ReadFeed Catalogue

BY
Pankaj Khemani
and
Nazar Kashif

Github Repository:

https://github.com/Prince-5/ReadfeedCatalogueCode

Introduction and Background:

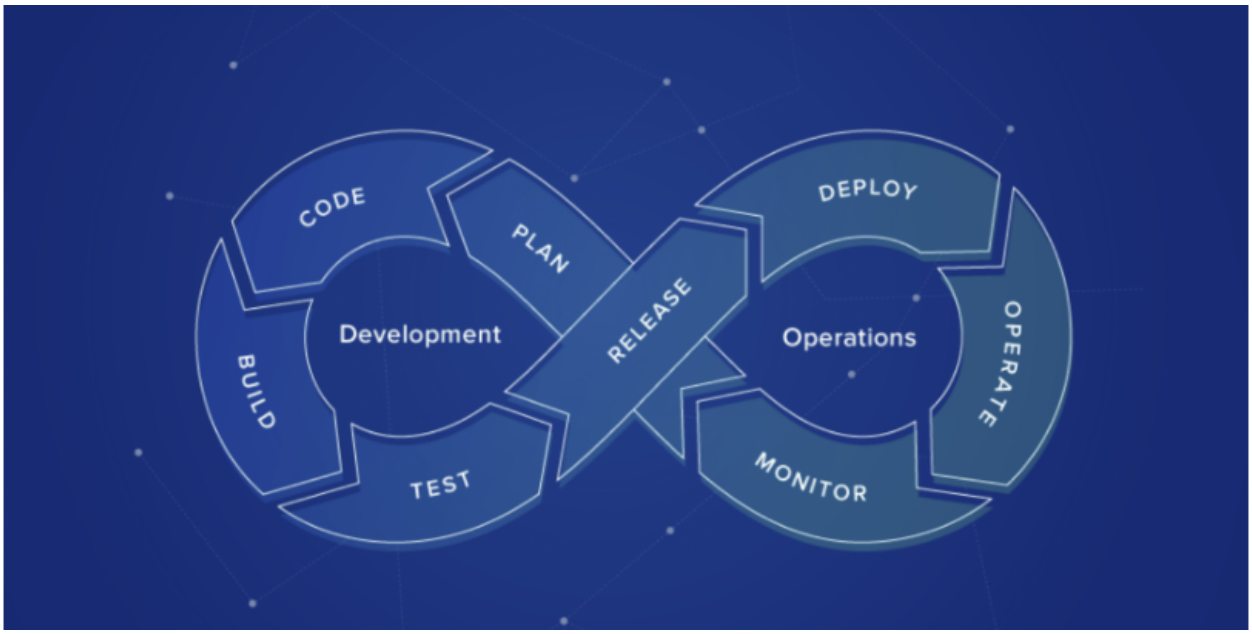
This project's main purpose is to use DevOps tools to create our ReadFeed Catalogue application. The goal is to study, understand, and apply DevOps tools to learn, understand, and implement Continuous Integration, Continuous Deployment, and Continuous Monitoring.

DevOps is a set of cultural concepts, practices, and technologies that improves an organization's capacity to produce high-velocity applications and services, allowing it to evolve and improve products at a faster rate than traditional software development and infrastructure management methods. Organizations can better service their clients and compete in the market because of this quickness.

Main principles of DevOps:

- Automation - Infrastructure as code
- Lean – Small batch sizes, focus on value for end user
- Measure – Measure everything; Show improvement
- Sharing - Collaboration between Dev and Ops

Devops Lifecycle:



Benefits of DevOps :

● Companies are more likely to stay up with technology innovations (such as the Internet of Things), and DevOps can help with market support for digital services. DevOps frees up resources for innovation as a result.
● Adaptability: Because tighter interaction leads to the production (and execution) of superior ideas, a flexible approach to enterprise-wide development is more efficient.
● IT and business integration. Another goal of DevOps is for corporate divisions to build mutual understanding in terms of innovation and digital transformation.
● Improving customer service and increasing customer satisfaction. One of DevOps' main objectives is to produce better software and deliver it to end users more quickly (which increases the loyalty of the latter)

About ReadFeed Catalogue:

- The ReadFeed Catalogue enables you to upload images of books and food that you have tried and review them.

- It allows users to see what has been posted and see the reviews on food and books that have been given by the people already using the platform.

- It has many features like the Login User, SignUp User, JWT authentication, Contact message notification on mail, Category Search, Search by a specific Keyword, Pagination, etc.

- It is like a small community where people can share their opinions regarding specific books and food items. It will be particularly useful for communities like students from a specific college, school, or people belonging to the same local area.

About Tech-Stack:

Back-End and Database:

We used the Django-REST framework for our backend and SQLite as our database. Django REST framework is a powerful and flexible toolkit for building Web APIs. WE chose Django-REST framework because of the following factors-

- It provides web-browsable APIs.
- Its authentication policies include packages for OAuth1a and OAuth2.
- It has serialization that supports both ORM and non-ORM data sources.
- It’s customizable all the way down - just use regular function-based views if you don't need the more powerful features.
- It has extensive documentation, and great community support.
- It is used and trusted by internationally recognised companies including Mozilla, Red Hat, Heroku, and Eventbrite.

The REST framework requires Python and Django.

We used SQLite for our database as SQLite file format is stable, cross-platform, and backwards compatible and it is small, fast, self-contained, highly reliable, full-featured. Also, SQLite is by default configured by the Django-REST framework, with no extra installations.

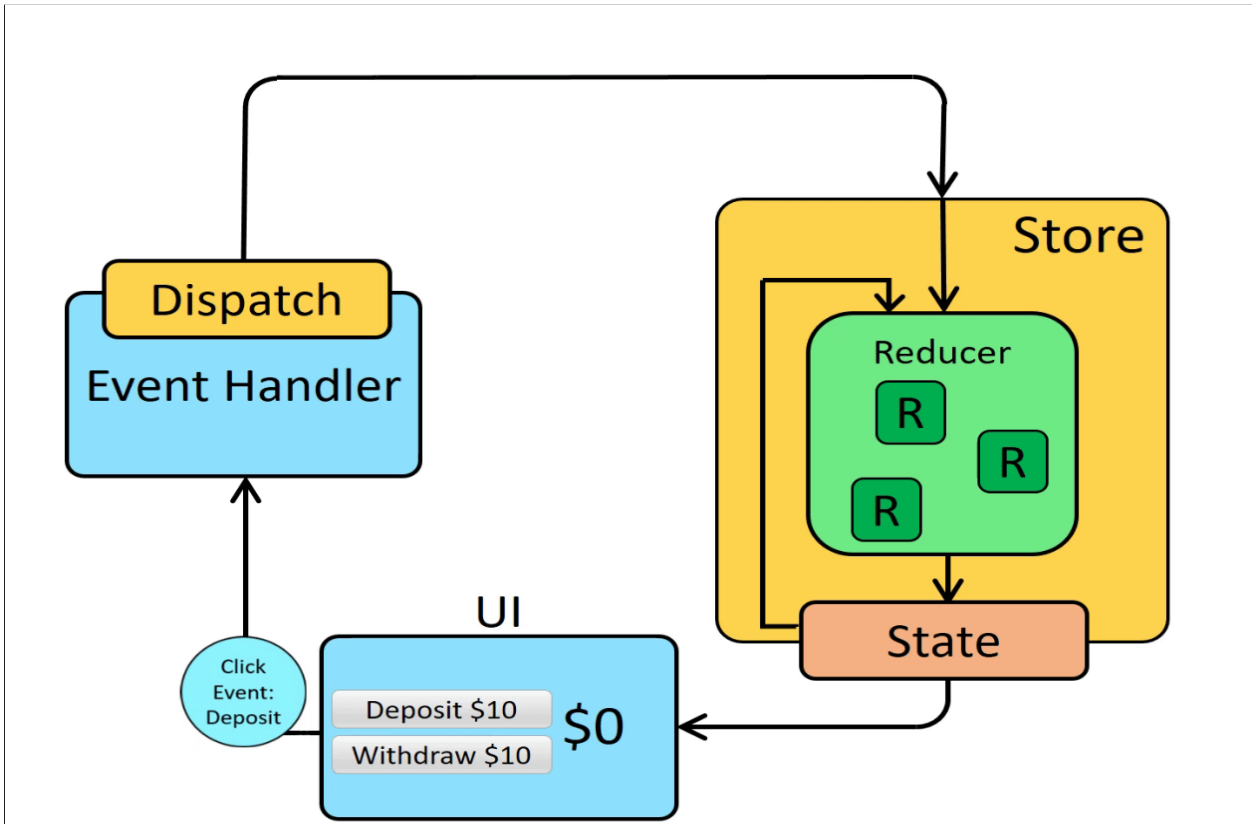
Front-End:

