Recruitment Task – Frontend Engineer (React)

Goal of the task

The goal of this task is **not to build a complete video editor**, but to demonstrate:

- how you design frontend architecture,
- how you iterate on code (including the use of AI).

You may (and are encouraged to) use AI tools.
What matters to us is **the process**, not only the final result.

Stack (required)

- **React 18+**
- **Redux / Zustand / React useContext / useState**
  The choice of state management is up to you.
- **TailwindCSS v3+**
- One base component library:
  - shadcn/ui **or**
  - Material UI **or**
  - Ant Design
  - or any other component library
- **Ramda (preferred)** or Lodash
- Asynchronous handling:
  - Promises / async–await
  - *Fluture.js or Effect.js* (if you know them)
- **ECMAScript (no TypeScript)**

Application description

Build a **simplified video editor** that runs entirely on the frontend.

The application **does not operate on real video** — the video preview is a placeholder (e.g. a static image).
The main functionality focuses on the **timeline** and user interaction.

Video preview (placeholder)

- A static image acting as a “player”
- A dynamic label indicating:
    - which clip is currently active
    - or which part of the timeline is selected

Timeline (key part of the task)

The timeline should allow:

- displaying multiple clips on a timeline
- moving the playhead (current time)
- selecting clips
- activating a clip
- splitting a clip into two
- adding new clips

Clips are abstract data objects — they do not contain real video.

Architecture and application state

The way application state is organized is **entirely up to you**.
We only expect the data flow to be well thought out and that you can justify your decisions.

We do not enforce a specific solution — we are interested in **your architectural choices**.

API

The API is provided by us and is **intentionally very flexible**.

For every entity (e.g. project, clip, note):

- the only required field is id
- all remaining content must live under the data field
- **the shape of the data object is not predefined** and should result from your own design decisions

The API acts as a **generic storage layer**, not as a domain-enforcing backend.

REST API – Projects

The REST API is used to store editor projects.

Required operations:

- GET – fetch projects
- POST – create a project
- PUT – update a project
- DELETE – delete a project

A project may include, for example:

- metadata
- timeline data (e.g. clips)

The API will be available in the forked repository along with setup instructions.

GraphQL API – Notes

The GraphQL API is used to handle **notes assigned to a project**.

Required:

- at least one query
- at least one mutation
- UI integration (e.g. notes list, adding a note)

The API will be available in the forked repository along with setup instructions.

Development process (very important)

1. Fork the repository we provide and create a **public GitHub repository**
2. Work iteratively:
    a. **commit every ~5–10 minutes**
    b. use meaningful commit messages describing changes or problems you are currently facing
    Commit even when the code does not work — we care about the process, not only a finished solution.
3. You may use AI:
    a. to generate code
    b. to refactor
    c. to analyze problems

If you used AI, we will ask you at the end to share conversation logs as links
(e.g. ChatGPT provides this feature even in the free version).

It does not matter whether the application is 100% complete — **the process is what counts**.

Deliverables

At the end, provide:

- a link to the GitHub repository
- links to LLM conversations
- a short description in the README explaining:
    - your architectural approach
    - what you would improve with more time

Optional (*nice to have*)

- *Support for multiple selected clips*
- *Undo / redo*
- *Timeline zoom*
- *Use of Fluture.js or Effect.js*
- *Refactoring AI-generated code*
- *Basic tests*
- *Advanced clip operations (e.g. grouping)*

Important

- UI design is not evaluated
- Feature completeness is not evaluated
- We evaluate **the quality of technical decisions**, code structure, and working process

Good luck 🚀