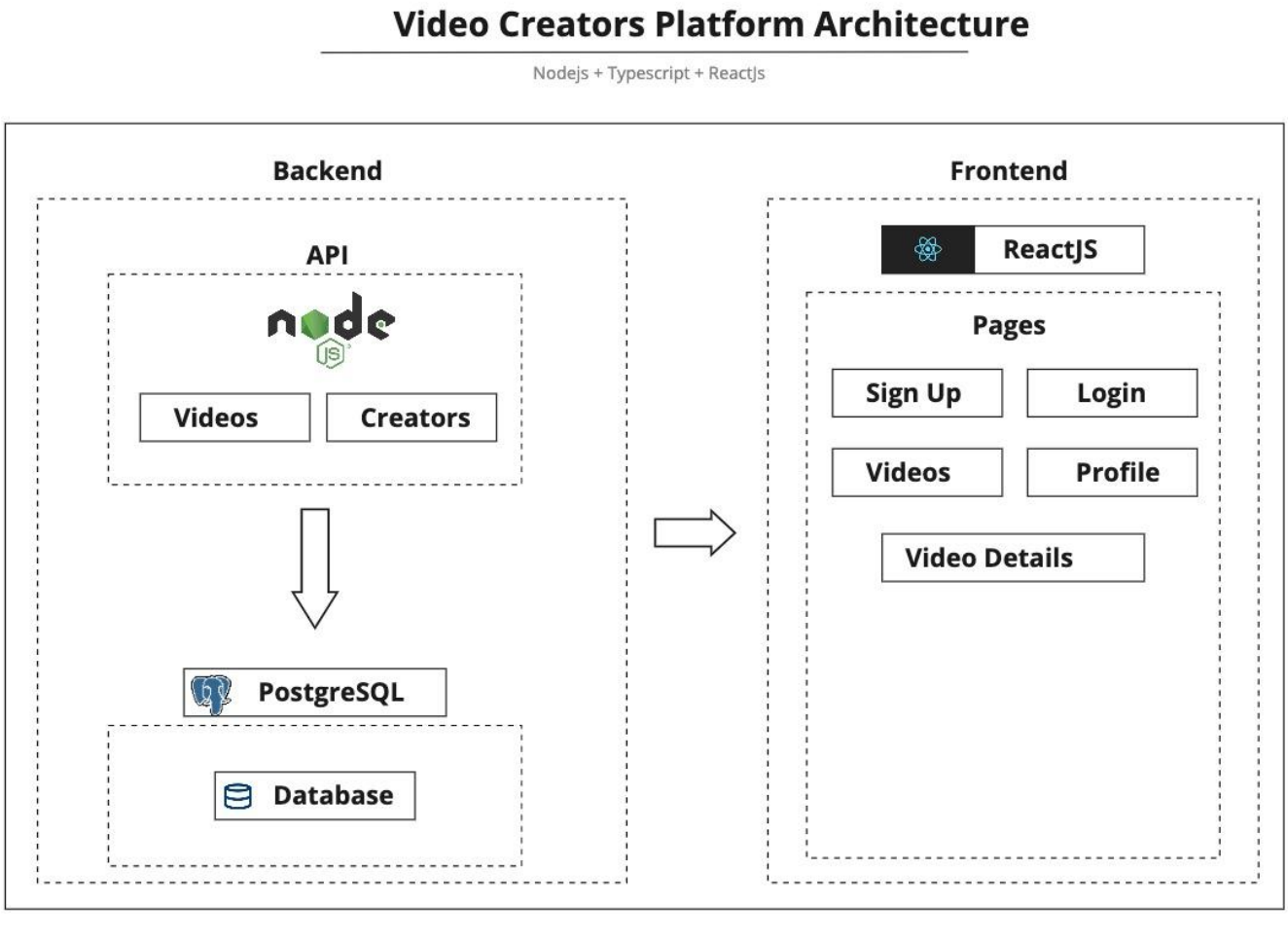
Full Stack Developer Assessment

NodeJS + Typescript + ReactJs

For this assessment we want you to create a full stack application from scratch, creating the backend and frontend independently, and then, integrating both projects. This project is a “Videos Creator Platform” , where new video creators can upload (**video URL**) new videos, sign up, list the available videos and video creators. You’ll have the ability to like videos and follow other video creators.

General Architecture

On this diagram, we give you a general overview of the architecture of this application. We encourage you to take a look at it in order to make any technical decision before you start the assessment, such as entities you’ll need, pages, database connections, etc.



Requirements

Since this is a full stack application, we know this could be a challenge, so we'll try to make the acceptance criteria of the assessment as clear as possible. In the list below you'll find the features we expect you to create for this project. This is the big picture of the features needed to fulfill this assessment, more details will be provided later in this document.

Feature	Description
Sign Up user	The ability to sign up a new user, you can sign up as Student or Teacher. You'll need to provide the name , photo (**URL**), email, password.
Sign In User	Once the user was created, you can sign in with the credentials you used in the sign up process.
List videos availables	List the videos available
Create Video	Create a video, provide basic information about a video, a property **published** should be false by default.
Like Videos	The user should be able to "like" one or more videos.
Publish/Unpublish a Video	The user should be able to publish and unpublish videos
Edit Video Information	The user should be able to update any video information.
Follow/Unfollow Creator	A creator has the ability to follow and unfollow creators.
Initial Bootstrap App	It will be needed that you create a mechanism of initial bootstrap for the app, where there is initial data, you can use migrations, seeders in the backend to load the initial data, such videos, users.

