CEN 308 Software Engineering

Project documentation

Bookmark manager - managefox

Prepared by: Amer Karamustafić, Selma Aljićević
Proposed to: Nermina Durmic Assist. Prof. Dr.
Aldin Kovačević Teaching Assistant

22.6.2022

Introduction

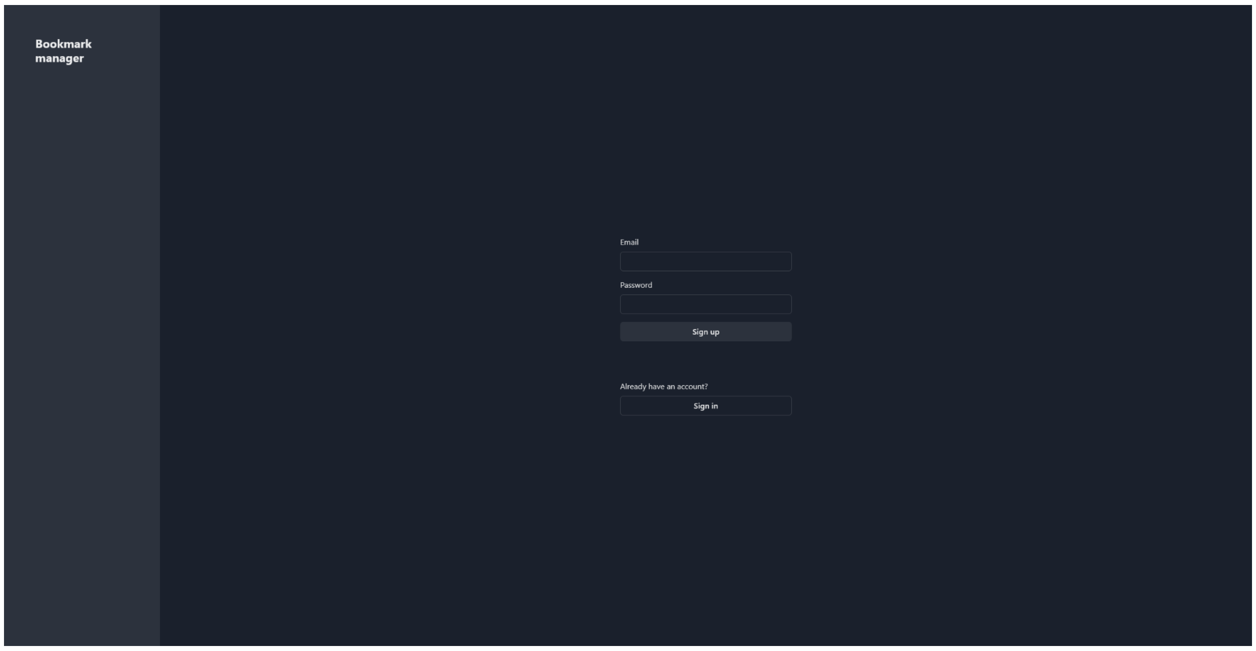
About the project

There are many bookmark manager options available on the market right now but most of them don’t offer advanced features such as support for collections and tags, backups, keyboard shortcuts and similar. Goal of our project is to fill that void. The project is hosted on vercel and you can find it on the following url managefox.vercel.app.

