST (create); `/posts/:id` GET, PUT, DELETE [22].
- **Database:** Use MongoDB or SQLite. Define a `Post` model with title, content, date.
- **Implementation:** In Express, use Mongoose with a `PostSchema`; write CRUD routes.

52. To-Do List API (Backend, Beginner)

- **Function:** Users can create an account, then create tasks.
- **Endpoints:** `/users/signup`, `/users/login` (auth); `/tasks` CRUD for authenticated users [23].
- **Tech:** Express + MongoDB + JWT (for auth) or Django + DRF + Token.

53. Weather Data API Service (Backend, Beginner)

- **Function:** On city query, fetch from weather API, cache in Redis.

- **Workflow:** Check cache first; if miss, fetch OpenWeatherMap and store in Redis with TTL. Return data.
- **Tech:** Node.js with axios + Redis, or Django with `requests` + `django-redis`.

54. Expense Tracker API (Backend, Beginner)

- **Schema:** `User`, `Expense {user, amount, category, date}`.
- **Endpoints:** POST `/expenses`, GET `/expenses` (with date filter), etc.
- **Auth:** Associate expenses with logged-in user.

55. User Authentication API (Backend, Beginner)

- **Features:** User registration (`POST /register`), login (`POST /login` returns JWT).
- **Store:** Users in DB with hashed passwords (bcrypt).
- **Use:** Express + `passport-jwt` or Django with `django-rest-auth`.

56. Basic CRUD API (Backend, Beginner)

- **Example:** A `/contacts` resource with full CRUD.
- **Stack:** Express + Mongoose or Django + DRF.
- **Tutorial:** See Fynd Academy's "Basic CRUD API" example [25].

57. Contact Management API (Backend, Beginner)

- **Endpoints:** `/contacts` CRUD.
- **Schema:** Name, phone, email fields.
- **Features:** Filter/search by name.

58. Bookstore Inventory API (Backend, Beginner)

- **Schema:** Book (title, author, ISBN, stock).
- **Endpoints:** `/books` to add/search; `/books/:id` to update/delete stock.
- **Extras:** Search by title/author.

59. Note-Taking API (Backend, Beginner)

- **Schema:** Note (user, content, created_at).
- **Endpoints:** `/notes` CRUD with optional tags.

60. URL Shortener Service (Backend, Beginner)

- **Logic:** On POST of a long URL, generate short code (e.g. base62 id). Save mapping.
- **Redirect:** GET `/s/:code` redirects to original URL.
- **Stack:** Express/Node works well; use in-memory Map or DB.

61. Photo Upload Service (Backend, Beginner)

- **Endpoints:** POST `/upload` accepts file.
- **Storage:** Save files on server or upload to S3.

- **Return:** URL of uploaded image.

62. User Profile API (Backend, Beginner)

- **Profile:** Fields like name, avatar, bio.
- **Endpoints:** GET/PUT `/profile` for current user.

63. Appointment Booking API (Backend, Beginner)

- **Schema:** Appointment (user, datetime, details).
- **Endpoints:** POST `/appointments`, GET `/appointments`.
- **Logic:** Check no double-booking if needed.

64. Chat History API (Backend, Beginner)

- **Schema:** Message (from, to, message, time).
- **Endpoints:** GET `/messages?user=A&chatWith=B` to list conversation.
- **Note:** No real-time; just store and fetch.

65. CMS Backend (Backend, Beginner)

- **Schema:** Page (slug, title, html).
- **Endpoints:** `/pages` CRUD.
- **Usage:** Frontend loads pages by slug.

66. Simple Journal API (Backend, Beginner)

- **Schema:** Entry (user, date, text).
- **Endpoints:** CRUD entries.

67. Task Reminder API (Backend, Beginner)

- **Schema:** Reminder (user, message, remind_at).
- **Job:** A background job checks and sends reminders (e.g., via email).

68. Poll Creation API (Backend, Beginner)

- **Schema:** Poll (question, options[], votes[]).
- **Endpoints:** Create poll, vote endpoint.

