
ADMIOS
Technical Challenge

AdmiosFlix is a company that manages movie titles!
We all love movies, so this should be fun homework.

In this challenge you will help AdmiosFlix by building a movie management system which allows users to view, search, add, and remove movies from a database.

Core Feature Requirements

- Users can see all movies on the application's home screen
- Movies are organized alphabetically and by genre
- Users can see movies details when a movie is selected
  - Rating
  - Genre
  - Cast & Crew
  - Release Date

Optional (Bonus) Feature Requirements

- Support paging if there are too many items

Please feel free to use whatever frameworks and languages you feel most comfortable with.

If you're using advanced techniques or interesting tools and frameworks please add notes in the readme file so we can easily find and evaluate them. This will count as an additional bonus in your evaluation.

We at Admios, as well as our customers, have very high standards for software engineering, so please do your best possible work here, to impress us and them.

Core Expectations

- **Functionality:** ideally the application should behave according to the feature requirements and contain no critical bugs.
- **Clean Code:** Good coding style, naming conventions, readability, easy to debug and most importantly to extend and add new features.
- **System Design:** be as thoughtful as you can designing your architecture.
- **Testability:** all the main features should be covered with automated unit tests or integration tests.
- **Documentation:** describe architecture, technologies you used in your solutions, startup scripts, and any additional steps required to run and test your application.
- **Programming Language:** ideally the language you're most proficient in.

Choose your path, based on the job you're applying for, Front End, Back End, Full Stack, Mobile or DevOps.

If you feel you want to venture on a different path, you may do so, and please read carefully all the specific requirements for the role you have chosen.

Good luck on your journey!

Your mission is to build a web application that implements the Core Feature Requirements listed above. And your challenge is to use our Admios brand look and feel to style your site. This will demonstrate your ability to conform to CSS and style guidelines. For reference on what our brand looks like, please check out www.admios.com

Of course do not let us limit your ability to be creative, and feel free to surprise us with an original design.

In addition to the Core Expectations, please also consider the following:

Expectations

- Fake API: you do not need to build an actual API for the Front End challenge. Show us how you can fake an API using a reasonable JSON schema for the responses
- Well designed UI, rendering smoothly and efficiently
- Using one of the popular Front End frameworks (React, Vue, Angular, Svelte)
- Responsive Design with attention to sound UX guidelines

Back End

Your mission is to ONLY build an API that supports all the Core Feature Requirements listed above, and design a database that is used to store the movies data. In addition, the API must allow users to:

- Create/update/delete a movie
- Filter movies by Genre

In addition to the Core Expectations, please also consider the following:

Expectations

- Well designed API either using RESTful endpoints or a single GraphQL endpoint
- Proper modeling of business domain and domain objects
- Design a database for storing and retrieving information; can be a relational SQL database or non-relational document based store
- Documentation for all API endpoints using Swagger or a similar tool
- Using a proper framework for data access such as ORM/ODM
- Detailed information and any required script for database creation

- Implement an authentication solution for the API endpoints; you may choose any Identity Provider implementation and any authentication scheme you want (i.e. Basic, Token Bearer, etc.)
- The solution runs on Docker

Full Stack

You are a master of both worlds! Your mission is to build an API and a Web Application that implement the Core Feature Requirements listed above.

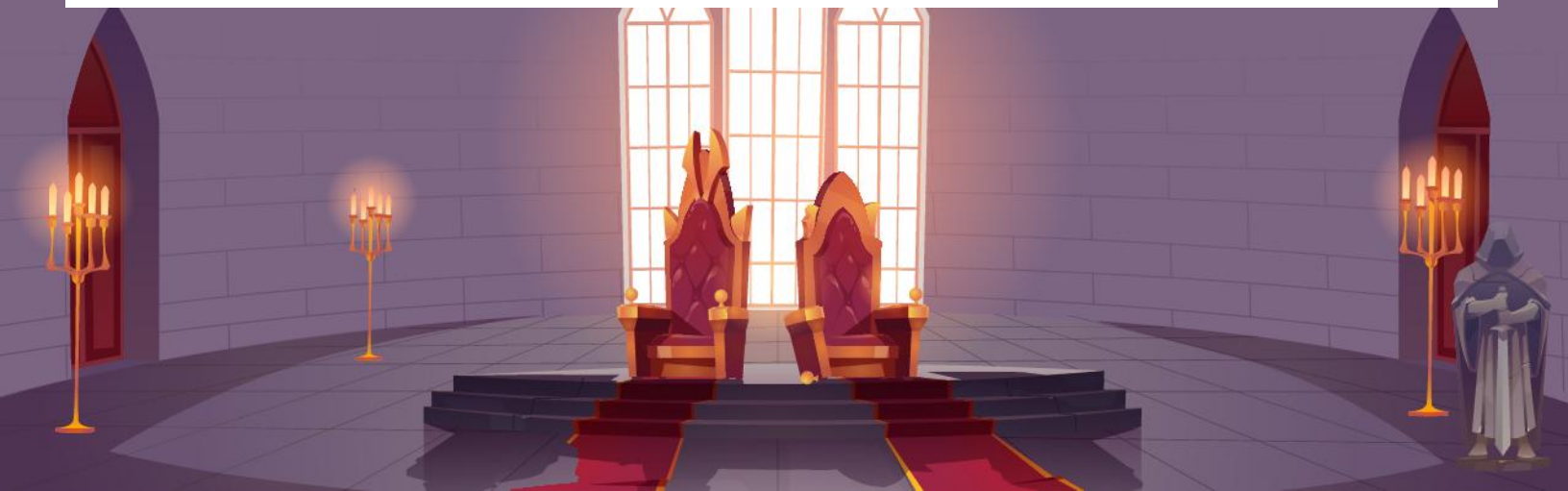
For the Back End, please follow the coding and documentation guidelines and expectations in the Back End section above. In order to make this challenge feasible in the shortest possible time, the API requirements will be simplified to the following endpoints:

