CVWO Reflection Writeup and User Manual

Name: Cao Wenjie
Matriculation Number: A0205344N
Hosted on Heroku: https://todo-app-manager.herokuapp.com/

Summary

This CVWO assignment opens up a whole new world for me as I have no experience in web development prior to this, so frameworks such as React and Ruby on Rails are completely alien to me. It has made me appreciate the amount of effort it takes to build a functional website from scratch as even a seemingly simple todo app took me countless hours of googling, experimenting and testing. Nonetheless, this is easily the most rewarding experience I had since doing Computer Science as I have never done a full-scale project like this on my own before.

Accomplishments and problems faced

The first thing I needed to learn was Ruby on Rails in order to support the backend of my todo app. Although the materials provided in the assignment sheet were certainly very helpful for understanding the basic MVC framework, I find it difficult to piece together the different features and apply them to my project requirements. As a result, I rely mostly on online tutorials to look for materials that are relevant to what I wanted to implement. It took me a while to understand that backend mainly handles the transfer of data to and from the database while frontend is responsible for displaying the pages that users see. One accomplishment I was proud of is that I managed to piece together information from different sources to merge backend rails with frontend react under one server using webpacker. This proves to be a difficult process as many online resources actually implement them separately. As a result, it was a tall order for me to understand how to incorporate React BrowserRouter with Rails routes, as well as using api calls to access backend data.

As I feel that different users should have access to a personalized todo list that only they can manage, my todo app would require a login and user authentication feature. Searching for an implementation exposed me to many abstract terms such as cookies, sessions and tokens which were not easy to grapple with and to this day I do not fully understand. Although I eventually settled on using the relatively simple rails sessions, I am really glad to be able to implement it successfully as such an authentication feature is highly transferrable and is often essential in many web apps. However, I do hope to learn and apply more secure implementations in the future such as JWT, which I was not able to manage this time.

One issue I was stuck for some time was how to ensure a logged in user does not get kicked out by refreshing the page. After much testing, I realized that it has to do with certain components prematurely checking the login status before the api call can update it from false to true even though the api call is made first. In the end, I managed to resolve it using async-await so that the function waits for the api call to be fully executed first before proceeding.

Areas for improvement

Although I was pleased that I managed to implement most features that I wanted to, there are certainly many rooms for expansion and improvement. I initially wanted to use the Heroku job scheduler to check the database of todo items every day 12am to mark out those that are overdue so that they are displayed differently in the frontend. However, to do so I would need to provide my credit card information so I decided to implement the checking entirely on frontend instead.

Furthermore, I feel that my react code can be better structured and organized. Currently, I have several duplicated functions that can potentially be abstracted out and my individual components are relatively large which can be broken down to multiple smaller components to make the code cleaner and more extendable. The way my props are passed to different components can also be improved as certain props currently are being passed indirectly.

My inexperience with bootstrap also led to several styling issues. One thing I can definitely work on is mobile optimization as currently certain pages do not appear as intended on mobile.

Final thoughts

This assignment has made me better appreciate the practical application of CS as I felt immense satisfaction from seeing the final end product and realizing how I have jumped through countless hurdles (correctly or otherwise) in order to achieve it. This is my very first web application and I know I have only touched the tip of the iceberg of web development. I hope to reinforce what I have learnt from this project and expand to more exciting projects in this realm.

User Manual

Homepage (landing page)

User can choose to sign up for a new account or login if they have an existing account.

Login page

User need to key in the correct email address and password they provided during sign up in order to login. Any attempt to access the main page without logging in will be redirected to the login page with an error message. Once user is authenticated successfully, he would be redirected to his personalized todo items page (main page). An error message would be displayed in login form if login is unsuccessful. From the login page, user can also navigate to the signup page to create a new account.

Signup page

User would need to fill in their username, email, password and password confirmation (which would need to match password) in order to register for a new account. An error message would be displayed if form is not filled in properly or the email address selected has already been taken. Upon successful registration, user will be redirected to login page with a success message. From the signup page, user can also navigate to the login page to login.

Todo items page (main page)

This page displays the current list of todo items that the user has using a table. Todo items that are overdue are in red and there is a message telling the user the number of overdue todo items he has.

User can search for the todo items by title or category. As user types into the search bar, the table of todo items will be filtered to include only those that matches the keywords. At the header of the table, user can also sort the items by title alphabetically (descending), by deadline (earliest to latest – default) or by creation date (latest to earliest).

Under options, user can update, complete or delete their items. The update button would open up a form for user to key in any changes for the item. The complete button would move the item to the completed todo item page while the delete button would remove the todo item entirely. A confirmation alert modal would appear when user completes or deletes an item to ensure that the action is intended before it is executed. If an action is executed successfully, there will be a success alert message.

From the navigation bar, user can toggle between this main page and the completed todos page, as well as logout.

Create new todo page

This is where user can create new todo items and input fields such as title, description, category and deadline. Only the title and deadline are compulsory fields. The datepicker is configured such that only future datetime can be selected. Once the todo is successfully created, user will be redirected with a success message to the main page where he can view his newly created todo.

Completed todo items page

This page contains all the todo items that the user has marked as completed. The purpose is to allow user to keep these items for potential future reference while not crowding up the main todo page. They can be deleted when they are no longer needed. The searching and sorting features are the same as those in the main page.