**High-Level Design Document (HLD)**

[**Movie Recommendation Application**](https://moviefilx.netlify.app/home)

***~ By Yasir Y. Lambawala***

# Document Version Control

| **Date Issued** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 31-01-23 | 1.0 | Initial HLD — V1.0 | Yasir Y. Lambawala |

**Table of Contents**

[**Document Version Control**](#_xeb1uu61no6d) **2**

[**Abstract**](#_py1bew6kjbs5) **4**

[Project links:](#_9iio9myx0na4) 4

[**1 Introduction**](#_2t5dqle1eyof) **5**

[1.1 Why this High-Level Design Document?](#_iqleihioxcje) 5

[1.2 Scope](#_4guauncjp9g) 5

[1.3 Definitions](#_d1aqbn8t4405) 5

[**2 General Description**](#_wukenywb0ox) **6**

[2.2 Problem statement](#_lfyzmrdgfnw6) 6

[2.4 Project Requirements](#_5cscwg2q59hg) 6

[2.5 Technical Requirements](#_cpbwnbqo1alj) 6

[2.6 Tools used](#_q5alqdlnuy4f) 7

[2.6 Hardware Requirements](#_c3hhuy83c1at) 8

[**3 Design Details**](#_3b9a9lcfisnx) **9**

[3.1 Process flow](#_c61goyhxzdzw) 9

[3.2 Backend Design](#_lro9yo1ty0yw) 10

[3.3 Frontend Design](#_gfohfq4l8y85) 13

[**4 Performance**](#_7qx3jcrbpqgr) **16**

[4.1 Reusability](#_ec9wdb2u4ahb) 16

[4.2 Accessibility](#_dcjm86n2dkbo) 16

[4.3 Best practices](#_e1e7xcl0i97) 16

[4.4 Deployment](#_uh70fkybgm3g) 17

[5 Features](#_fjbg9738i1hd) 18

[**6 Conclusion**](#_a5b23k39vk8a) **19**

# 

# Abstract

This is a report on an iNeuron internship project. It was a frontend project but I didn’t find a proper API for the project so I decided to make the backend also. The project is the “Movie Recommendation Application” which recommends movies based on the selected categories.

The application is named as “Movieflix”. The project was created to practice the skills I learned in the iNeuron Fullstack JavaScript Web Development Bootcamp.

This report is a detailed representation of the internship project with project description, technologies used to make this project, process flow of this project, and project features.

## Project links:

* [**Github Frontend Link**](https://github.com/Yasir284/Movieflix_Internship_project_FRONTEND)
* [**Github Backend Link**](https://github.com/Yasir284/Movieflix_Internship_Project_Backend)
* [**Live Project Link**](https://moviefilx.netlify.app/home)

# 

# 1 Introduction

## 1.1 Why this High-Level Design Document?

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding and can be used as a reference manual for how the modules interact at a high level.

The HLD will:

* Present all of the design aspects and define them in detail
* Describe the user interface being implemented
* Describe the hardware and software interfaces
* Describe the performance requirements
* Include design features and the architecture of the project

## **1.2** Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

## 1.3 Definitions

| **Term** | **Description** |
| --- | --- |
| JS | Javascript |
| CSS | Cascading Stylesheet |
| HTML | Hyper Text Markup Language |
| API | Application Programming Interface |

# 

# **2** General Description

**2.1 Product Perspective**

Design a “Movie Recommendation Application based on category” where the users can get movie recommendations based on the category.

## **2.2** Problem statement

An application that recommends movies to users based on some criteria is a Movie

Recommendation Application.

Now that you understand what a Movie Recommendation Application is, let’s discuss some of the “Movie Recommendation Application based on category” functionality you will design.

1. Design a homepage where users can see the available categories.
2. Design a page where the recommendations based on the category can be shown to the user.

**2.3 IMPROVEMENTS**

This was a frontend project but, it is further improved by adding the backend also which can get movies based on selected categories, search for movies, add movies, delete movies, update movies, and also added user login, logout, and signup.

## 2.4 Project Requirements

In this project we need data on movies, based on the selected categories.

* Data is fetched from the backend APIs
* Movies list is obtained from the MongoDB database.

## 2.5 Technical Requirements

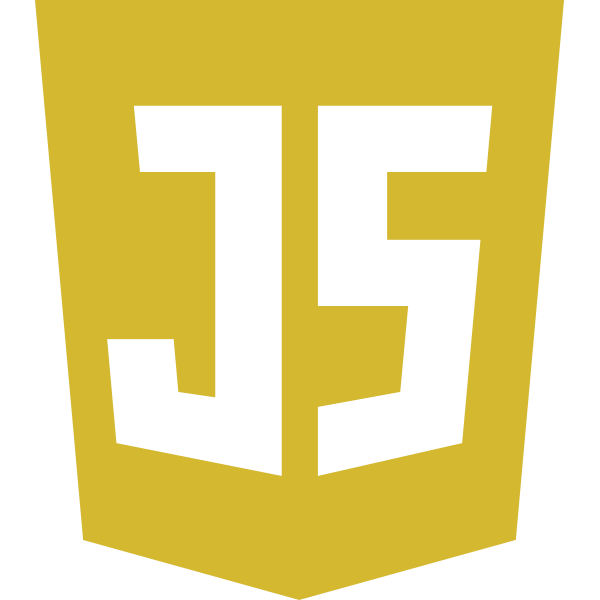
As this project is just a web application, there are some requirements for the project which are as follows:

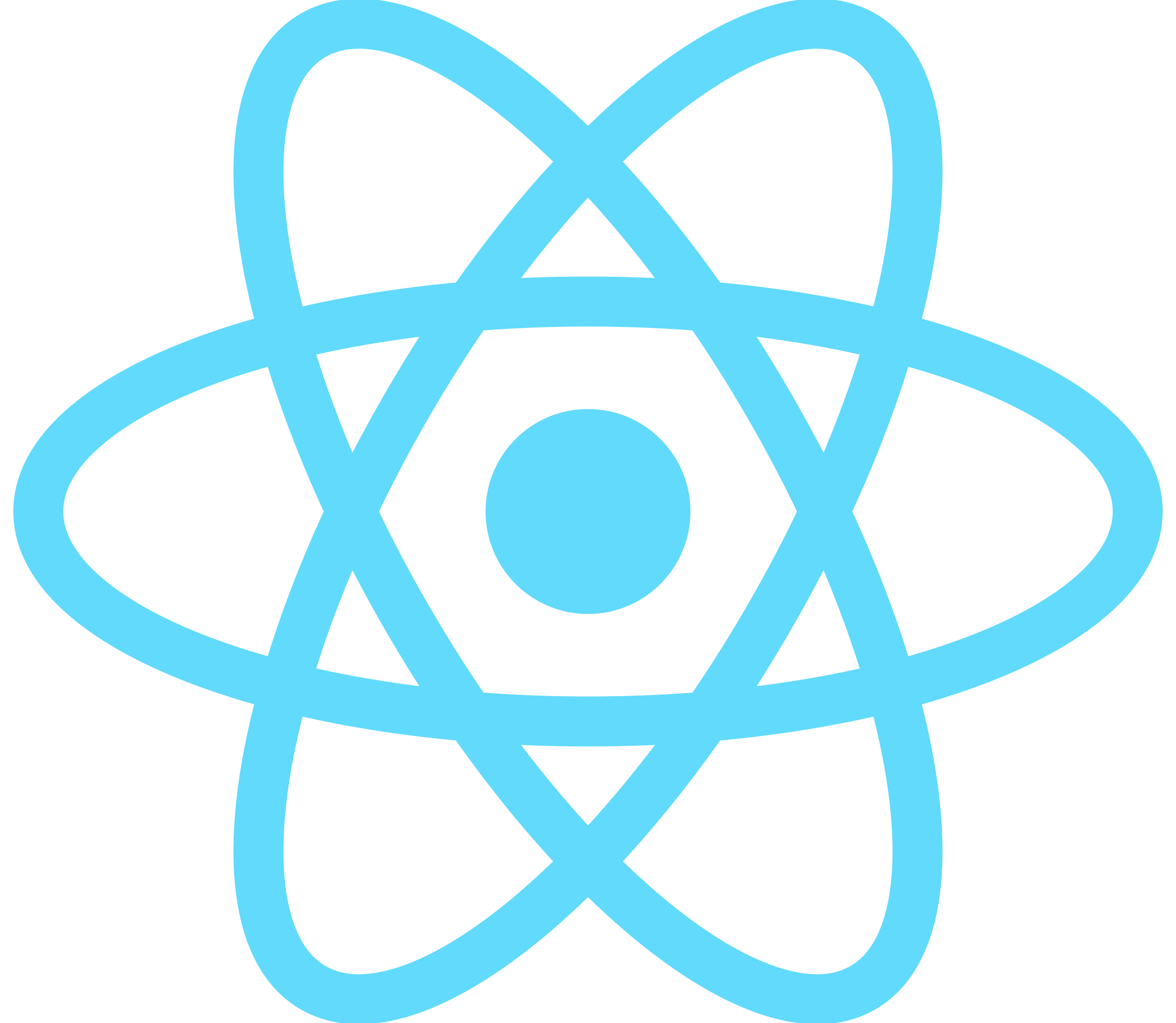
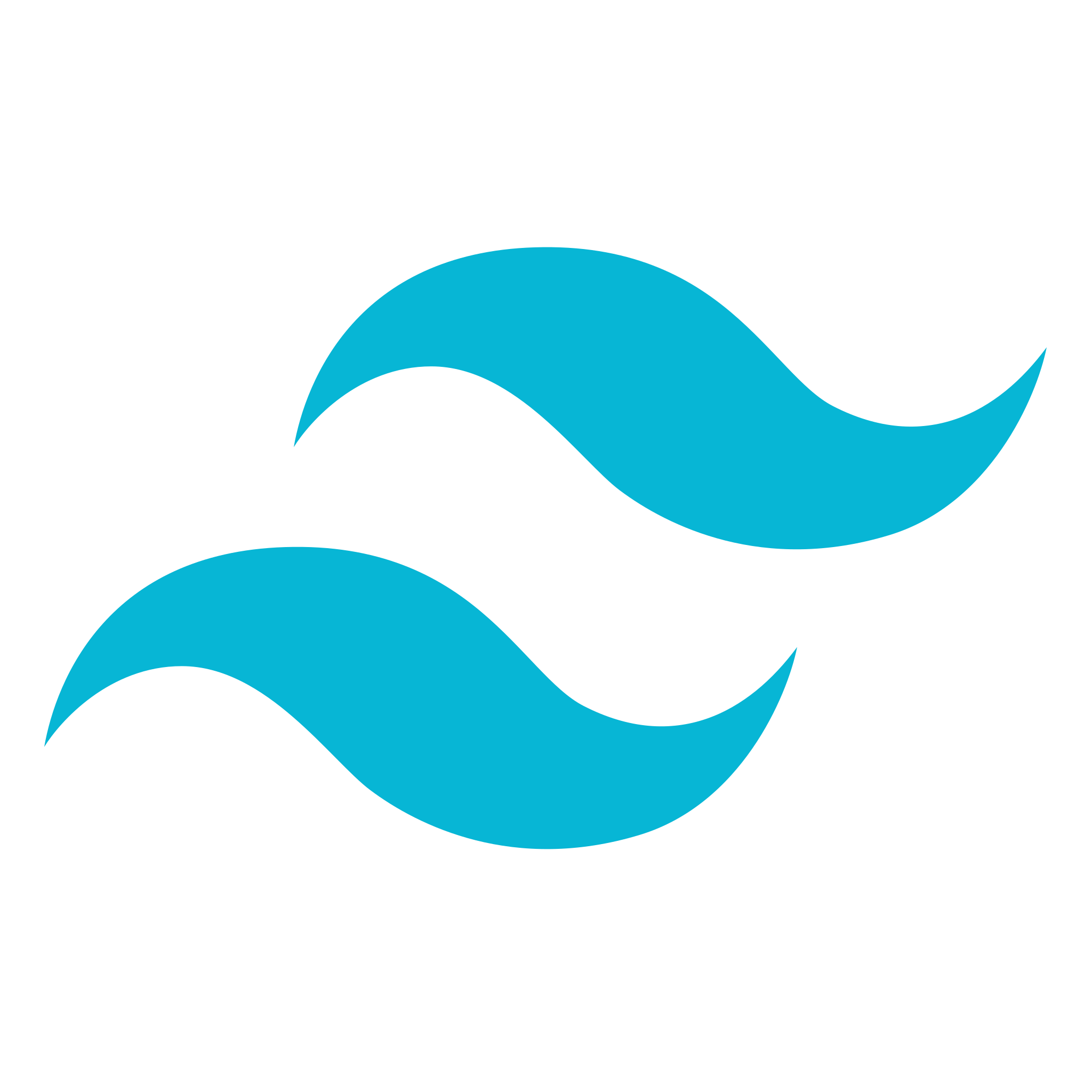
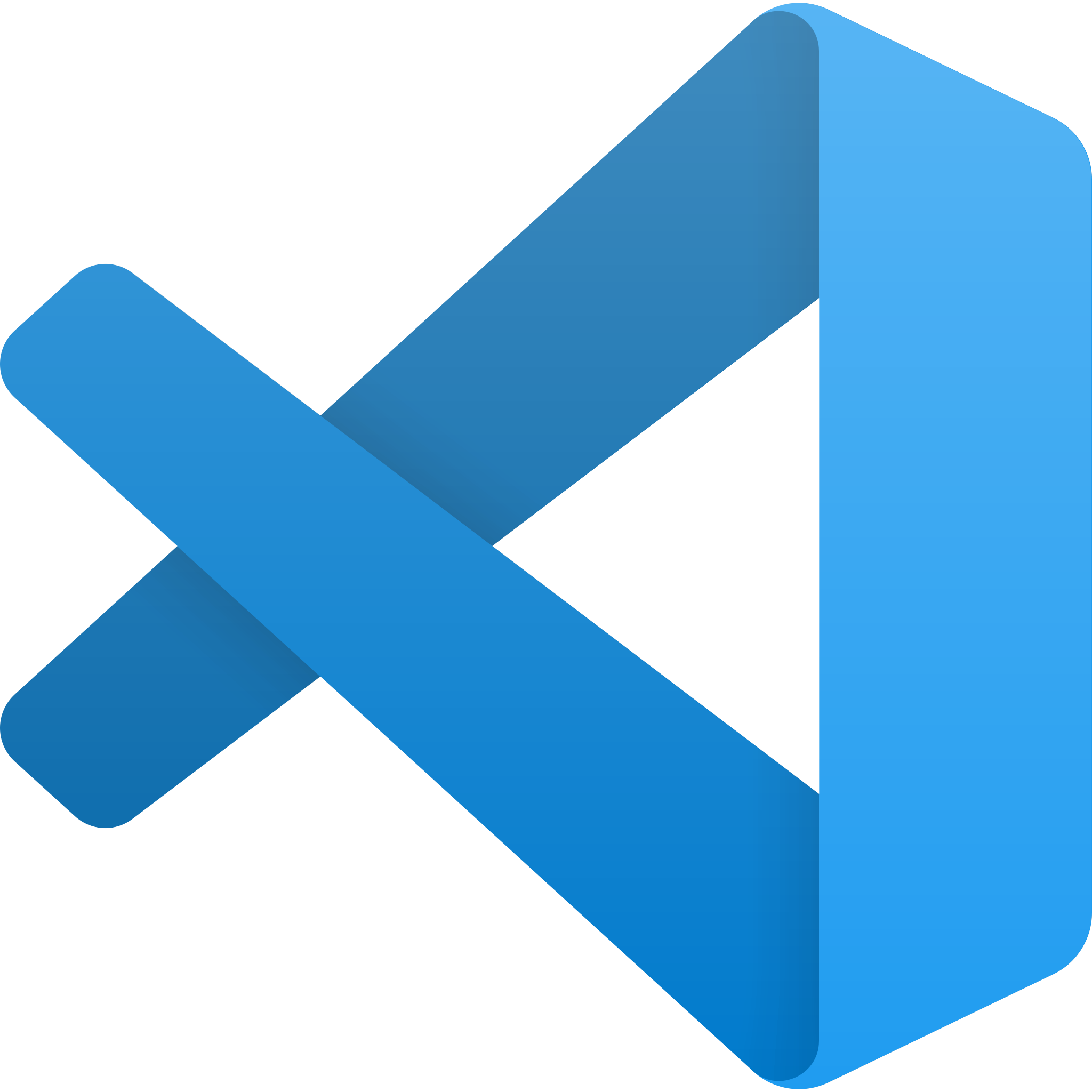
* Mobile Responsive Design
* Website Hosting Service
* Website Page Load Time
* Website accessibility
* Use best practices in code