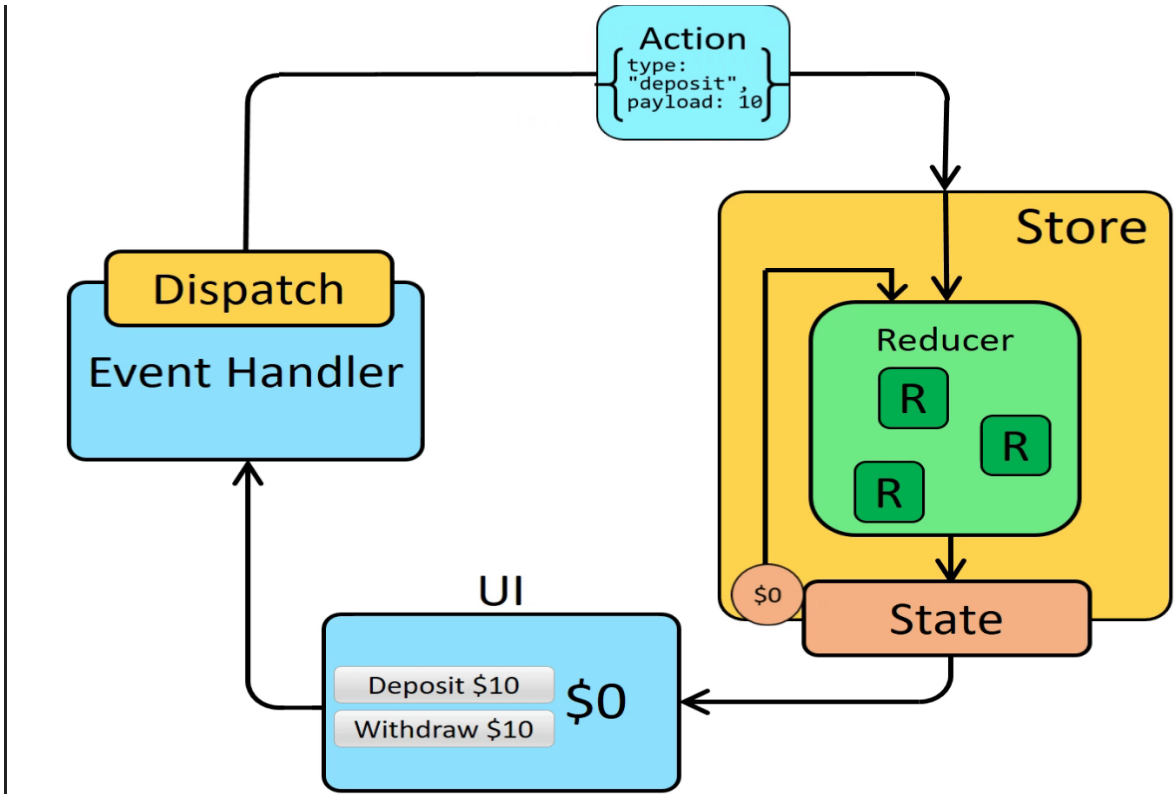
We used React and Redux for our frontend.
React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes. We chose React because of the following factors-

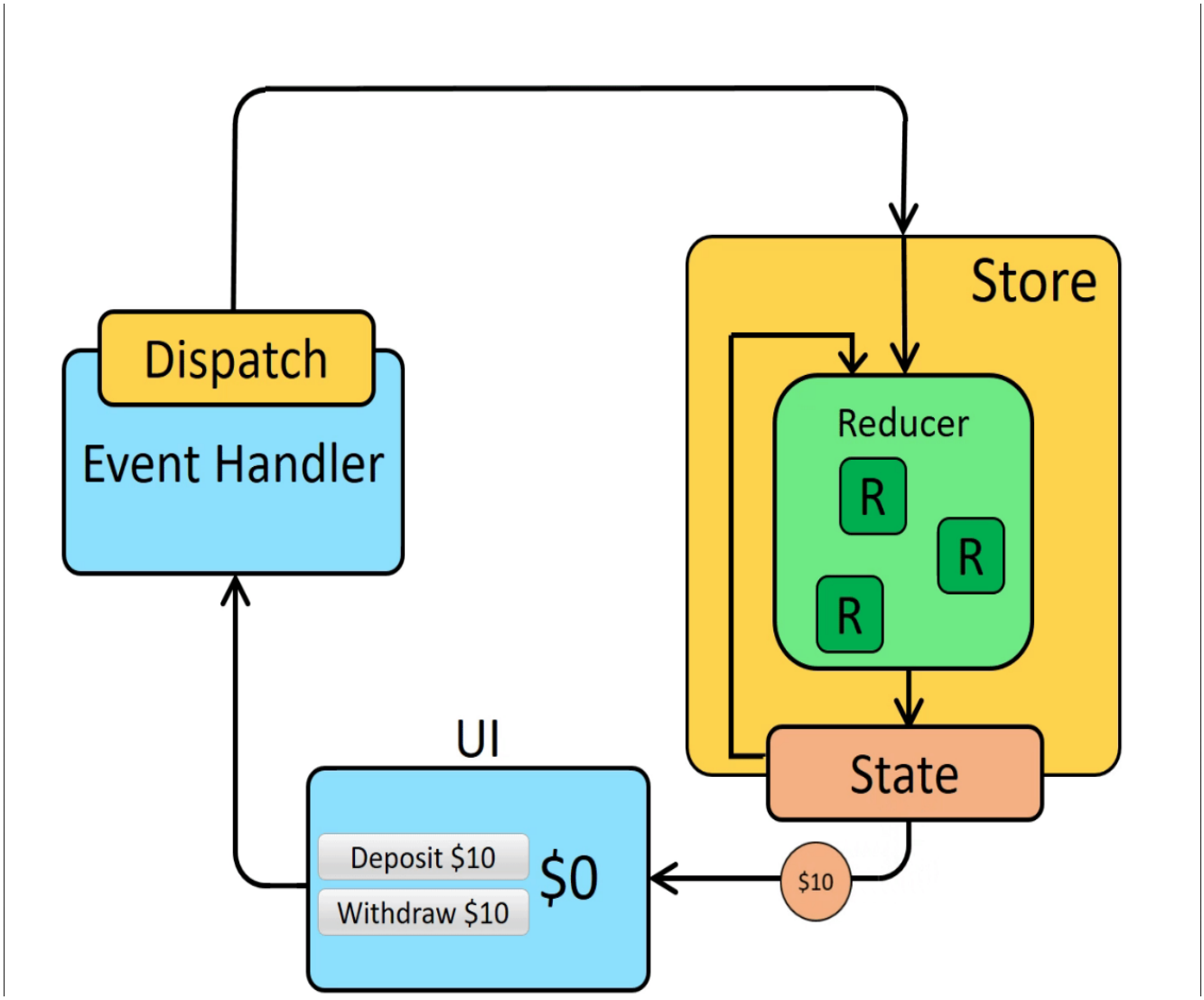
- It is declarative and declarative views make the code more predictable and easier to debug.
- We can build encapsulated components that manage their own state and then compose them to make complex UIs.
- Its Learn Once, Write Anywhere policy makes it a perfect fit for frontend as they don't make any assumptions about the rest of our technology stack, so we can easily develop new features in React without rewriting any of the existing code.
- React can also render on the server using Node that makes it even more powerful.

We also used Redux that acts as a state container for our application. We used Redux because of the following factors-

- Redux helps us write applications that behave consistently, run in different environments and are easy to test.
- It helps in centralizing the application's state and logic that in turn enables capabilities like undo/redo, state persistence etc.
- The Redux DevTools make it easy to trace when, where, why, and how your application's state changed.
- Redux works with any UI layer, and has a large ecosystem of addons to fit our needs. In fact this was one of the main reasons that made us choose Redux.

