CMPE 275 - Enterprise Application Development

Project - NFT Marketplace

Group 2

SJSU ID	TEAM MEMBERS
016114189	Devansh Modi
015530827	Karol Josef Bustamante
016054753	Nandakishor S

Table of Contents

1. Motivations:

To learn and apply Java, SpringBoot, MySQL, ReactJS technologies to develop an NFT Marketplace Application which is scalable and reliable. We use Google Authentication with OAuth for Authentication. Java Mail for sending verification Email.

2. Introduction:

The objective of the project is to develop a NFT Marketplace Application. NFT Marketplace Application a portal for buying and selling Non-Fungible Tokens with Crypto Currencies.

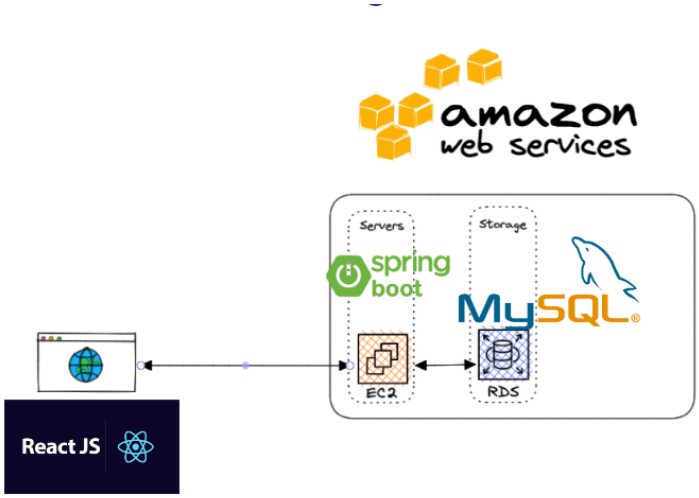
Our Application supports two crypto currencies out of the box, Bitcoin and Ethereum. However, we have designed the backend in such a way that the number of supported crypto currencies can be scaled dynamically.

The features available in our application are as follows:

1. Create and Login to your user account.
2. Manage Wallet for multiple crypto currencies.
3. Dispose/Withdraw amount from tokens
4. Create NFT tokens and publish them for standard or auction sale.

We have included validations on every API call to maintain data integrity and security through the system. The server is tested for edge cases and extreme loads. The Test plan has been implemented to verify the various scenarios.

3. System Architecture Design:



Amazon AWS

- Popular
- Experience
- Ease of use

4. Tech Stack Choices

UI

- React
  - Component Design
- MaterialUI
  - Ease of Use
  - Aesthetic

Backend

- Spring Framework
  - Spring Boot
  - POJOs
- Java Libraries
- OAuth
- JavaMail

Infrastructure

- AWS
  - EC2
  - MySQL RDS
  - API Gateway

5. High level and Component level design

Frontend: The user facing application is built using ReactJS. After a detailed analysis of the requirements, we decided to proceed with separate components. We have used React Hooks and React Routers to navigate between different functionalities and interfaces in our application.

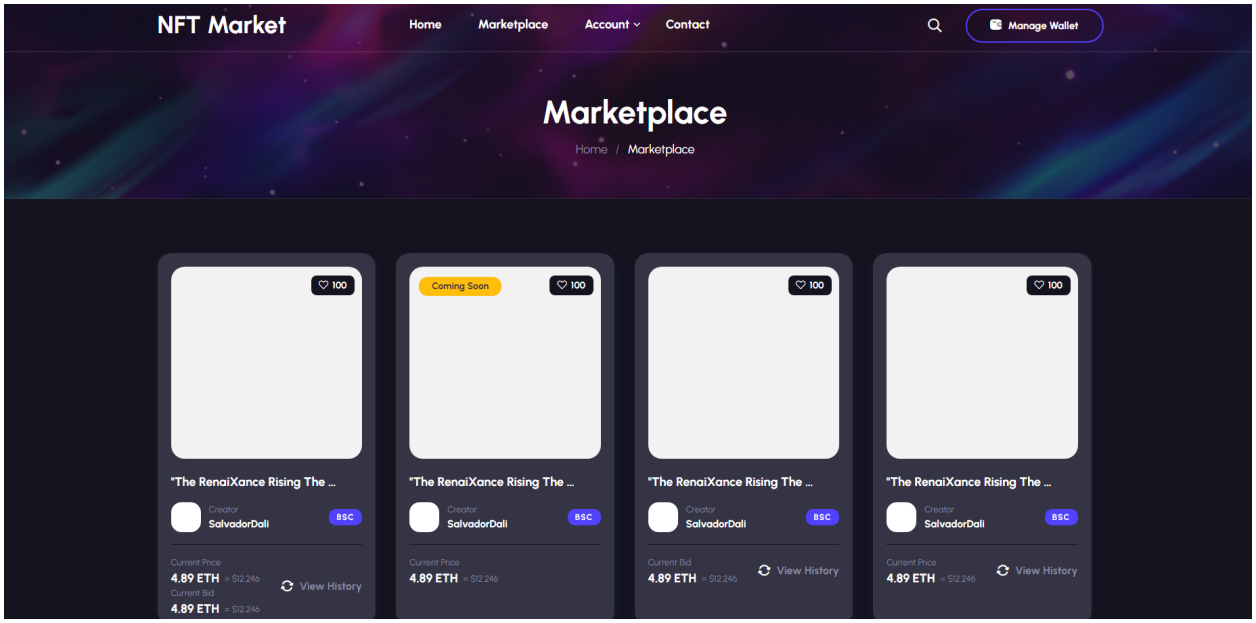
Backend: The backend is written in Spring Boot. We use Spring's RESTful interfaces for creating REST APIs. Our data models are written in POJO classes to focus on readability, reusability, and overall code maintenance. We have used the JPA to communicate with our remote database hosted in a separate AWS RDS instance.