## 2.6 Tools used

The tools used in making this application are as follows:

* Language - JavaScript
* Frontend library - React
* CSS framework - Tailwind
* Database - MongoDB & Mongoose
* Backend - Node.js & Express
* Animation library - Framer motion

* React is used to build the frontend of the application.
* Tailwind is used as a CSS framework.
* MongoDB is used as a database.
* Cloudinary is used to store the images.
* Node.js and Express are used to build the backend.
* Framer motion is used to add some animations.
* VS code is used as a code editor

## 2.6 Hardware Requirements

Here are some system requirements for using the application:

|  | **Windows requirements** | **Mac requirements** | **Linux requirements** |
| --- | --- | --- | --- |
| **Operating system** | Windows 7 or later | macOS High Sierra 10.13 or later | 64-bit  Ubuntu 14.04+,  Debian 8+,  openSUSE 13.3+, or  Fedora Linux 24+ |
| **Processor** | Intel Pentium 4 or later | Intel | Intel Pentium 4 or later |
| **Memory** | 2 GB minimum, 4 GB recommended | | |
| **Screen resolution** | 1280x1024 or larger | | |
| **Application window size** | 1024x680 or larger | | |
| **Internet connection** | Required | | |

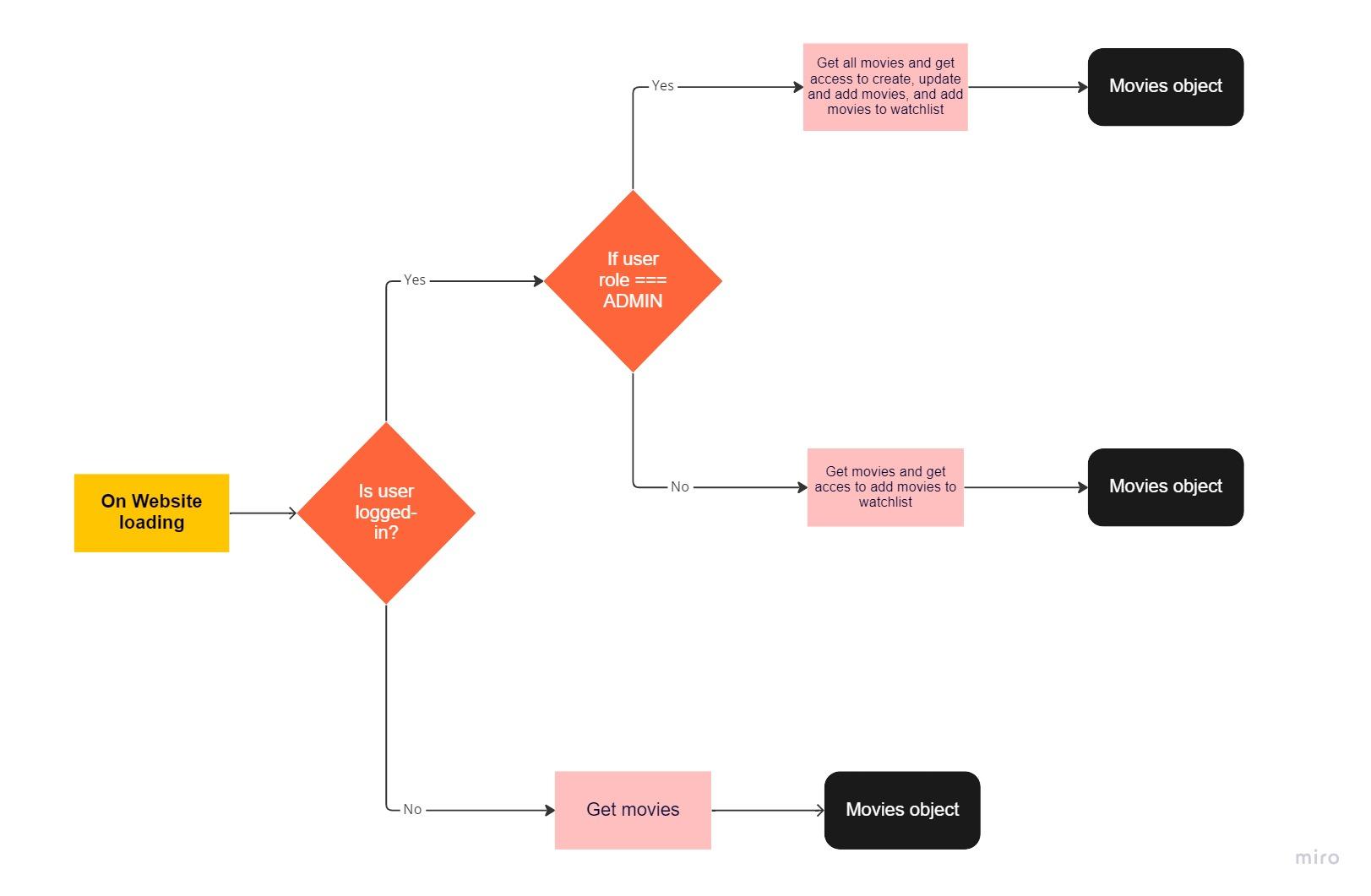
# 

# **3** Design Details

## 3.1 Process flow

When the site loads we get all the movie data from the backend and when the client selects any category of movies, then we query movies based on the categories.

**Process when the client loads the site**



## 

## **3.**2Backend Design

The backend is made with Node.js, Mongoose, and Express, and MongoDB is used as a database. The backend is deployed on render.

The backend is structured as follows:

1. **Model:**
   1. **movie.schema**

contains attributes such as;

* category
* name
* rating
* trailerUrl
* photo
* streamingPlatform
* description
* wishlist
  1. **user.schema**

contains attributes such as;