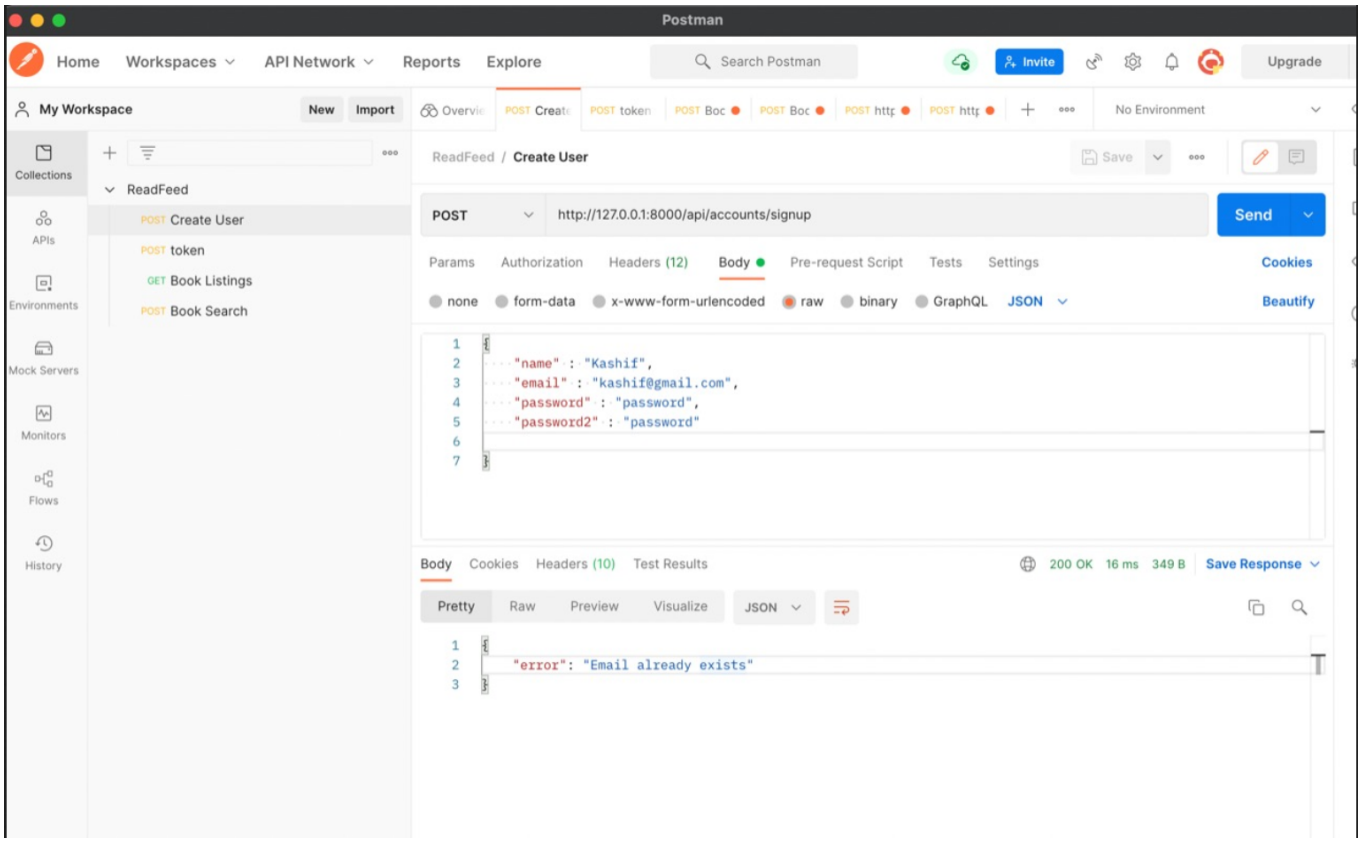

Store
Dispatch
Event Handler
Reducer
R
R
R
State
UI
Click
Event:
Deposit
Deposit $10
Withdraw $10
$0


Action
type:
"deposit",
payload: 10
Store
Dispatch
Event Handler
Reducer
R
R
R
$0
State
UI
Deposit $10
Withdraw $10
$0


Store
Reducer
R
R
R
Dispatch
Event Handler
UI
Deposit $10
Withdraw $10
$0
State
$10

Postman:

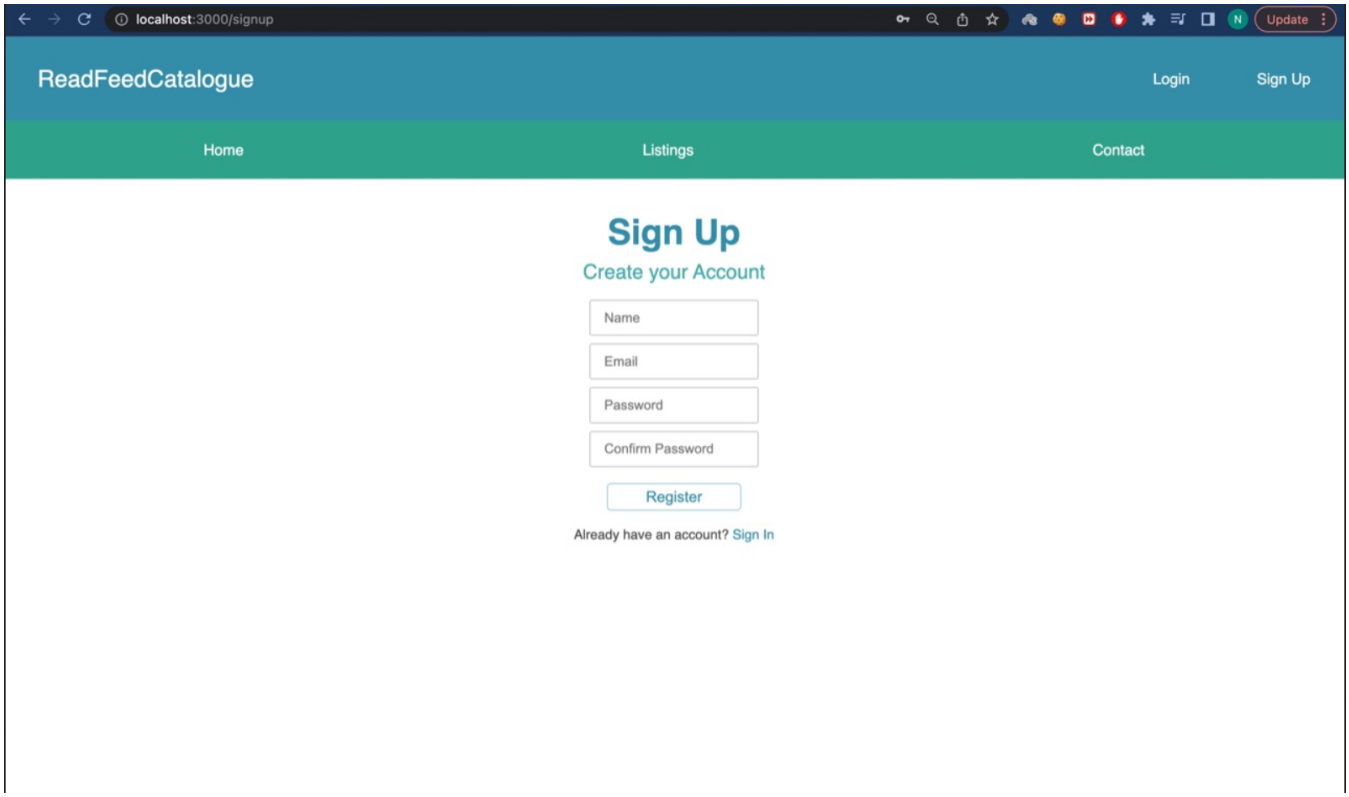
We used Postman for testing the API and checking the response.



