Upcoming Movies Web App

PROJECT DESCRIPTION

As a full stack software developer you've been tasked with the development of an webapp for cinephiles and movie hobbyists. This first version (MVP) of the app will be very simple and limited to showing the list of upcoming movies. The app will be fed with content from The Movie Database (TMDb). No design specs were given, so you're free to follow your UX and UI personal preferences. The front-end for this app should be simple, so try to keep all the business logic on the back-end, including access to the TMDb API. As mentioned before, all data must come from TMDb, but feel free to create your own data storage strategy.

FUNCTIONAL REQUIREMENTS

The first release of the app will be very limited in scope, but will serve as the foundation for future releases. It's expected that user will be able to:

- See a list of upcoming movies - including the movies' name, poster or backdrop image, genre and release date. **The list should not be limited to only the first 20 movies as returned by the API.**
- Select a specific movie to see its details (name, poster image, genre, overview and release date).
- Search for movies by entering a partial or full movie name.

TECHNICAL REQUIREMENTS

You should see this project as an opportunity to create an app following modern development best practices , but also feel free to use your own app architecture preferences (coding standards, code organization, third-party libraries, etc).

A **TMDb API** key is already available so you don't need to request your own:
`1f54bd990f1cdfb230adb312546d765d`

The API documentation and examples of use can be found here:
https://developers.themoviedb.org/3

- You can use any combination of frontend and backend technology
- You should create your **own backend API layer**, which will be responsible to send requests to the TMDb API
- The frontend app should request data **only from your backend API**
- Feel free to use any package/dependency managers if you see fit

DELIVERABLES

The project's source code and dependencies should be made available in GitHub. Here are the steps you should follow:

1. Create a public repository on GitHub (create an account if you don't have one).
2. Create a master branch, where you will keep all of your boilerplate code. Then create a development branch, which should only contain code written by you. Also, open a Pull Request from Development into Master, which we'll use to review your code challenge. **It's very important that you DO NOT include boilerplate code on your PR.**
3. Include a README file that describes:
    - Your architecture (how did you approach the problem?)
    - Assumptions that you have made, if any
    - Special build instructions, if any
    - List of third-party libraries used and short description of why/how they were used
4. Once the work is complete, create a pull request from "development" into "master" and send us the link. We will review your code and might ask you some questions directly on your pull request.
1. You must provide us with a URL to access a running instance of your application. In order to do that, you can use any free hosting services available online (e.g. Heroku - https://devcenter.heroku.com/).

IMPORTANT

1. Keep in mind that we will only evaluate the code created by you, so the pull request should contain only the code written by the candidate. Any boilerplate code provided by frameworks, libraries or code generators should not be present in this pull request. **Projects sent with boilerplate code or unnecessary files on the PR will be disqualified. Code should be as clean as possible to facilitate review.**
2. In this trial you should have a PR from development branch to master. **Projects sent without a PR will be disqualified. We suggest you spend some time looking through the PR yourself to make sure you're not including unnecessary files missing anything important. This is how the reviewer will see your code.**

NOTES

Here at ArcTouch we're big believers of collective code ownership, so remember that you're writing code that will be reviewed, tested and maintained by other developers. Things to keep in mind:

- First of all, it should compile and run without errors
- Be as clean and consistent as possible
- Despite the project's simplicity, don't ignore development and architecture best practices.

It's expected that you choose a code architecture that encourages clear separation of concerns and supports project growth, but try not to over engineer. You should try to balance project simplicity and yet demonstrate you care about and understand code architecture best practices.

This project description is intentionally vague in some aspects, but if you need assistance feel free to ask for help. We wish you good luck!