* name
* email
* password

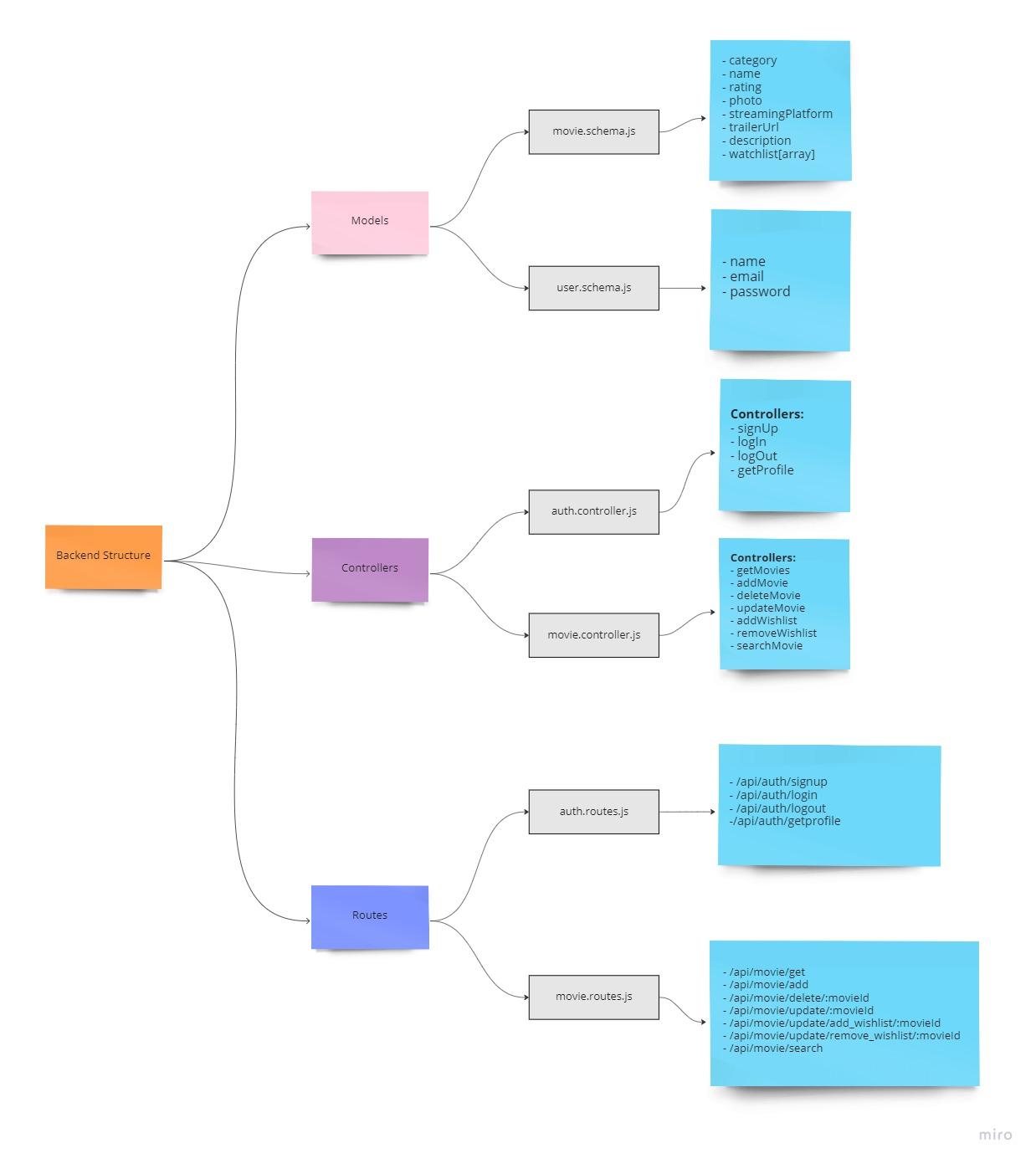
1. **Controllers:**
   1. **auth.controllers**

* Signup controller
* Login controller
* Logout controller
* Get profile controller
  1. **movie.controllers**
* Get movies controller
* Add movie controller
* Delete movie controller
* Update movie controller
* Add to wishlist controller
* Remove from wishlist controller
* Search movies controller

1. **Routes:**
   1. **auth.routes**

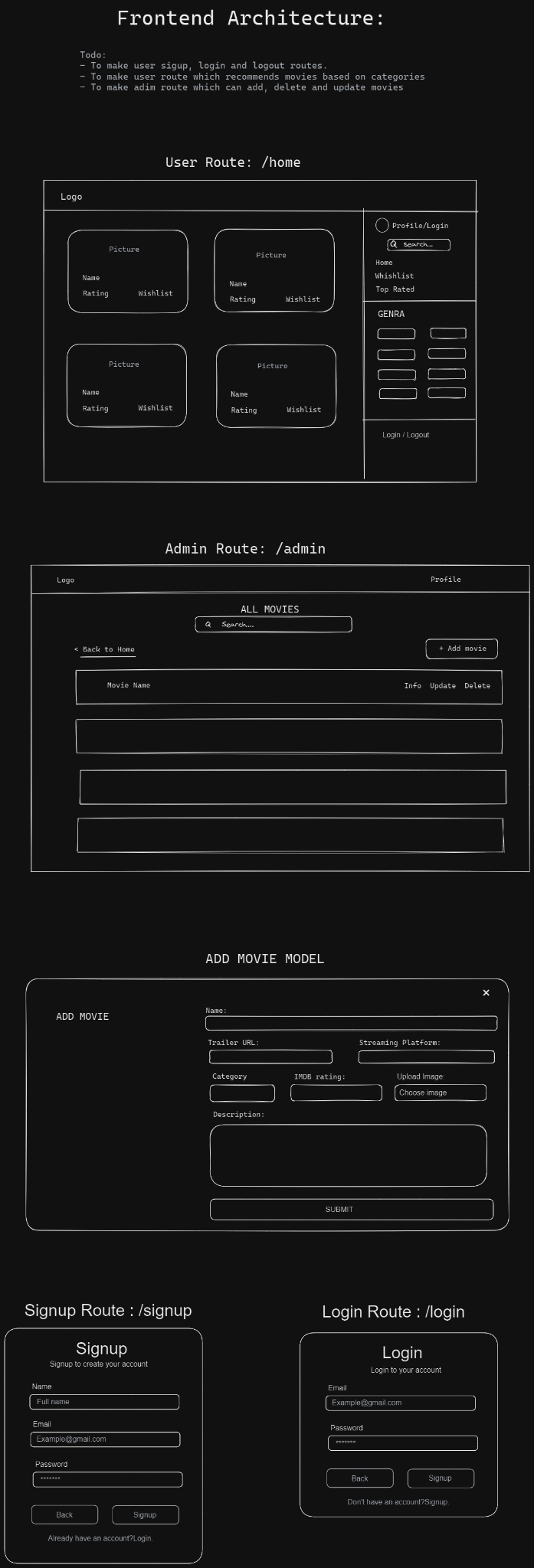
* /api/auth/signup
* /api/auth/login
* /api/auth/logout
* /api/auth/getprofile
  1. **movie.routes**
* /api/movie/get
* /api/movie/add
* /api/movie/delete/:movieId
* /api/movie/update/:movieId
* /api/movie/update/add\_wishlist/:movieId
* /api/movie/update/remove\_wishlist/:movieId
* /api/movie/search