Main Features of our application:

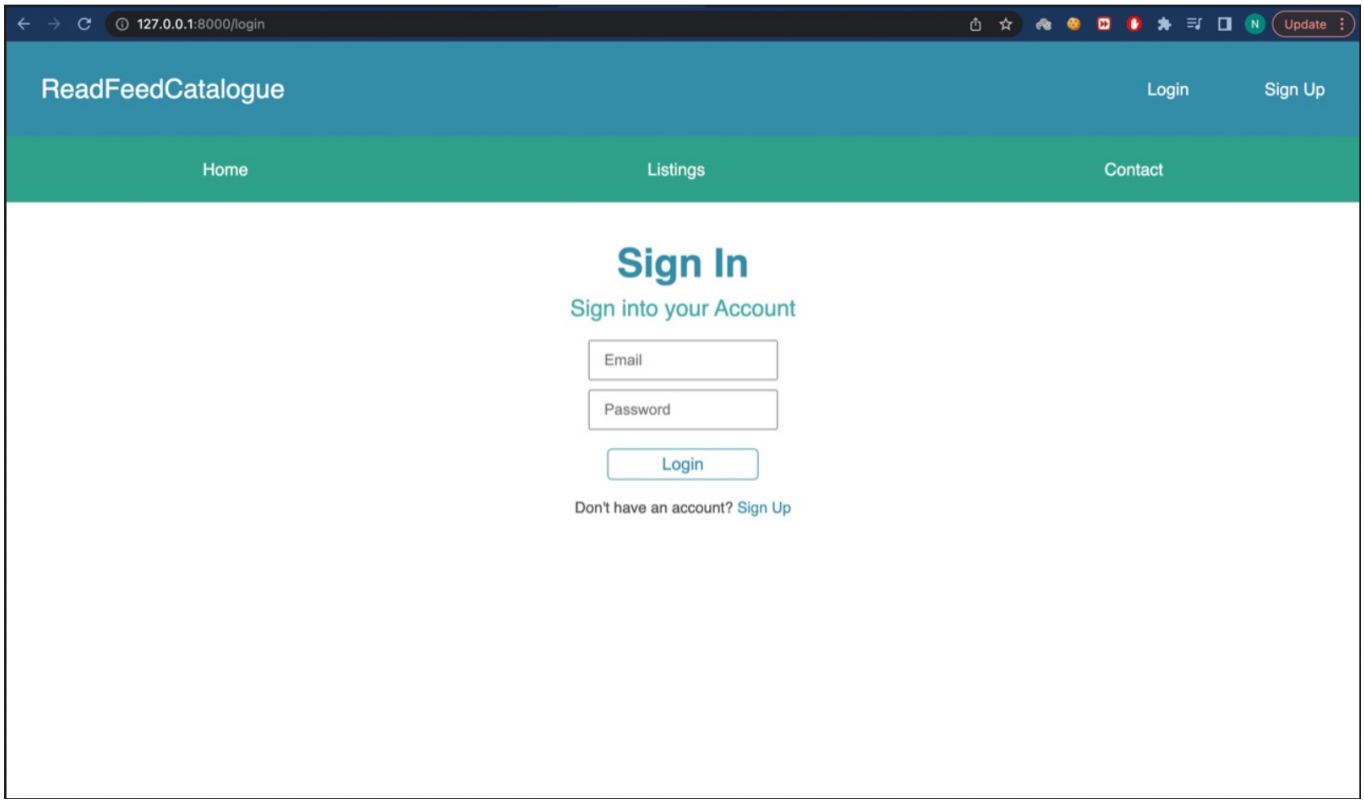
SignUp User:

Allow all new users to make a new account.



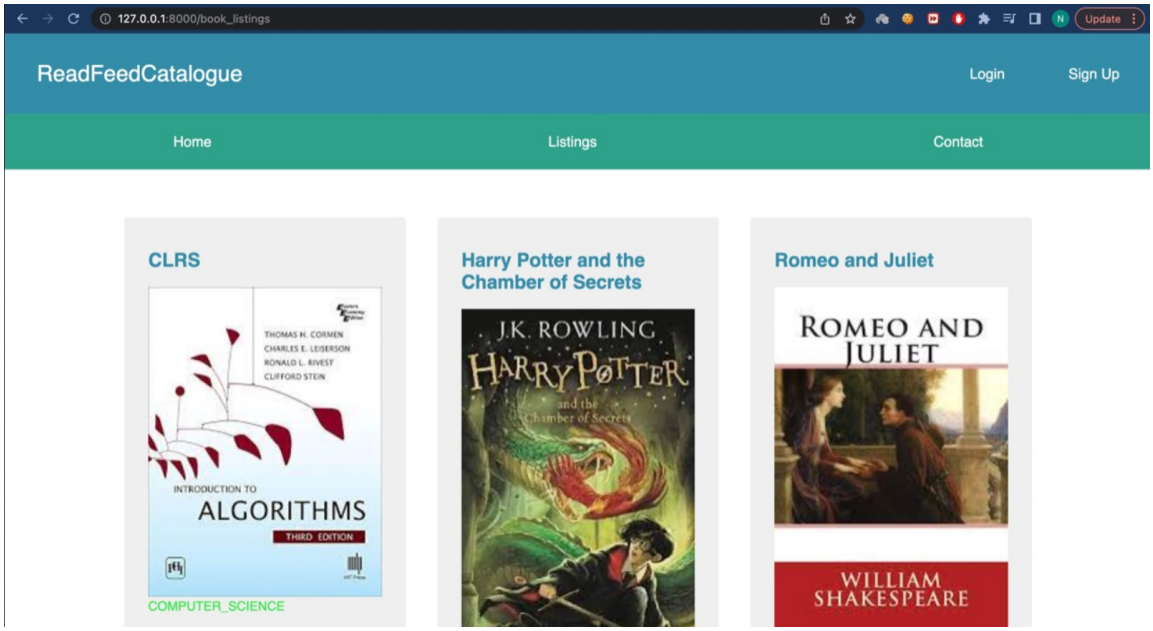
Login User:

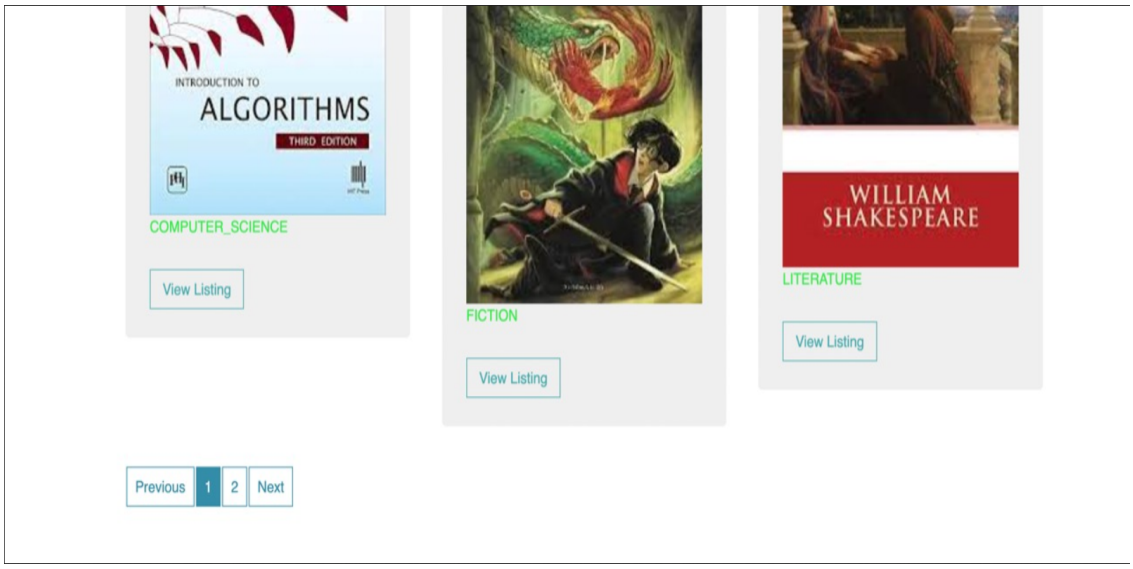
Allow existing users to login.