Other Technologies:

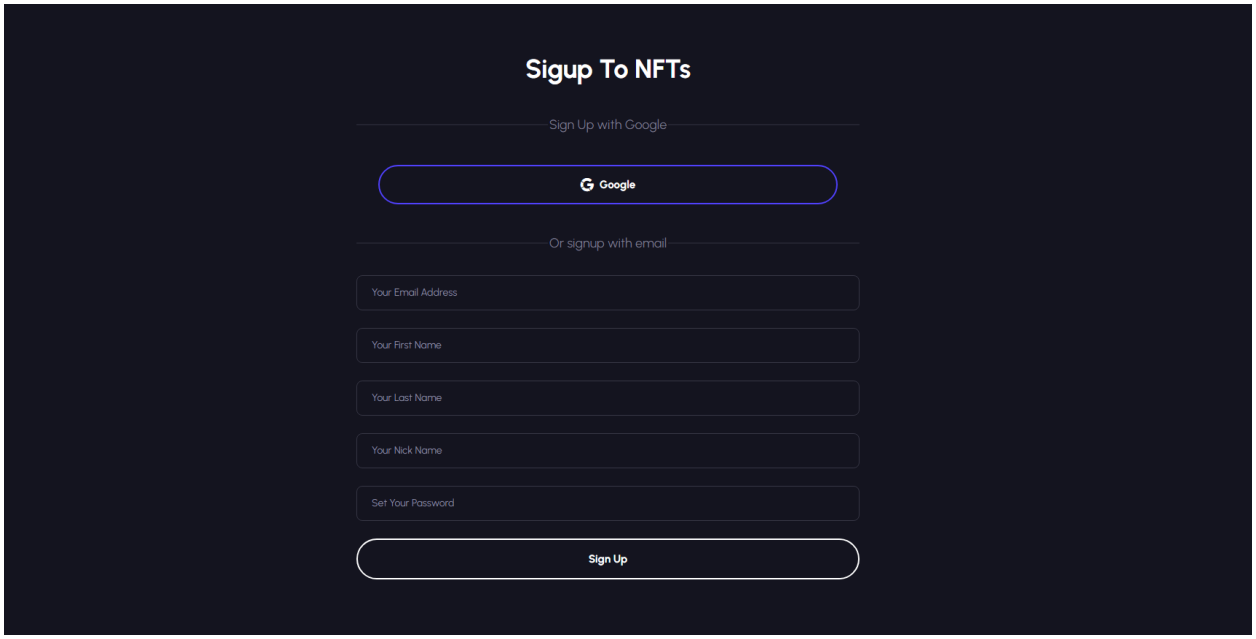
1. Java Mail: JavaMail API provides a protocol-independent method to write messaging applications. We have used JavaMail to send out verification emails to users upon first SignUp.
2. OAuth: We have additionally used Google Authentication with OAuth. It is an open standard for authentication and access delegation without requiring a password.