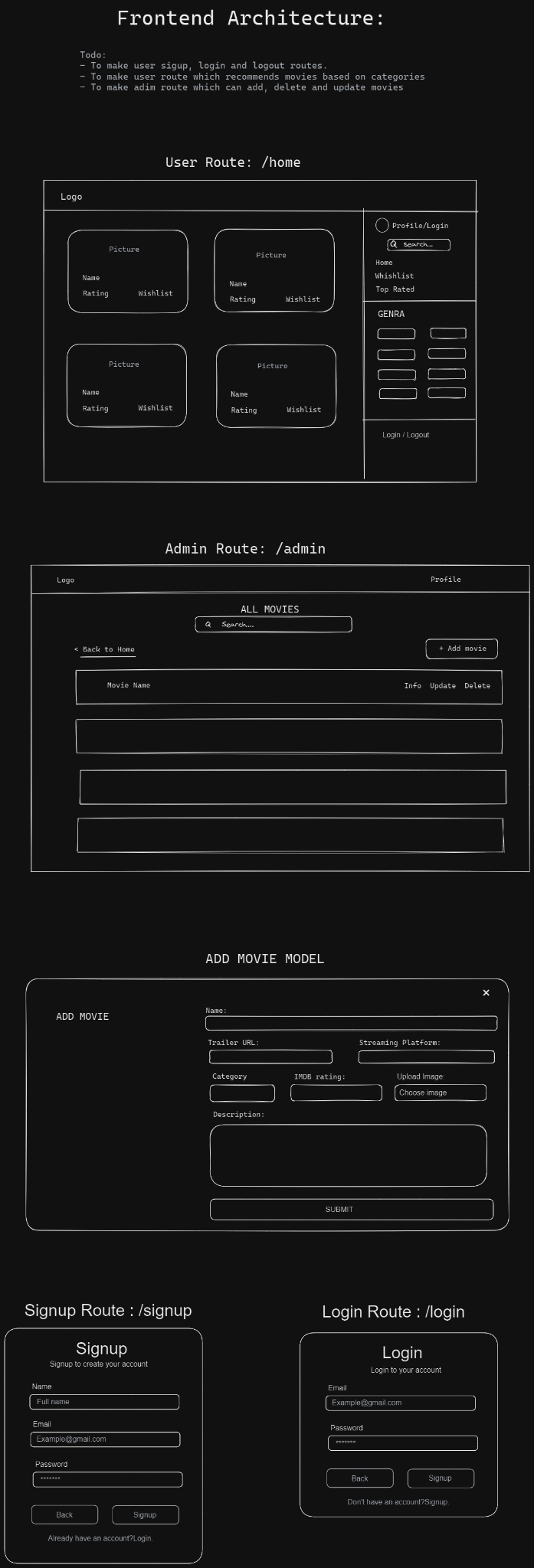
**Backend Planning**

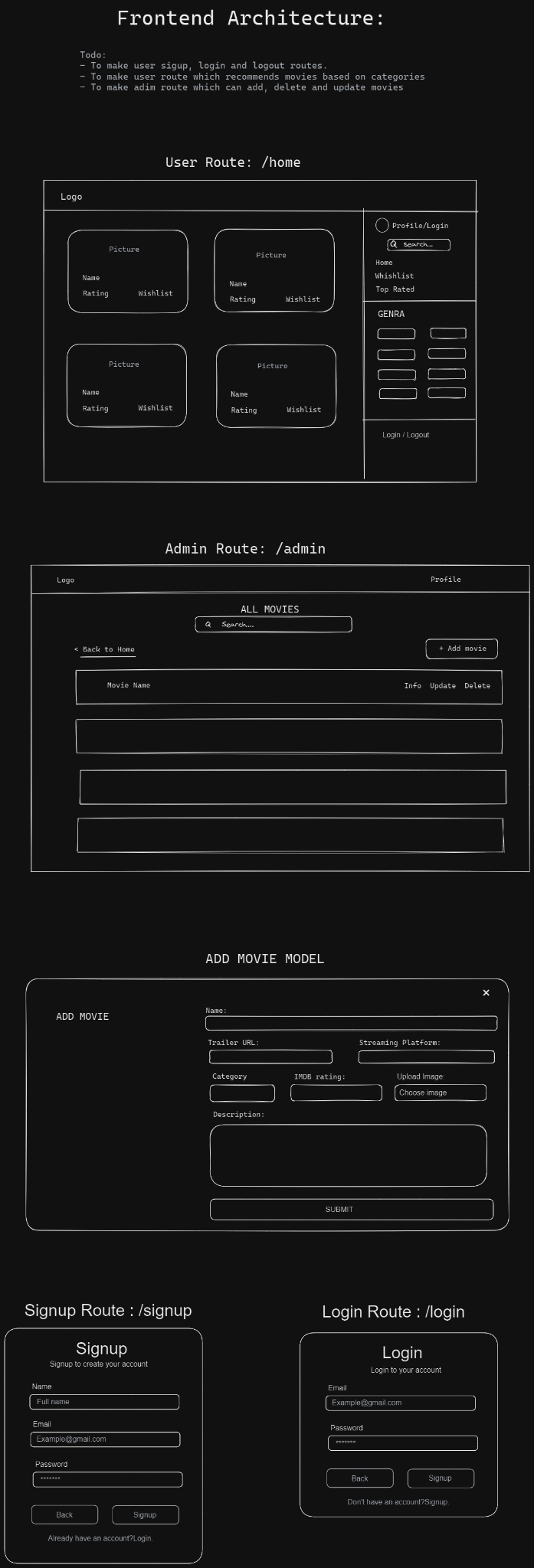


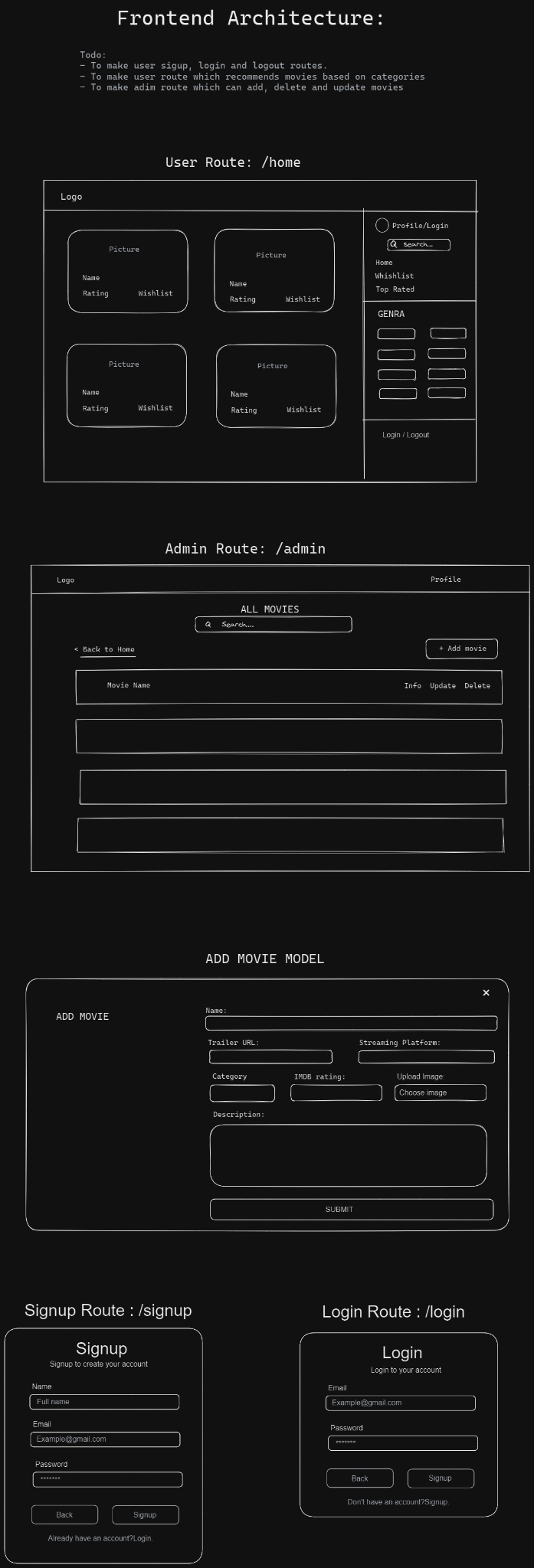
## 3.3 Frontend Design

The frontend is made with React and tailwind and framer motion are used to add some animation. The frontend is deployed on Netlify. Below is the frontend rough design:









The frontend contains the following routes:

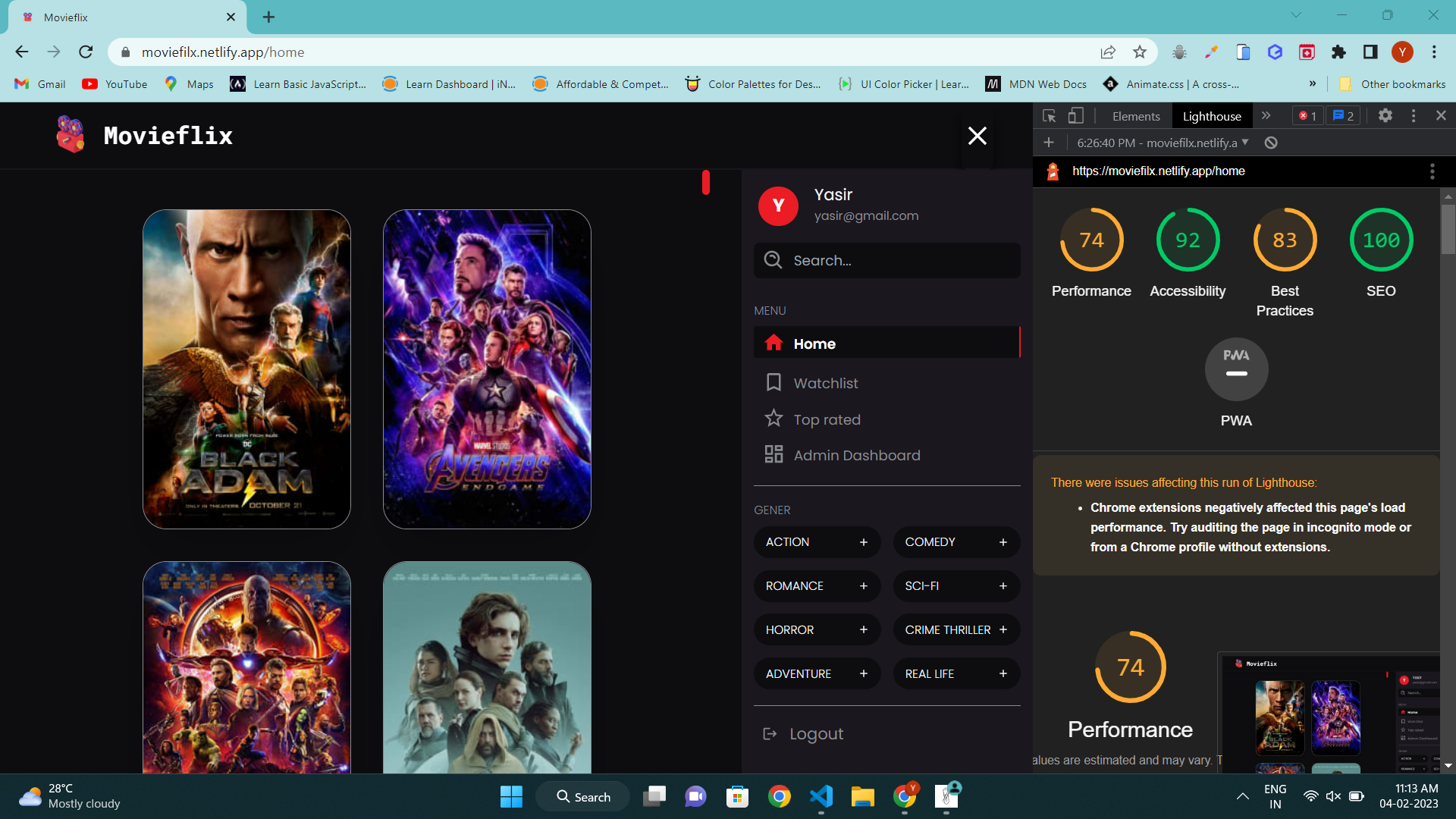
1. /signup
2. /login
3. /home
4. /home/watchlist
5. /home/top-rated
6. /admin

The Axios, framer motion, react-router-dom, react-toastify, and react-icons are the dependencies used to make this application.