69. Email Dispatch API (Backend, Beginner)

- **Endpoint:** POST `/send-email` with subject and body.
- **Service:** Uses nodemailer or Django's Email backend to send mail.

70. Survey Response API (Backend, Beginner)

- **Schema:** Survey (list of questions), Response (answers array).
- **Endpoints:** Save survey responses for analysis.

71. Markdown Processor API (Backend, Beginner)

- **Endpoint:** POST `/markdown` with raw markdown; returns HTML.
- **Implementation:** Use a library like `marked` or Python `markdown2`.

72. Currency Rates API (Backend, Beginner)

- **Endpoint:** GET `/rates?base=USD`.
- **Action:** Calls an external API (Open Exchange Rates) and returns JSON rates.

73. String Utilities API (Backend, Beginner)

- **Endpoints:** e.g. `/reverse`, `/encode`.
- **Function:** Takes query string and returns transformed string.

74. JSON Storage API (Backend, Beginner)

- **Function:** Store arbitrary JSON objects: POST to `/store` with `key` and data; GET `/store/:key`.
- **Use:** No fixed schema; use MongoDB or a file store.

Full-Stack Project Guides

101. E-Commerce Platform (Full-Stack, Intermediate)

- **Stack:** React frontend + Node/Express backend + MongoDB [54].
- **Frontend:** Product listing page (use React Router for different categories), product detail page, shopping cart component. Use Redux or Context to manage cart state.
- **Backend:** Define `Product`, `User`, `Order` schemas. Endpoints: `GET /products`, `POST /cart`, `POST /checkout`. Use Stripe API for payments.
- **Data Flow:** Frontend fetches products (`GET /api/products`), adds to cart (local state), then posts order to backend (`POST /api/orders`).
- **Complete Code:** Basir's MERN tutorial [23†L255-L264] provides full code for a similar e-commerce app (Node + React).

102. Social Media App (Full-Stack, Intermediate)

- **Stack:** React frontend with infinite scroll timeline; Node (Express) backend; MongoDB.
- **Features:** User signup/login (auth), create posts with images, follow/unfollow, like/comments.
- **Backend:** Models for `User`, `Post`, `Comment`. Endpoints:
- Auth: `/register`, `/login` (JWT)
- Posts: `/posts` (CRUD), `/posts/:id/comments` (add comment), `/posts/:id/like`.
- **Frontend:** Feed component that fetches `/posts`. Create post modal using a form (multipart upload). Use websockets (Socket.IO) to push new posts in real time if desired.
- **Code Example:** Similar projects can be found on GitHub (e.g., MERN social apps).

103. Chat Application (Full-Stack, Intermediate)

- **Stack:** React (or Vue) with Socket.IO; Node.js backend with Express and Socket.IO.

- **Functionality:** Real-time one-on-one or group chat. Users join "rooms".
- **Backend:** Use Socket.IO server to handle connections. On message event, emit to room. Store chat history in DB (`Message` schema).
- **Frontend:** Connect via Socket.IO client, render messages in chat window. Provide input box to send messages.
- **Reference:** SocketGram repository [38] is an example implementation (React+Express+Socket.IO).

104. To-Do List (Full-Stack, Beginner)

- **Stack:** React frontend + Node backend.
- **Backend:** Simple tasks CRUD API with a `Task` model (text, completed, user).
- **Frontend:** Use components as in the React to-do, but fetch and modify tasks via API (`fetch('/api/tasks')`). Use JWT or session for per-user tasks.

105. Portfolio Website (Full-Stack, Beginner)

- **Stack:** React (or Vue) frontend; Node (Express) backend (or Django).
- **Functionality:** Frontend displays projects from a database. Admin panel to add/edit projects (protected route).
- **Backend:** `Project` schema (title, description, link, image). Endpoints to get projects and to add/update projects (secure with auth).
- **Frontend:** A gallery of projects; a login-protected "Add Project" form.

