**YouTube Favorites**

This application allows a User to register/login to the application using JSON web tokens (JWT) for user authentication. The user has the ability to enter a search term, which makes a request to the YouTube API. A grid of results will then be displayed, the user has the ability to select a video from the list for a detailed view. A logged in User may save videos to their personal profile and view all saved videos in their favorites list. Saved videos may be removed from Users favorite’s view. The application is designed to be mobile responsive and user friendly.

Career Crusader is an online platform designed to connect job seekers with employers by providing an easy-to-use job listing and application management system. Users can sign up and create an account with their email and password, which allows them to search for job listings based on keywords, location, or categories. Employers can also sign up and create job listings, which job seekers can apply to directly on the platform. Users can also view and interact with user-submitted job listings, such as commenting or upvoting/downvoting. Additionally, users can manage their job applications, including receiving email notifications about their application status. The platform is designed to be responsive and user-friendly and protects personal information for both job seekers and employers.

**User Story:**

As a YouTube video user, I want to be able to search by topic and save videos to my user favorites dashboard/profile. I also want the ability to remove a video from my saved favorites list.

**Acceptance Criteria:**

As a user, I can register and sign-in to my account using my email and password.

As a user, I can add to and delete from my favorites listings if I am an authenticated user.

As a user, I can search for videos using the YouTube API based on user keyword input.

As a user, I can view the details of a keyword search, including its title, thumbnail preview, content creator and short description.

As a user, I can add friends and see their list of saved favorites – yet to implement

As a user, I can access the online platform from any device, and the platform should be responsive and user-friendly.

As a user, I expect the online platform to be secure and protect my personal information.

**Development:**

Frontend Development:

* Design a clean, responsive user interface using React.
* Implement CSS styling utilizing React Bootstrap.
* Implement React Router for browser navigation.
* Develop components and views for navigation, user registration and login authentication, search videos, results list, video details and favorites dashboard.
* Implement form validation and error handling for user input.
* Add interactivity to the video detail’s view, if a user is logged in and authorized, they can save or remove a favorite by clicking on a favorite’s icon.

Backend Development:

* Design and implement the database schema using NoSQL MongoDB database and Mongoose ORM, including tables for users, favorites
* Develop API routes and controllers for user creation, authentication validation, listing saved favorites, category management, and user-submitted job listing management.
* Implement secure user authentication and session management using JWT’s and cookies.
* Ensure the protection of sensitive information through the use of environment variables.
* Implement the logic for user interactions, such as adding and removing a video from the user’s favorites list.

Integration and Testing (APIs):

* Connect the frontend components and views to the backend server API routes and controllers, ensuring seamless data flow and functionality.
* Integrate any external APIs that may be required, such as video keyword search (i.e., YouTube API).
* Write React-Testing-Library/Jest unit and component functionality testing.

Deployment:

* Deploy the application to Render.com with the MongoDB Atlas cloud database and proper environment variable configurations. Setup proper routing/rewrites to allow frontend deployment and backend server deployment to communicate.

Project Setup:

* Initialize a git repository for version control.
* Set up your project structure, separating frontend and backend code and adhering to the MVC paradigm.
* Use NodeJS and ExpressJS for backend development.
* Use ReactJS for the frontend development
* Use MongoDB and Mongoose ORM for database, connection, and CRUD interactions.
* All team members set up the development environment, including tools Node.js

**RESTful Routes**

**/users**

**GET - /** Get single user by req.user object

**POST - /register** Create new user

* + body : { first: “ ”, last: “ ”, username: “ “, email: “ “, password: “ “, confirm: “ “ }

**POST - /login** Login user

* + body : { email: “ ” , password: “ ” }

**GET - /logout** Logout user (remove jwt **token** from cookie)

**GET - /verify-token** Verify users JWT token is valid

**/favorites**

**GET - /** Get all favorites

**GET - /:id** Get single favorite by ID

**POST - /create** Add video to users favorites

**DELETE - /:id** Remove video from users favorites

**/api**

**POST - /** Send query with API request to YouTube API

* + body : { query : “query search” }

**GET - /favorites/:id** Get single favorite by ID

**State Management:**

React ContextAPI

* + userContext
  + favoritesContext
  + searchResutlsContext
  + historyContext
  + authContext

**To – Do Items:**

* Remove favoritesContext (explore further to verify?)
* Update Results-List Component with loading state while querying for users associated favorites
* Remove extraneous comments
* Remove console logs
* In Search Container, **TEST** if both state and context should be used in the same component
* Refactor/Simplify server routes
* Add compare/validation logic for SERVER/users/register route logic for password == passwordCheck
* Figure out what the logic in the route /favorites/find/:id is supposed to be doing? Remove if not needed.
* Add **Key** property to ResultsList Component data
* Remove Favorites-Container component
* Remove **favorites** link in Navigation Component when user is NOT logged in.

—----