* The Axios is used for calling APIs,
* Framer motion is used for adding animations,
* React-router-dom is used for client-side routing,
* React-toastify is used to display success, warning, and error messages.

# 4 Performance

The application is named Movieflix which suggests movies according to your selected categories, also the user can add movies that they want to watch to their watchlist. The performance report of the application derived from the lighthouse is as follows:



## 4.1 Reusability

The code written and the components used should have the ability to be reused with no problems.

## 4.2 Accessibility

The application is made by considering user accessibility.

* The website is user-friendly,
* Alt text is defined for the images,
* Clickable elements are given button tags for better user accessibility,
* Website text and background color combination are made such that text is clearly visible to the users.
* Title attributes are added to the buttons for better understanding.
* Success, error, and warning messages are displayed on the screen using react-toastify.

## 4.3 Best practices

Best practices such as not logging the errors in the console in the frontend, logging errors to the backend, minimizing the use of javascript, removing unused javascript, etc. are used in making the application.

## 

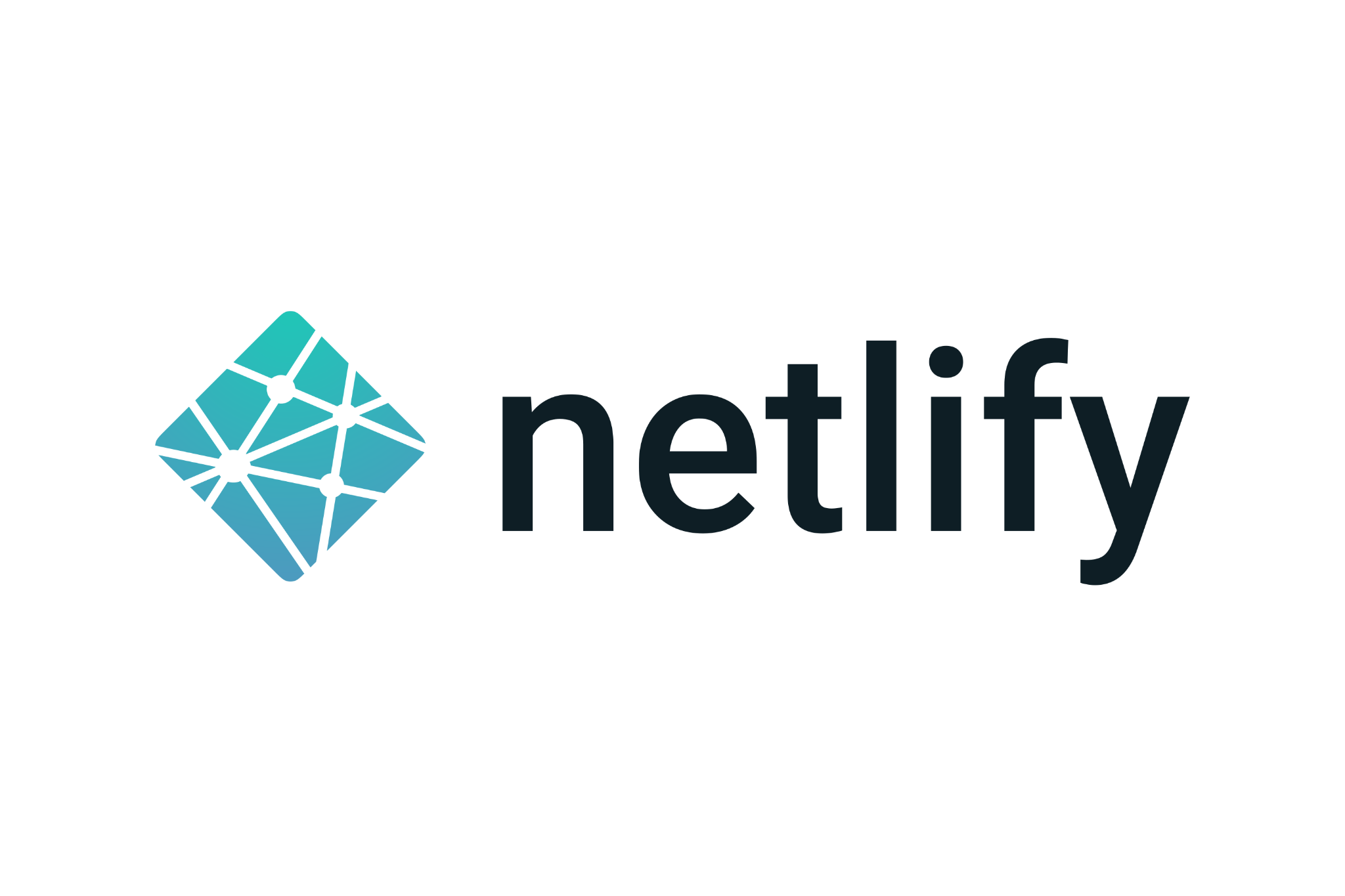
## 4.4 Deployment

MongoDB is used as the database and Cloudinary is used for uploading movie images.

## 

The frontend is deployed on Netlify and the backend is deployed on render.

## 

## **5** Features

The Movieflix contains the following features:

* User Authentication such as login, logout, and signup.
* Add user roles such as ADMIN and USER.
* User with ADMIN role gets access to the “/admin” route in which he can add, delete and update movies.
* Logged-in users can add and remove movies from their watchlist.
* User can see the Movie trailer, description, IMDb rating, and the platform at which the movie is available.
* Movieflix recommends movies based on the selected categories.
* User can also search movies based on movie name.
* The application is responsive on small and medium screens.

# 

# **6** Conclusion

In this Internship I have made a web application called Moviefilx. Moviefilx suggests movies based on selected categories, it contains movie details such as IMDb rating, movie trailer, description, and movie streaming platform link, and the logged-in user can add movies to their watchlist.

While doing this internship I learned about how to upload, edit and delete images in coudinary, about Mongoose, MongoDB, react-router dom v6, react hooks, how to add models, and how to deploy the frontend and backend of your application.