6. Application Screenshots

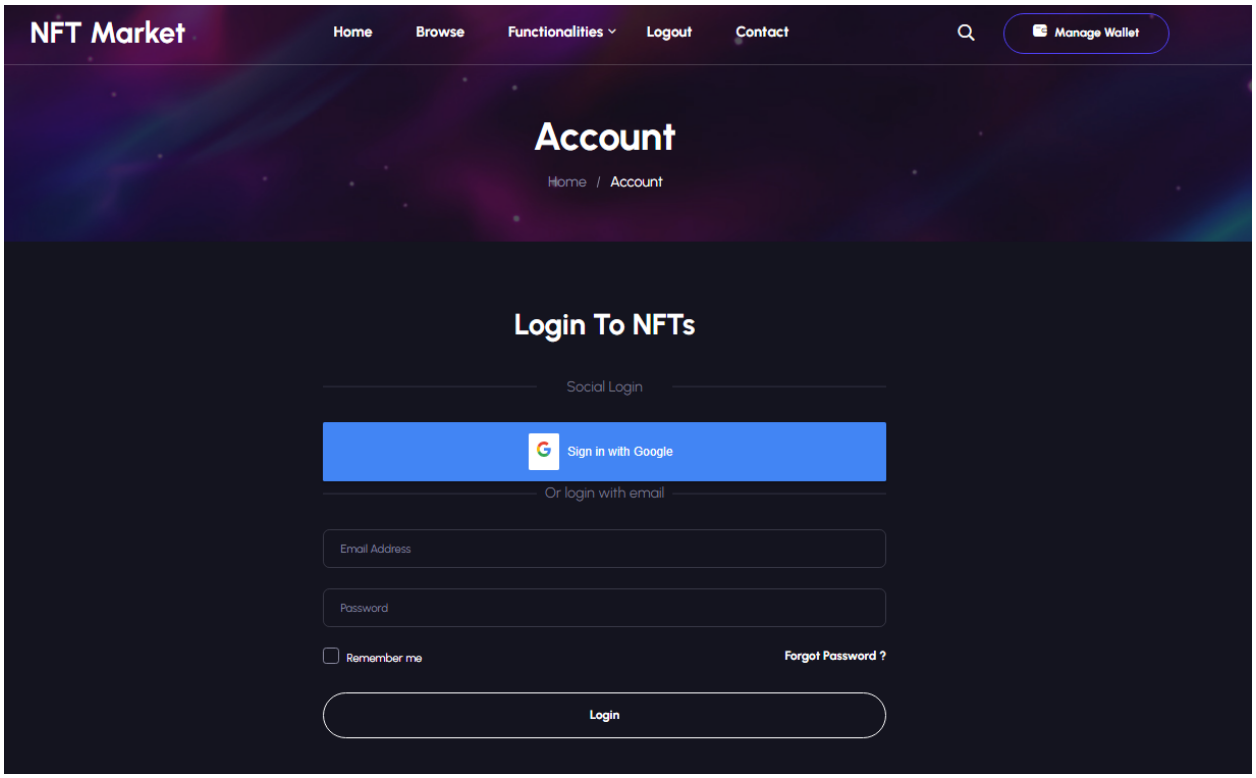
Marketplace: The main Marketplace screen provides users to view all the items available on sale. It also lets users buy a particular piece of NFT.



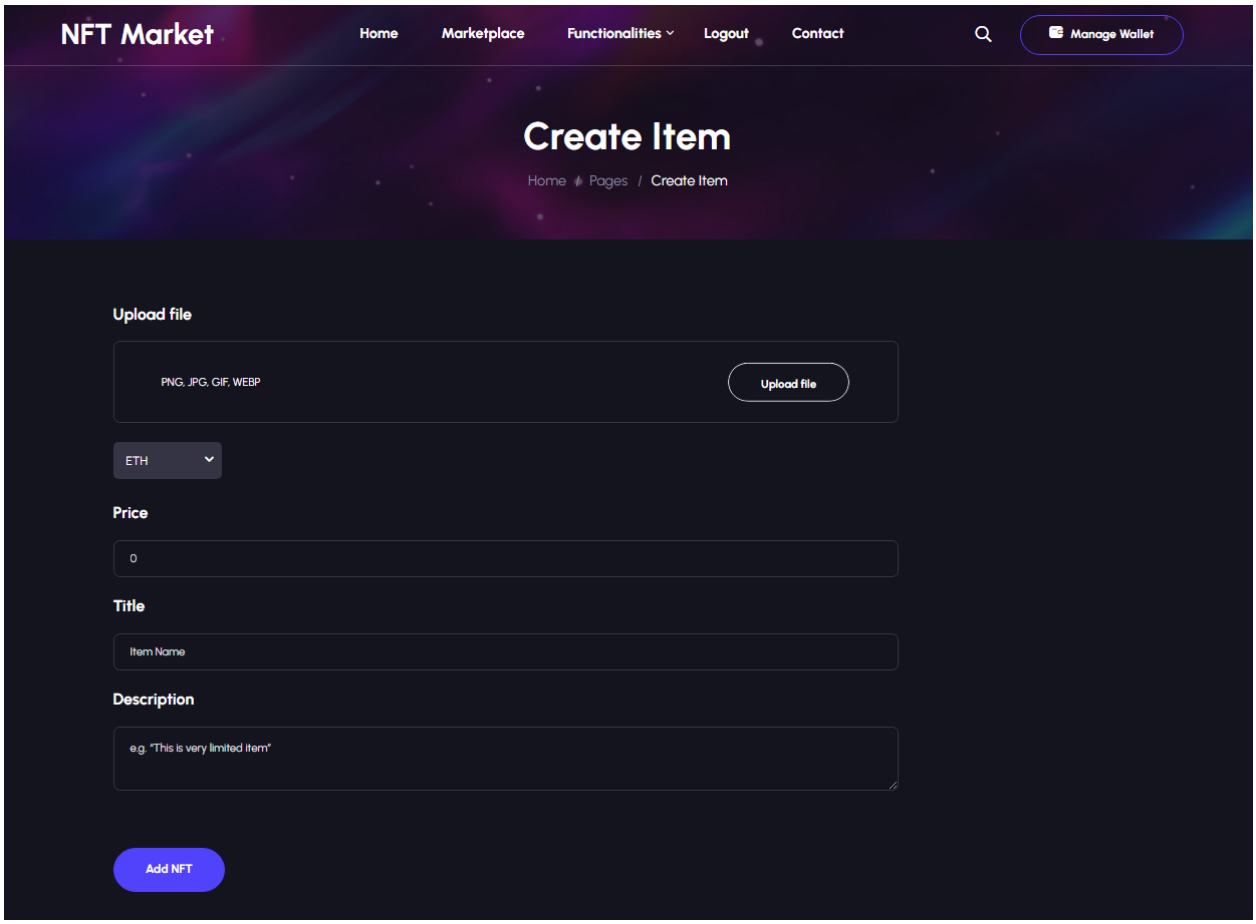
Sign Up: Two sign up options are available for the user. Google Authentication and Email authentication. A verification link is sent to the user with JavaMail.