Listings:

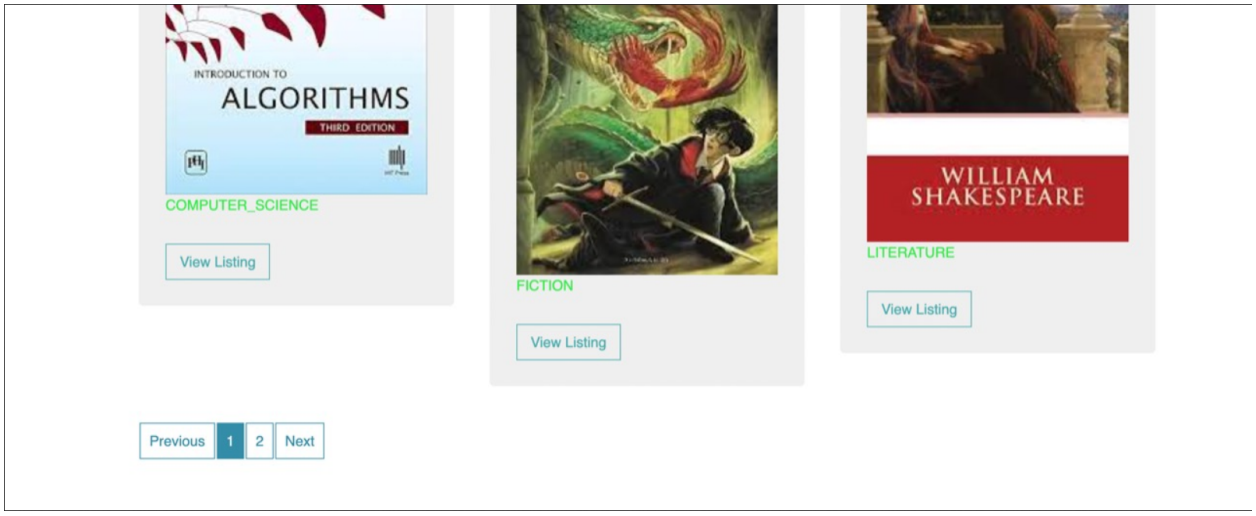
Display all the books and food reviews at one place.

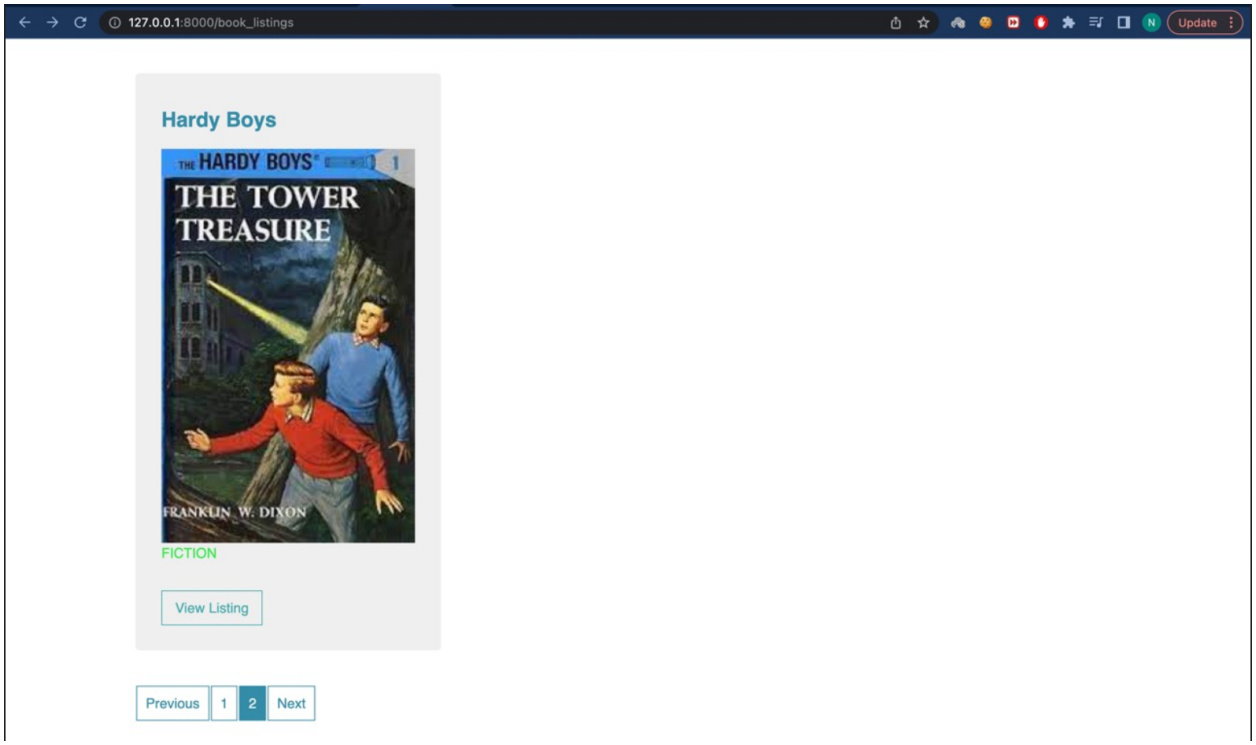




Pagination:

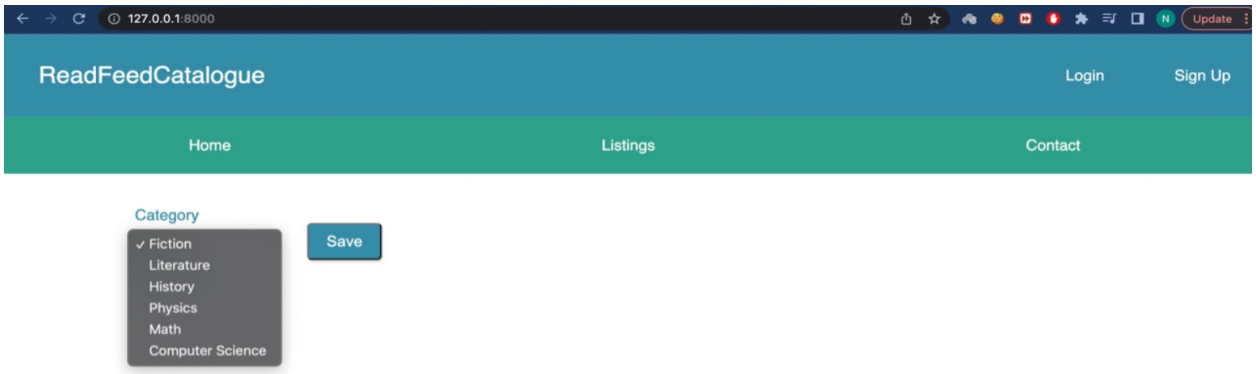
Pagination for handling too many reviews page by page.

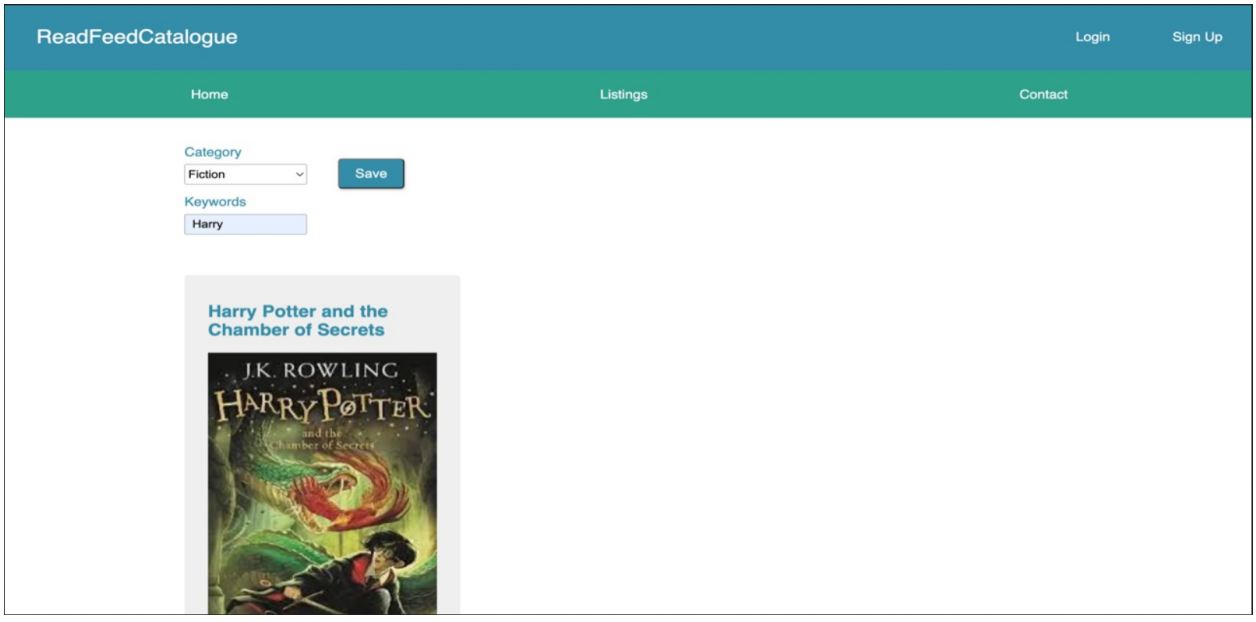




Category Search:

We can categorize the reviews for food and books. We can even further subcategorize the books into their genre (like fiction, history, math, etc.)