Backend

For the backend application you need to create a Nodejs REST API. Where you will expose all the resources needed by the **CLIENT**. You'll find in the list below the stack you'll need to use:

Tech Stack

- Nodejs/Express
- Typescript
- Nodejs
- JWT
- Database
    - PostgreSQL
    - Any ORM - Sequelize preferred

API Endpoints

In the table below you'll find the resources needed by the **CLIENT** application in order to work properly for the users.

Resource: Name of the Resource
Description: A brief description of the resources is needed.
Additional Constraints: Any extra constraints or information that needs to be considered at the moment the developer creates this resource.
Secured: This indicates if the resource needs to be secured, only authenticated users should be able to access it.

Resource	Description	Additional Constraints	Secured
Sign Up User	Create a creator with the name, email, password.	Emails should not be duplicated. No plain password should be saved into the database. Created the access token for consuming secured resources.	**NO**
Sign In User	Sign In user once the	Created the access token for consuming secured resources.	**NO**
Creator Profile	List the profile information about the creator	List general information + videos created + followers + like videos.	**YES**
Create Video	Create a Video	Provide basic information such as title, creation_date, published **(false by default)**, etc. Developer **SHOULD NOT** develop the full upload feature, for the src of the video, it just needs the **URL**	**YES**
Publish/ Unpublish Video	Publish or unpublish of an existing video	Only set or unset the boolean value for the **publish** property	**YES**
List Videos	List the videos available	List all published videos	**YES**
Like Videos	List the videos a creator has liked.	Creators can like their own videos as well	**YES**
Video Details	List the details of a video	N/A	**YES**
Edit Video	Update video created	**Publish** property should be ignored for updating this resource since there's an endpoint for publishing/unpublishing the video.	**YES**
Follow Unfollow Creator	Follow/Unfollow creators	N/A	**YES**

Initial Bootstrap App	Provide initial data for bootstraping the application, this could be managed by migrations and seeders	****NOTE**** This is **NOT** an endpoint to be created, this could be a script using the CLI.	**N/A**

****Please remember you'll need to make any technical decision around the entities and the properties for them. ****

Some stuff to keep in mind

- Make use of linters, formatters for the code.
- Follow coding principles for creating clean code
- Follow best practices for the framework/tools you are using.
- Repository should be public for downloading after the developer submits the proposal.
- Good README.md file for setting up the repository locally and testing it.

Some extra bonus points

- Renew Access Token
- Security Policies (**CORS**)
- API Documentation

- Use docker for the database (only in dev mode)
- For the **follow creator** endpoint, it would be an extra point if the developer is capable of implementing a pub/sub mechanism, **only** when a creator follows another creator triggers a "notification" through a terminal level.
- Deployment
- Unit Testing & Integration Testing
- Docker implementation for the database

Frontend

For the frontend application the developer needs to create a ReactJS APP. Where the app is capable of consuming all the resources created in the **BACKEND**. You'll find in the list below the stack you'll need to use.

Tech Stack

- ReactJS

- Typescript
- Any CSS Library - Material UI Preferred

In the table below you'll find the pages/views needed for the **CLIENT** application to work properly for the users.

Page: Name of the Page
Description: A brief description of the resources is needed.
Additional Constraints: Any extra constraints or information that needs to be considered at the moment the developer creates these pages/views.
Secured: This indicates if the page/view needs to be secured, only authenticated users should be able to access it.

Page	Description	Additional Constraints	Secured
Sign Up	This page should be capable of creating a new user of the videos creators platform	This should be a form where the creators are capable or signing up Name, Email, Password, should be provided. Display any error if the operation returned any error.	**NO**
Sign In	This page should be capable of signing in a creator on the platform.	Email / Password should be provided. Display any error if the operation returned any error. After successfully logged in, it should redirect to the Videos Page.	**NO**
Videos	Display the available videos for the user	Display all the videos available. Creators should see an option for creating a video, the developer has the freedom if he/she wants to create a separate page or in the same page.	**YES**
Video Details	Display all the details of a selected video	The developer should feel free of the UI/UX for this page.	**YES**
Creator Profile	Display the information about the current creator returned from the backend.	The developer should feel free of the UI/UX for this page.	**YES**

Some stuff to keep in mind

- Make use of linters, formatters for the code.
- You can use any ReactJS flavor you want.
- You can use any CSS Library for getting this CLIENT application up and running.
- Follow coding principles for creating clean code
- Follow best practices for the framework/tools you are using.

- Repository should be public for downloading after the developer submits the proposal.
- Good README.md file for setting up the repository locally and testing it.

Some extra bonus points

- Responsive Design
- Deployment - Connected to the backend created.
- Unit Testing & Integration Testing

Additional Notes

- [ ] Provide the repository **URLs** for these projects, **DO NOT** send zip files.
- [ ] Provide **URLs** for the deployments if the developer got it done.
- [ ] Give more priority to the **BACKEND** implementation, if the developer gets the two of them, that would be awesome.