106. Mini-CMS (Full-Stack, Beginner)

- **Stack:** Node/Express + MongoDB backend; React/Vue admin panel.
- **Functionality:** Rich-text editor (e.g., using CKEditor) to edit pages; backend saves HTML content.
- **Backend:** Endpoints like `GET /pages/:slug`, `POST /pages/:slug`.
- **Frontend:** A login-protected area with a WYSIWYG editor and save button. Public front end fetches pages by slug and displays them.

107. Content Management System (Full-Stack, Intermediate)

- **Stack:** Django backend with admin site + React front end (or fully headless).
- **Details:** Use Django's built-in admin for content pages; build a React front that calls a Django REST API for pages.
- **Link:** See GeeksForGeeks suggestion for rich features [43] .

108. Food Ordering App (Full-Stack, Intermediate)

- **Stack:** React (front) + Node (backend) + MongoDB.
- **Features:** Browse menu by restaurant, add items to cart, checkout.
- **Backend:** `Restaurant`, `MenuItem`, `Order` models. Endpoints: `/restaurants`, `/restaurants/:id/menu`, `/orders`.
- **Frontend:** Restaurant list page; menu page with "Add to Order"; order summary page. Use Axios to talk to backend APIs.

109. Grocery Delivery App (Full-Stack, Intermediate)

- **Stack:** Similar to #108. Includes search for products, order checkout.
- **Difference:** Might add user addresses, delivery slots.

110. Blogging Platform (Full-Stack, Beginner)

- **Stack:** React + Django (or MERN).
- **Features:** User auth, create/edit posts, view posts by user or category.
- **Backend:** Use Django REST Framework with `Post` and `Comment` models (see Django Girls tutorial [55] for setup).

111. Video Conferencing (Full-Stack, Advanced)

- **Stack:** React front-end, Node backend with WebRTC signaling.
- **Key Parts:** Use WebRTC for media; server handles "rooms" and user signaling (offer/answer).
- **Steps:** Integrate a WebRTC library (like `simple-peer`). Run a signaling server in Node (Socket.IO).

112. Job Search Portal (Full-Stack, Intermediate)

- **Stack:** React (front) + Django (backend).
- **Features:** Employers post jobs (title, description, requirements); users browse/filter/apply.
- **Backend:** `Job`, `User` models. Endpoints: `GET /jobs`, `POST /jobs` (employer), `POST /jobs/:id/apply`.
- **Frontend:** Job listings page with search bar; job detail page with "Apply" button (opens form).

113. Project Management Tool (Full-Stack, Intermediate)

- **Stack:** React + Node.
- **Features:** Create projects, add tasks, assign to users. Drag-and-drop interface for tasks (Kanban).
- **Backend:** `Project`, `Task`, `User` schemas. Endpoints for project/task CRUD.

114. Online Learning Platform (Full-Stack, Advanced)

- **Stack:** React + Node or Django.
- **Features:** Courses with video lessons (hosted on S3 or streaming service), user progress tracking.
- **Backend:** Serve course data, track user lesson completion, handle payments for courses.

115. Chatbot Integration (Full-Stack, Intermediate)

- **Stack:** React front + Node backend.
- **Function:** Frontend sends messages to backend; backend processes with a chatbot (e.g. Dialogflow or simple rule engine) and returns replies.
- **Implementation:** Use REST or websockets for messaging.

116. Social Q&A Forum (Full-Stack, Intermediate)

- **Stack:** Node (backend) + React (frontend).
- **Features:** Ask questions, post answers, upvote. Similar to StackOverflow.

- **Backend:** `Question`, `Answer`, `User` schemas. Endpoints for creating questions/answers and voting.

117. Event Ticketing System (Full-Stack, Intermediate)

- **Stack:** React + Node.
- **Features:** List events (with date/time), select seats or tickets, purchase.
- **Backend:** Event model with available tickets; reduce inventory on order. Use payment API for buying.

118. Food Diary App (Full-Stack, Intermediate)

- **Stack:** React + Node.
- **Features:** Log meals and calories; view daily/weekly summaries.
- **Backend:** `Entry { user, food, calories, date }`.

119. Fitness Tracker (Full-Stack, Intermediate)

- **Stack:** React + Django.
- **Features:** Similar to [78], but with React charts. Users log exercises, view weekly stats.