- Get all movies
- Create a movie

For the Front End please follow the coding and design guidelines and expectations in the Front End section above. Also, the Front End will be simplified to only accommodate the back end requirements for this full stack challenge, so your web application needs to have as a minimum:

- A page that displays all movies ordered alphabetically and grouped by genre.
- A form for creating a movie in the database using the API endpoint.

In addition to the Core Expectations, please also consider the following:



Full Stack

Expectations

- Implement an authentication solution for the web application; you may choose any Identity Provider implementation (or 3rd party) and any authentication scheme you want (i.e. Basic, Token Bearer, etc.)
- The solution runs on Docker

Bonus

(Optional) Features

Use a cloud provider to create a CI/CD pipeline and deploy the application

Mobile

Your mission is to build a mobile application that implements the Core Feature Requirements listed above. And your challenge is to use our Admios brand look and feel to style your site. For reference on what our brand looks like, please check out www.admios.com

In addition to the Core Expectations, please also consider the following:

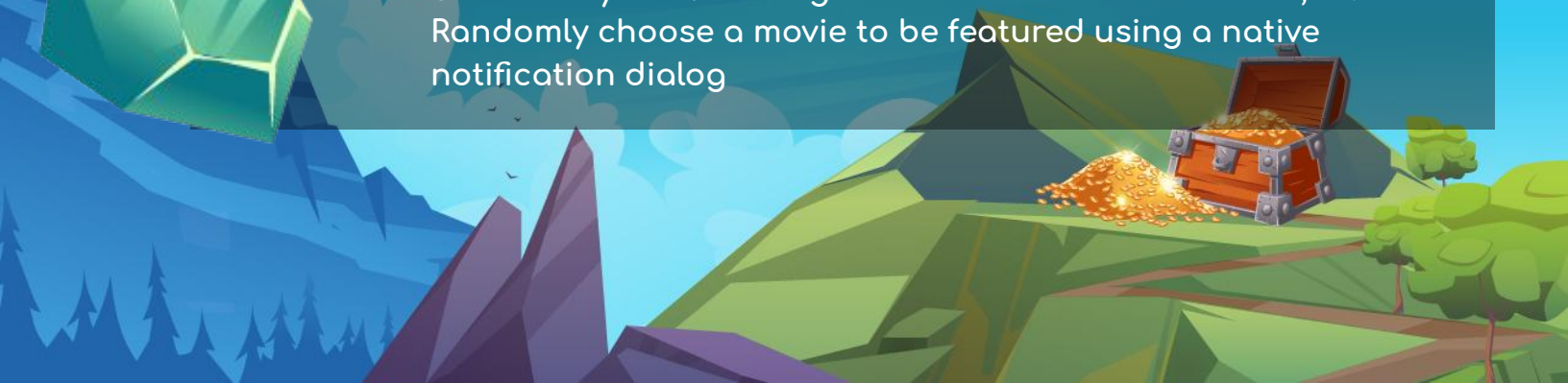
Expectations

- Fake API: you do not need to build an actual API for the Mobile challenge. Show us how you can fake an API using a reasonable JSON schema for the responses
- Well designed UI, rendering smoothly and efficiently
- Using one of the mobile frameworks (iOS native, React Native, Android)
- Conformance with the platform specific (i.e. iOS or Android) design guidelines
- Offline mode; choose an approach to guarantee a minimal set of functionalities in offline mode.

Bonus

(Optional) Features

- Implement an AdmiosFlix Movie Widget; you may choose what functionality that widget offers
- Showcase your knowledge of local device notifications; ex. Randomly choose a movie to be featured using a native notification dialog



DevOps

Your mission is to design and document a cloud architecture that will host the AdmiosFlix applications. You work closely with the software development team at AdmiosFlix, and the current application architecture is as follows:

- Web API that exposes endpoint to retrieve and update movie information
- A Queue that stores the 10 most recently added movies
- Cloud email service that periodically sends out emails to subscribed users with the latest movies added to the system (read from the Queue)
- A database that stores movies related data
- A Single Page Application frontend accessible to end users

Please use **only diagrams and documentation** to implement the following:

Expectations

- Select the cloud provider of your choice
- Using clear and detailed diagrams, describe the cloud architecture to host all the software components of the AdmiosFlix application
- Recommend one option for a cloud based database solution to be used to store the movies data and explain your rationale for the design approach
- Demonstrate understanding of Load Balancing, security and describe your approach and implementation
- Recommend an approach for scalability and resiliency
- Demonstrate how you would design a CI/CD that deploys all the aforementioned artifacts

DevOps

Bonus

(Optional) Features

- Use Docker and / or container orchestration, ECS, Fargate, EKS, Kubernetes, etc...
- Explain how you would manage and instrument the application (i.e. logs aggregation and application metrics and insights)