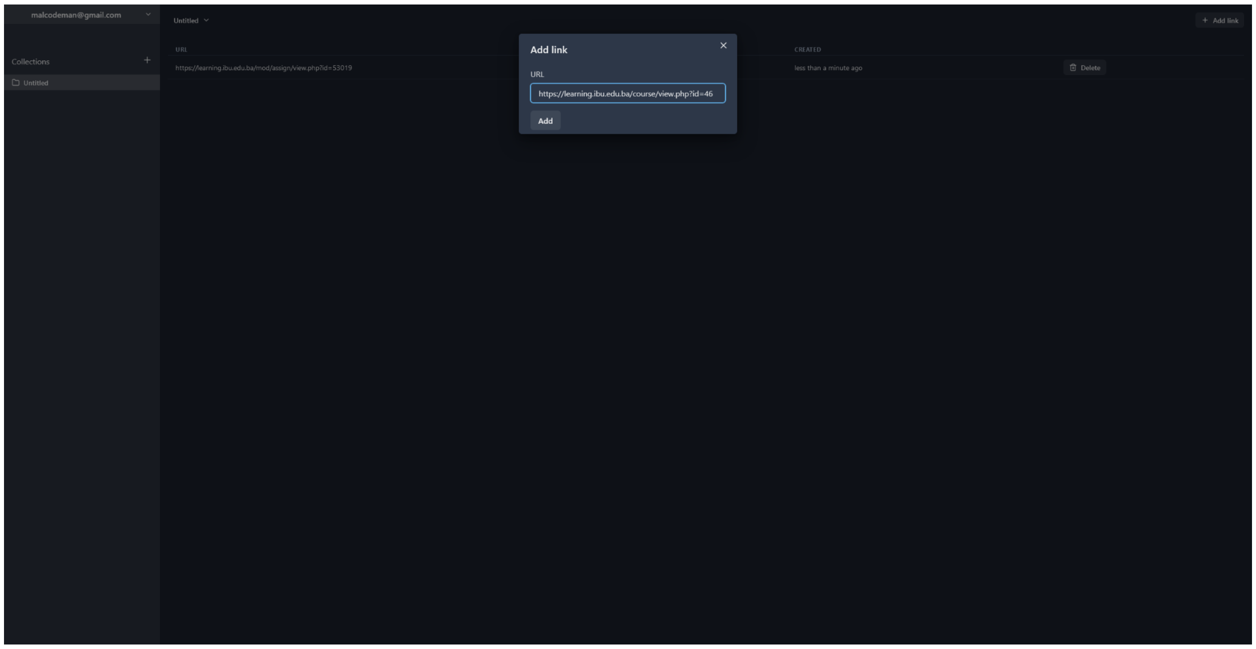
Project Functionalities and Screenshots

In order to attract users with high feature standards we implemented several features.

Keyboard shortcuts
Users should be able to use keyboard shortcuts to accomplish most of the available tasks.
Clipboard copy-paste
If the browser supports interaction with the user clipboard we simply add links from the clipboard to enable easier and faster link saving.
Optimistic updates
Interaction with the app should be extremely fast and this will be accomplished with optimistic updates feature which means UI update won’t depend on the speed of the network calls.