120. Expense Manager (Full-Stack, Beginner)

- **Stack:** React + Node.
- **Features:** Track spending by category; show charts (Chart.js) of expenses.

121. Multi-Author Blog (Full-Stack, Intermediate)

- **Stack:** Django + React (or MERN).
- **Features:** Several users write posts; comment system. Similar to #110 with auth.

122. Real Estate Listings (Full-Stack, Intermediate)

- **Stack:** React + Node.
- **Features:** Browse houses/apartments with photos. Filter by location/price.
- **Backend:** `Property` schema with address, price, images.

123. Car Rental Service (Full-Stack, Intermediate)

- **Stack:** React + Node.
- **Features:** Select car model and rental dates, book it.
- **Backend:** `Car` model, availability calendar; booking reduces availability.

124. CRM System (Full-Stack, Intermediate)

- **Stack:** React + Django.
- **Features:** Manage contacts, leads, sales pipeline.
- **Backend:** Models for Contacts, Deals, Notes.

125. Appointment Scheduler (Full-Stack, Beginner)

- **Stack:** React + Node.
- **Features:** Users pick a date/time for a service; backend checks conflicts.

126. Auction/Bidding Platform (Full-Stack, Advanced)

- **Stack:** Node + React with Socket.IO.
- **Features:** Users list items; others bid in real time.
- **Backend:** `Item` and `Bid` models. Use websockets to broadcast new highest bid.

127. Event Management App (Full-Stack, Intermediate)

- **Stack:** Node + React.
- **Features:** Create events; attendees RSVP and get email reminders.

128. Music Streaming App (Full-Stack, Advanced)

- **Stack:** React + Node.
- **Features:** Upload or link songs; play them in the browser. Manage playlists.

129. Photo Sharing Network (Full-Stack, Intermediate)

- **Stack:** React + Node.
- **Features:** Upload, like, comment on photos. (Frontend shows grid of images with likes).

130. Professional Network (Full-Stack, Advanced)

- **Stack:** Django + React.
- **Features:** Profiles, network connections, posting updates (like LinkedIn).

131. Travel Planner (Full-Stack, Intermediate)

- **Stack:** React + Node.
- **Features:** Search flights/hotels (using external APIs), save itinerary.

132. Social Q&A Platform (Full-Stack, Advanced)

- **Stack:** Extend #116 with advanced moderation, chat features.

133. Online Auction (Advanced)

- **Stack:** Scale up #126 with payment and high concurrency.

134. Collaborative Editor (Advanced)

- **Stack:** Node + React with CRDT library (e.g. ShareDB).
- **Features:** Real-time multi-user text editing (like Google Docs).

135. Game Leaderboard (Full-Stack, Intermediate)

- **Stack:** Node + React.
- **Features:** Users submit game scores; show top scores and user rank.

136. Portfolio Builder (Full-Stack, Beginner)

- **Stack:** React + Node.
- **Features:** Users input content (projects, bio) via forms; site is generated (could use a template engine).

137. Crowdfunding Platform (Full-Stack, Advanced)

- **Stack:** Django + React.
- **Features:** Campaign pages, pledges, payment processing, goal tracking.

138. Marketplace (Full-Stack, Intermediate)

- **Stack:** Node + React.
- **Features:** Users create “stores” and list items for sale [50] . Buyers browse categories and checkout.

139. Restaurant Reservation (Full-Stack, Intermediate)

- **Stack:** React + Node.
- **Features:** Select restaurant and book a table slot, similar to #125.

140. Weather Dashboard (Full-Stack, Beginner)

- **Stack:** React + Node.
- **Features:** Users can add cities to their dashboard; backend caches and provides weather data.

141. Job Tracker (Full-Stack, Intermediate)

- **Stack:** React + Django.
- **Features:** Track job applications through stages [52] .

142. Health Monitor (Full-Stack, Intermediate)

- **Stack:** React + Node.
- **Features:** Log blood pressure, sugar, etc.; display weekly charts.