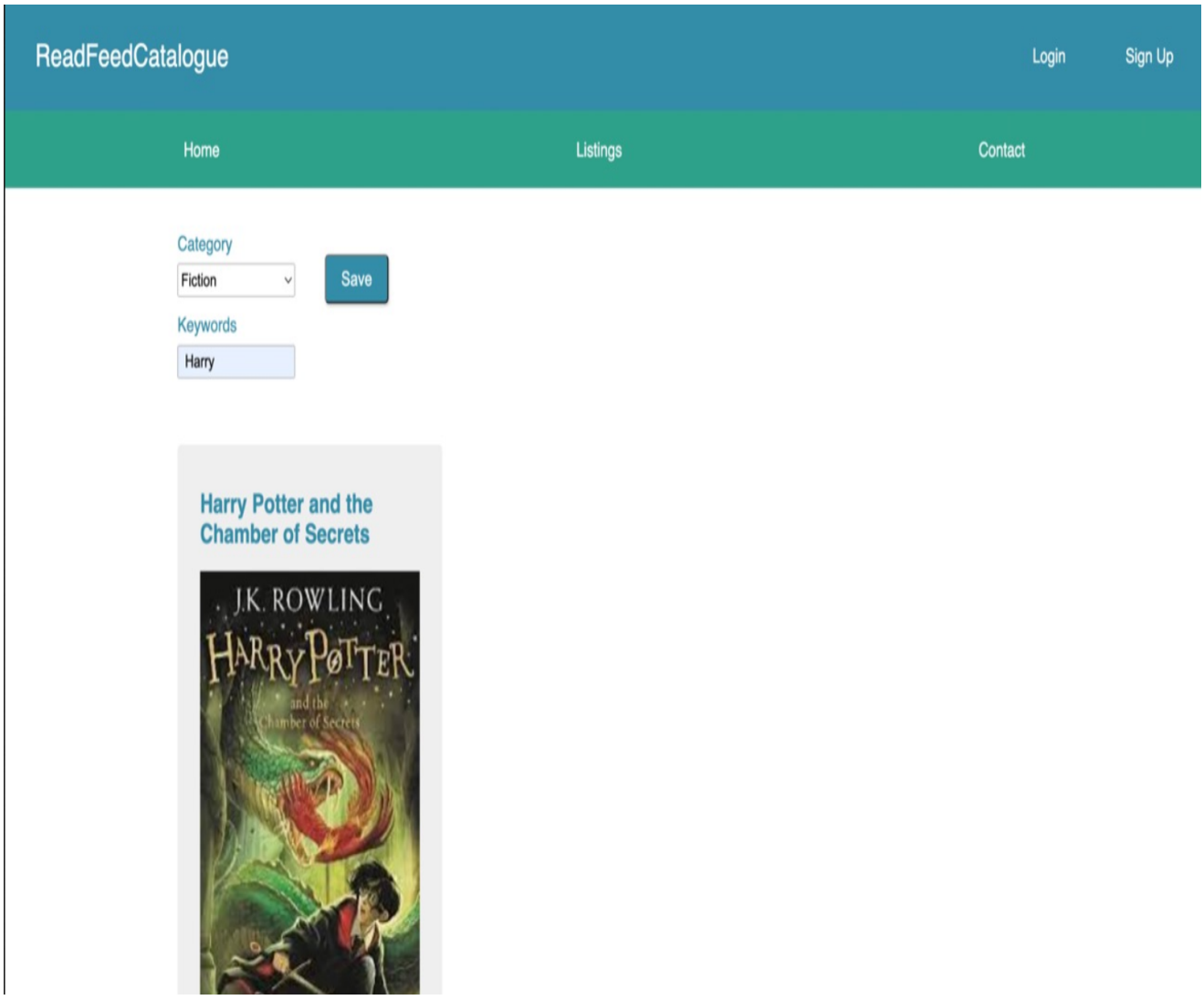




Search by a specific keyword:

If you want to search the review by entering a part of its title this feature will help you.

This feature also supports partial keyword search for our application.