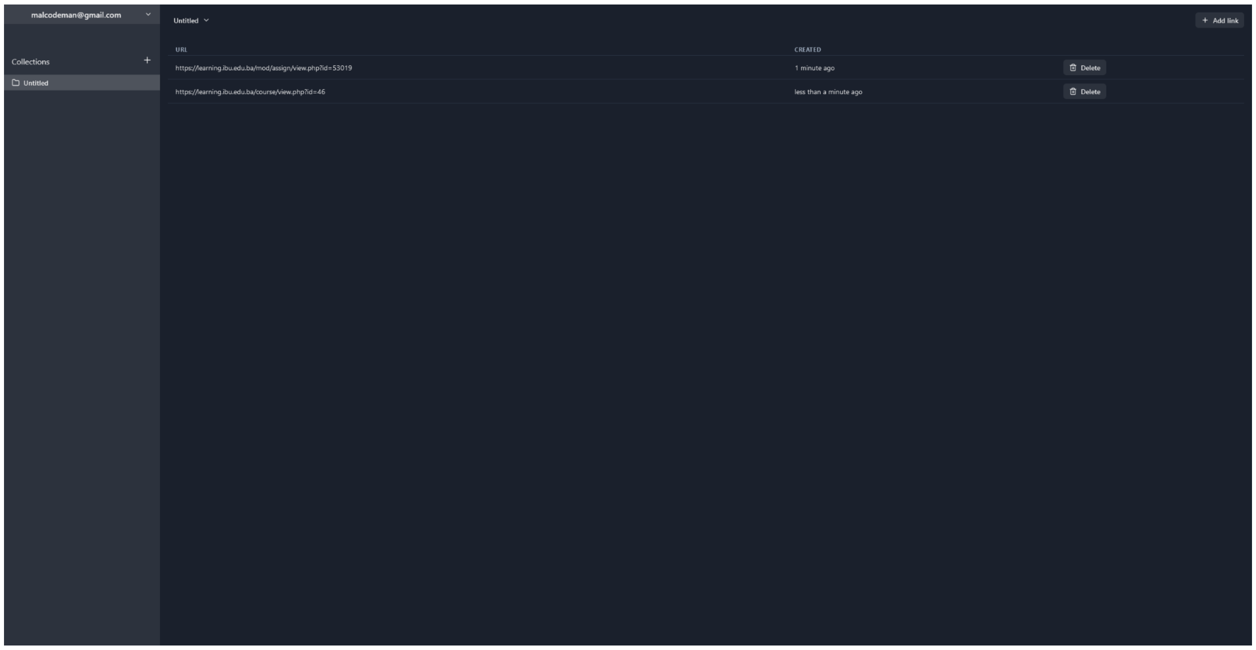
Sign up page



URL	CREATED	
https://learning.ibu.edu.ba/mod/assign/view.php?id=53019	less than a minute ago	Delete

URL: https://learning.ibu.edu.ba/course/view.php?id=46

Insert link modal



URL	CREATED	
https://learning.ibu.edu.ba/mod/assign/view.php?id=53019	1 minute ago	Delete
https://learning.ibu.edu.ba/course/view.php?id=46	less than a minute ago	Delete

Collection with links page

Project Structure

Technologies

For the frontend we use Next.js which is React.js framework which enables us to create static-rendered websites which will in turn boost our search engine optimization rating and enable faster growth. Our database is Postgres running on Supabase backend as a service provider. Data fetching is done with supabase REST api. We use ramda library for all our functional programming needs.

Database Entities

- Users - our users table used for storing information about users
- Collections - “folders” for links
- Links
- Users (supabase) - internal supabase users table for authentication

Architectural Pattern

Client-server pattern
In the client-server architecture pattern, there are two main components: The client, which is the service requester, and the server, which is the service provider. Our client in this case is our next.js front-end web application, and our server is supabase which is our back-as-a-service provider. We choose this pattern because it is simple to understand and implement, it also goes well with our intention to keep the costs of operation as minimal as they could be.

Design Patterns

Stateless Components
We used stateless components in order to make our codebase as simple as possible and to enable us to reuse as many components as we could.
Examples of this can be found in our table component, for now we only use it for links but in the future we could use it for all sorts of data listings.
Conditional Rendering
In the process of writing react components, the need often arises to render a certain JSX code based on the state. This is achieved through conditional rendering. Example of this can be found in our collection rename handling, where we render the editable component if the user clicks on the rename menu item.
Render Props

Render props prove really handy as they allow us to share the same state across different components. Instead of hardcoding the logic inside each component, you can use a function prop to determine what to render. Examples of this can once again be found in our table component where we pass different props such as size, columns and data.

React Hooks
Hooks let us use state and other react features without writing a class, we used them extensively. We used react, chakra hooks and even built our own.
useAuth hook is a great example of just how powerful hooks are.
Controlled Components
By default react forms have support for both controlled and uncontrolled components. It is highly recommended that you use controlled components so we can fully control how our components behave. For our form management we use react-hook-form which even further simplifies our controlled component, an example of this can be found in account modal form.

Conclusion

Design and implementation of this project was really challenging and interesting, primarily because we used technologies that we never encountered before such as supabase. The most challenging part of this project is figuring out how to utilize supabase properly and how to implement the keyboard shortcuts. If we were to redesign some part of the application that would most likely be api requests, we would opt to use GraphQL since just recently supabase added support for it and it’s proven to be super fast. One of the most important aspects of our implementation is how we managed to keep the operating costs minimum, hosting on vercel with supabase as a backend costs us nothing so we could potentially have a long runway to find profit with this project if we turn it to startup.