143. LMS (Advanced)

- **Stack:** Django + React.
- **Features:** Full course management, user roles, grading.

144. Collaboration Whiteboard (Advanced)

- **Stack:** Node + React.
- **Features:** Draw on a canvas; broadcast strokes via websockets to others in room.

145. Calendar Planner (Full-Stack, Intermediate)

- **Stack:** Node + React.
- **Features:** Shared calendar of events; drag-and-drop event creation.

146. Freelance Marketplace (Advanced)

- **Stack:** Django + React.
- **Features:** Post projects, submit bids (like Upwork). Complex user roles.

147. Crypto Trading Simulator (Advanced)

- **Stack:** Node + React.
- **Features:** Simulate trading with live crypto API prices.

148. Code Submission Platform (Full-Stack, Intermediate)

- **Stack:** Node + React.
- **Features:** Users submit code; backend runs it against testcases (like a simple judge system).

149. Multi-language Blog (Full-Stack, Intermediate)

- **Stack:** Django + React.
- **Features:** Store translations of posts; switch UI language.

150. Educational Games Portal (Full-Stack, Intermediate)

- **Stack:** React + Node.
- **Features:** Host mini-games (math puzzles, quizzes); track user scores.

Code and Resources

For complete code implementations, refer to open-source examples:

- **E-Commerce App:** Basir's MERN tutorial [23†L255-L264] (React + Node + MongoDB) provides a working full app with products, cart, and checkout.
- **React Projects:** Many beginner/intermediate example projects (to-do, weather, blog) are available on GitHub (e.g., s-shemmee/React-Weather-App for a weather app, aaghajani/react-todo-app for a to-do list).
- **Socket.IO Chat:** See SocketGram repository for a real-time React chat app using Socket.IO.
- **Django Examples:** Django Girls tutorial (cited by JetBrains) demonstrates a blog app [55].
- **General:** Search GitHub with tags like *react-todo-app*, *mern-blog*, *django-ecommerce* for starter code.

Each project's implementation involves similar steps: setting up the framework (React/Django), designing models/endpoints, and building components. Use the cited resources as starting points and adapt to the specific features of each project.

Sources: Front-end project ideas from Rohan Fulzele's list [56] [57] and others; back-end ideas from roadmap.sh guide [22] and Fynd Academy [25]; full-stack ideas from GeeksforGeeks [43] [42] and tutorial repositories [54] [38]. Each cited source provides context or examples for the project concepts above.

---

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [46] [56] [57] 50+ Beginner and Intermediate level React Project Ideas ♀ | by Rohan Fulzele | Medium
https://medium.com/@rohan.fulzele/50-beginner-and-intermediate-level-react-project-ideas-%EF%B8%8F-809b396faa39

[22] [23] [24] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [39] [40] [41] 20 Backend Project Ideas to take you from Beginner to Pro
https://roadmap.sh/backend/project-ideas

[25] [26] [27] 30 Backend Projects Ideas in 2025 [with Source Code]
https://www.fynd.academy/blog/backend-project-ideas

[38] GitHub - kelvinleandro/react-chat-websocket: Real-time Chat Application with React (Vite), Socket.IO, and Express
https://github.com/kelvinleandro/react-chat-websocket

[42] [45] [47] [48] [49] [50] [51] 50 Innovative Full-Stack Project Ideas
https://us.greatassignmenthelp.com/blog/full-stack-project-ideas/

[43] [44] [52] 12 Best Full Stack Project Ideas in 2025 | GeeksforGeeks
https://www.geeksforgeeks.org/best-full-stack-project-ideas/

[53] GitHub - s-shemmee/React-Weather-App: A React.js weather app (5 day forecast) using OpenWeatherMap's API + SheCodes weather API. Deployed via Netlify.
https://github.com/s-shemmee/React-Weather-App

[54] GitHub - basir/node-react-ecommerce: Build ECommerce Website Like Amazon By React & Node & MongoDB
https://github.com/basir/node-react-ecommerce

[55] Django Project Ideas | The PyCharm Blog
https://blog.jetbrains.com/pycharm/2024/09/django-project-ideas/