FICTION

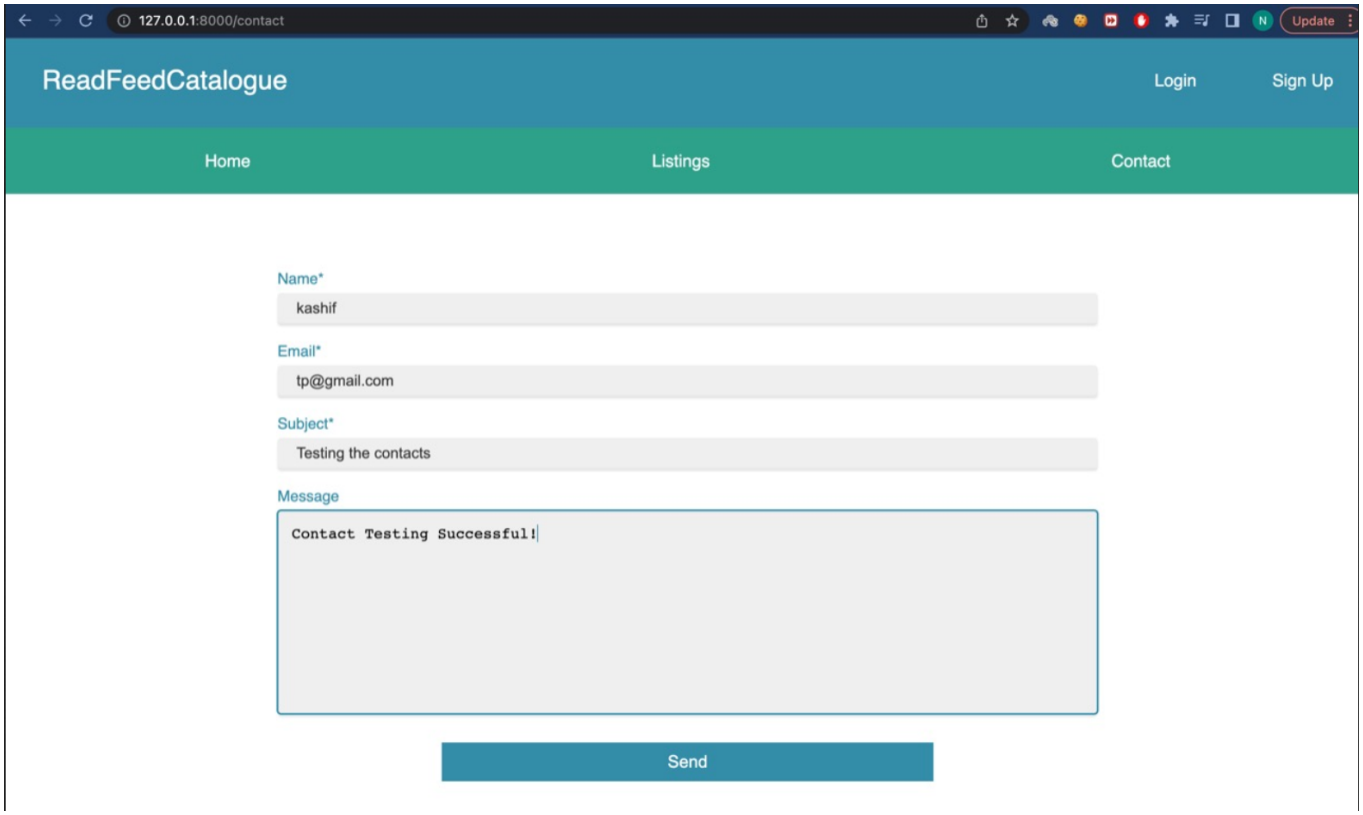
View Listing

Previous 1 Next

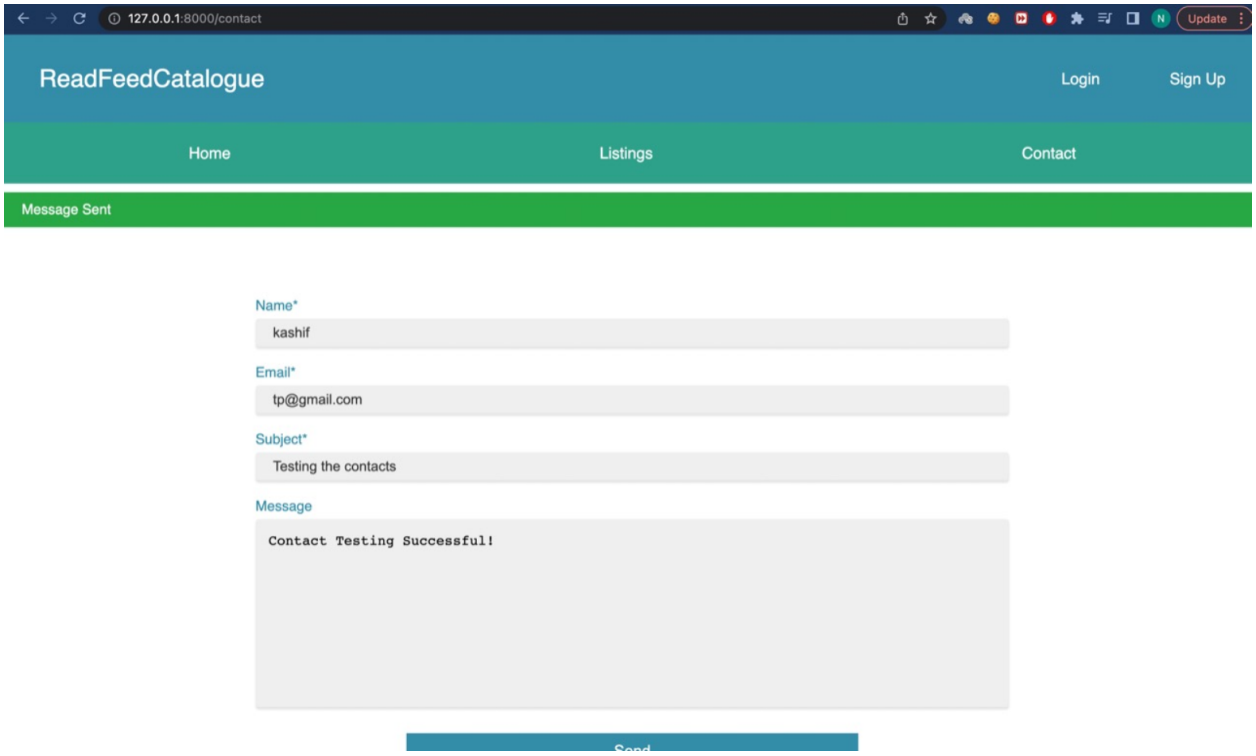
Contact Message Notification on Mail:

Any user can send an email to us using this feature. It’s for contacting us directly. If someone is facing any problem on the app or even bug reports, etc. can directly be communicated with us.

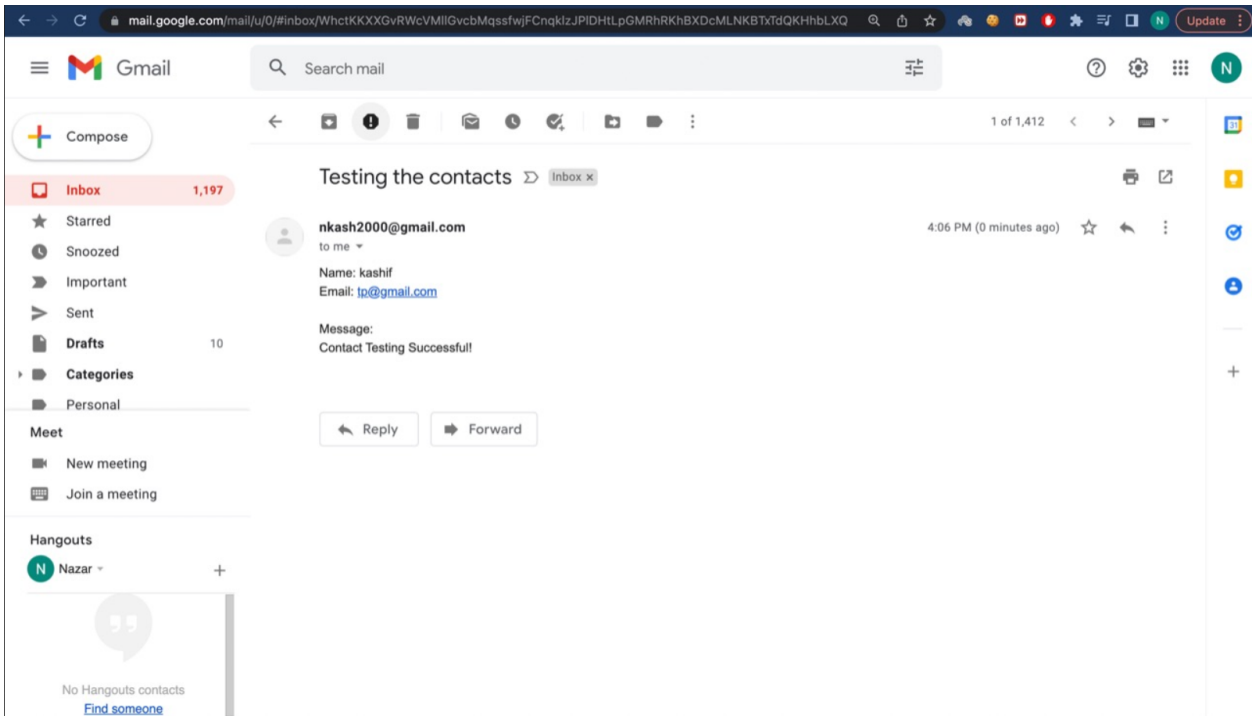
The email is being sent to us on gmail through SMTP(Simple Mail Transfer Protocol).



After clicking the send button:

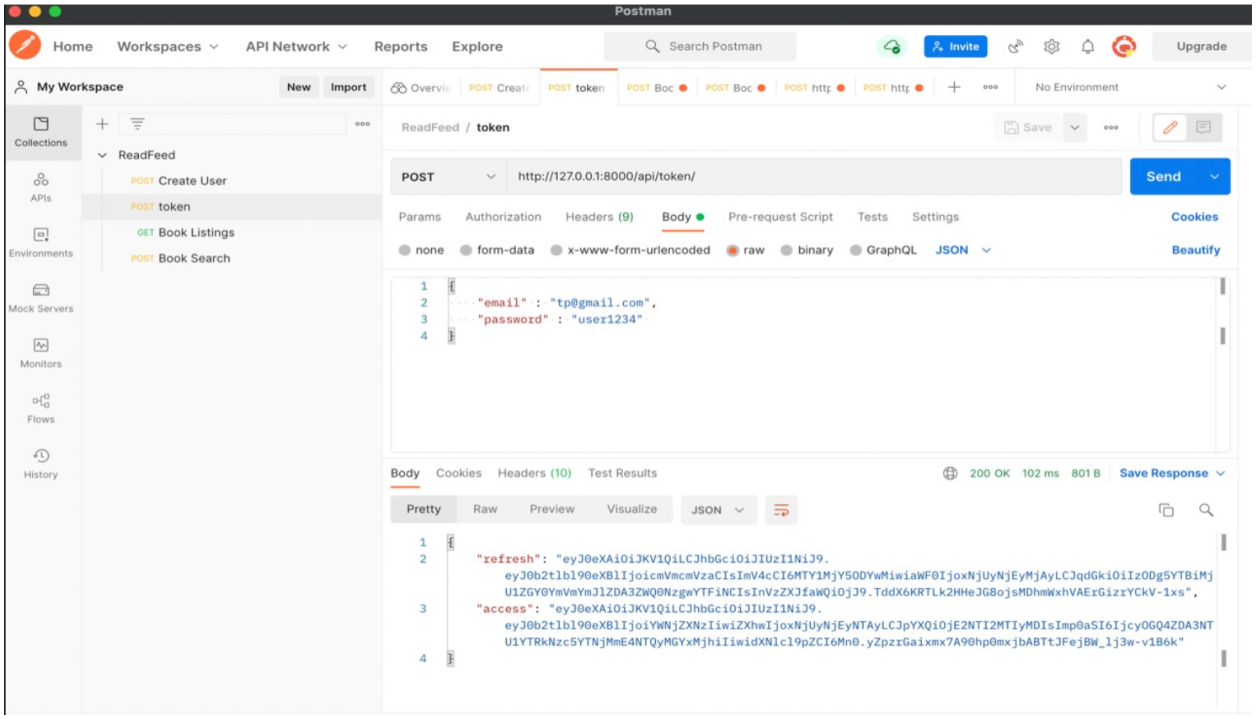


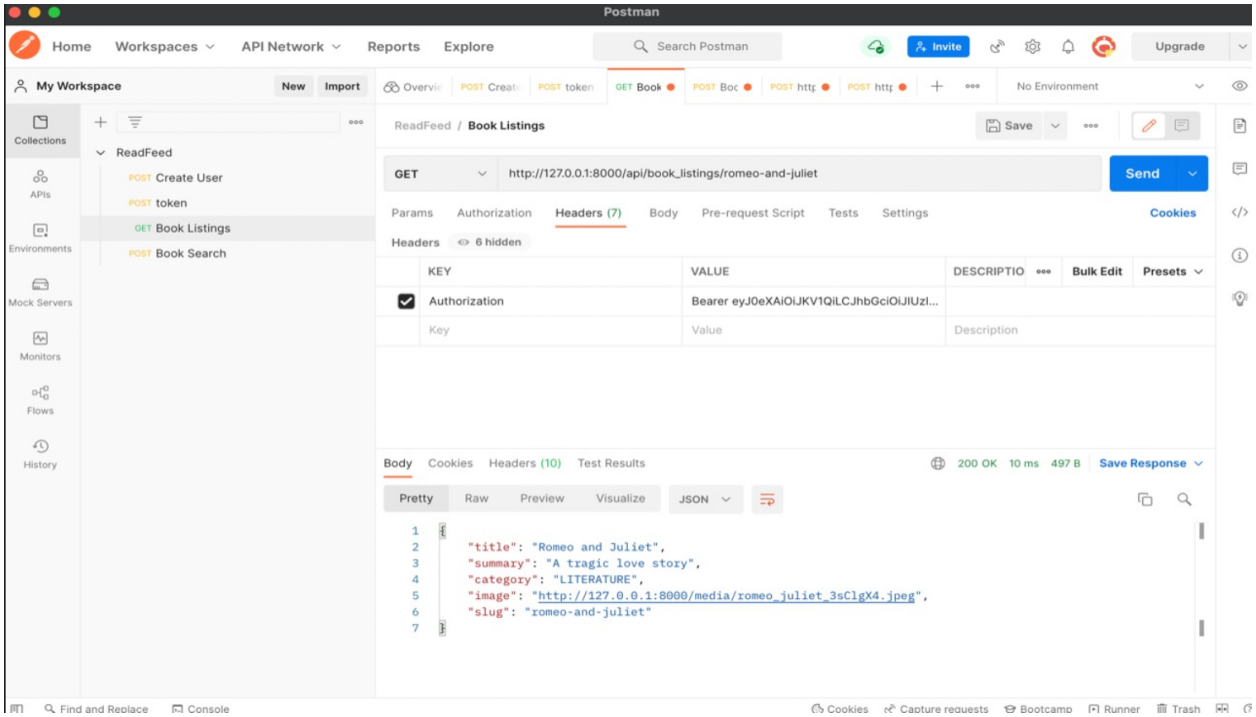
We receive the above subject and message in our email:

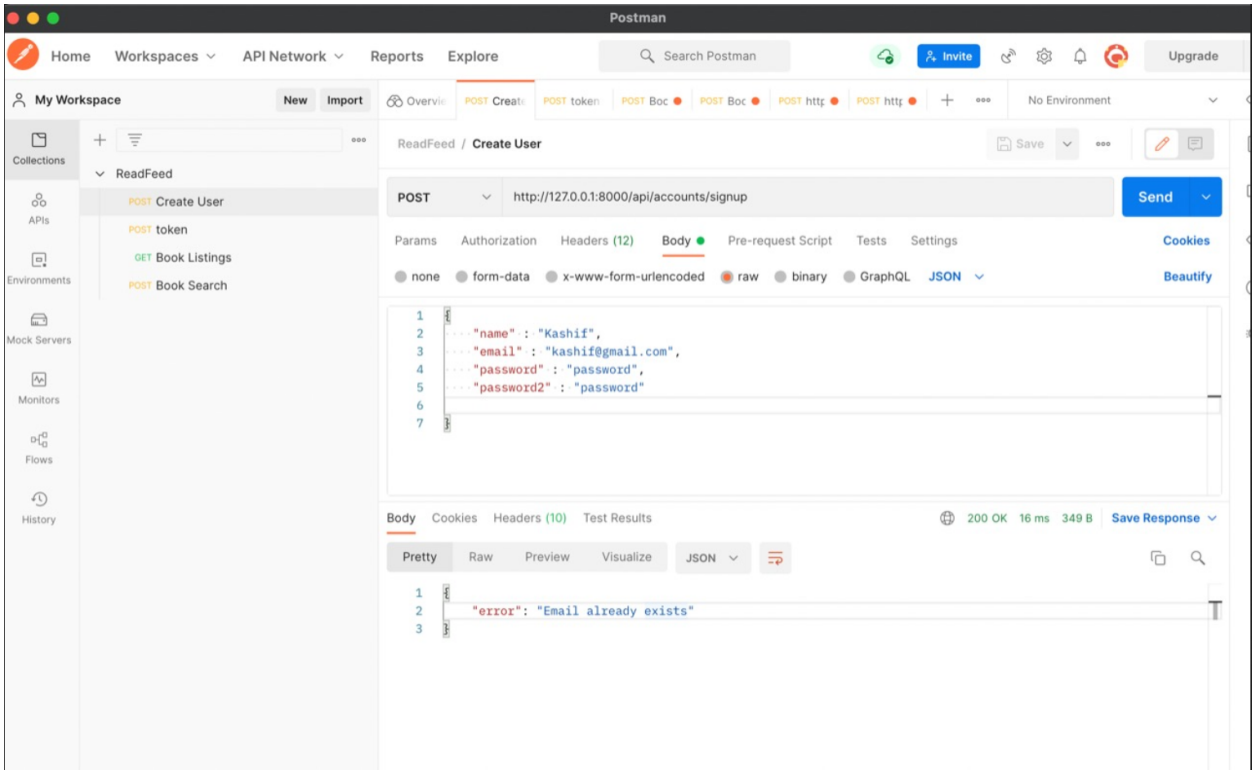


JWT Authentication:

For safely authenticating the user that is using the application.







JSON Web Token (JWT) defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA.

Below are the reasons for which we should use JWT:

- Authorization: This is the most common scenario for using JWT.

  Once the user is logged in, each subsequent request will include

  the JWT, allowing the user to access routes, services, and

  resources that are permitted with that token. Single Sign On is a

feature that widely uses JWT nowadays, because of its small overhead and its ability to be easily used across different domains.

- Information Exchange: JSON Web Tokens are a good way of securely transmitting information between parties. Because JWTs can be signed—for example, using public/private key pairs—you can be sure the senders are who they say they are. Additionally, as the signature is calculated using the header and the payload, you can also verify that the content hasn't been tampered with.

We implemented JWT using the simpleJWT library from Django.