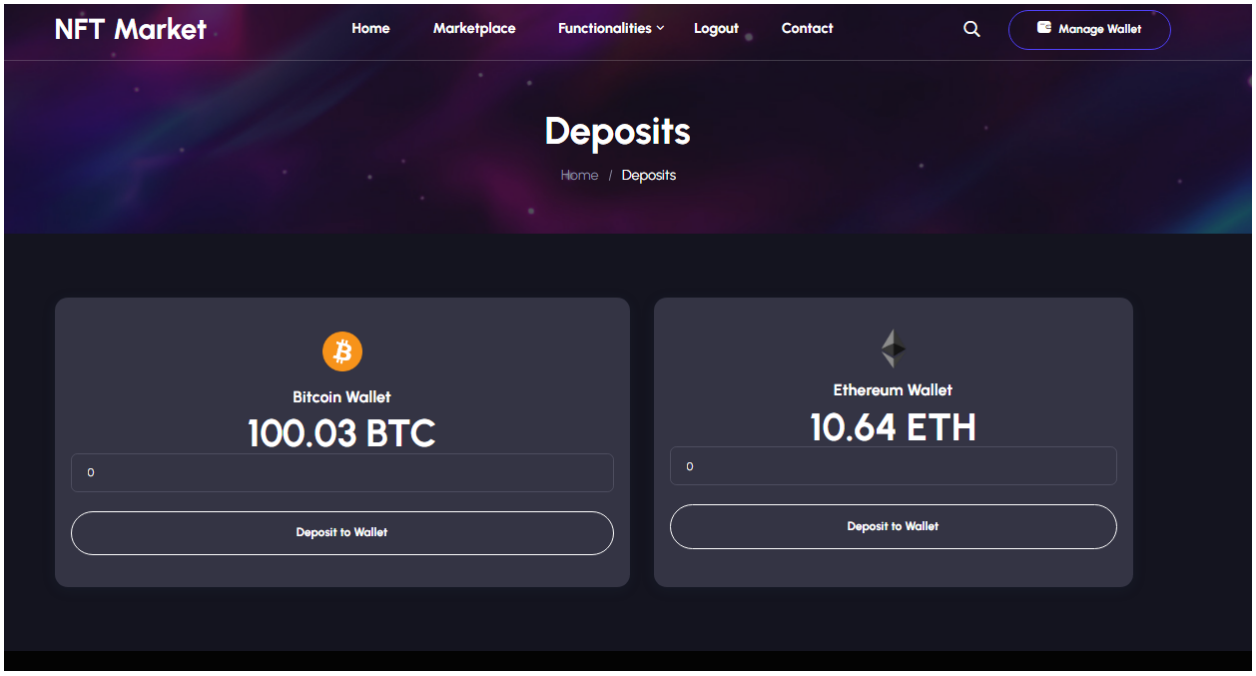
Login: Existing users can login after they have verified their email address.

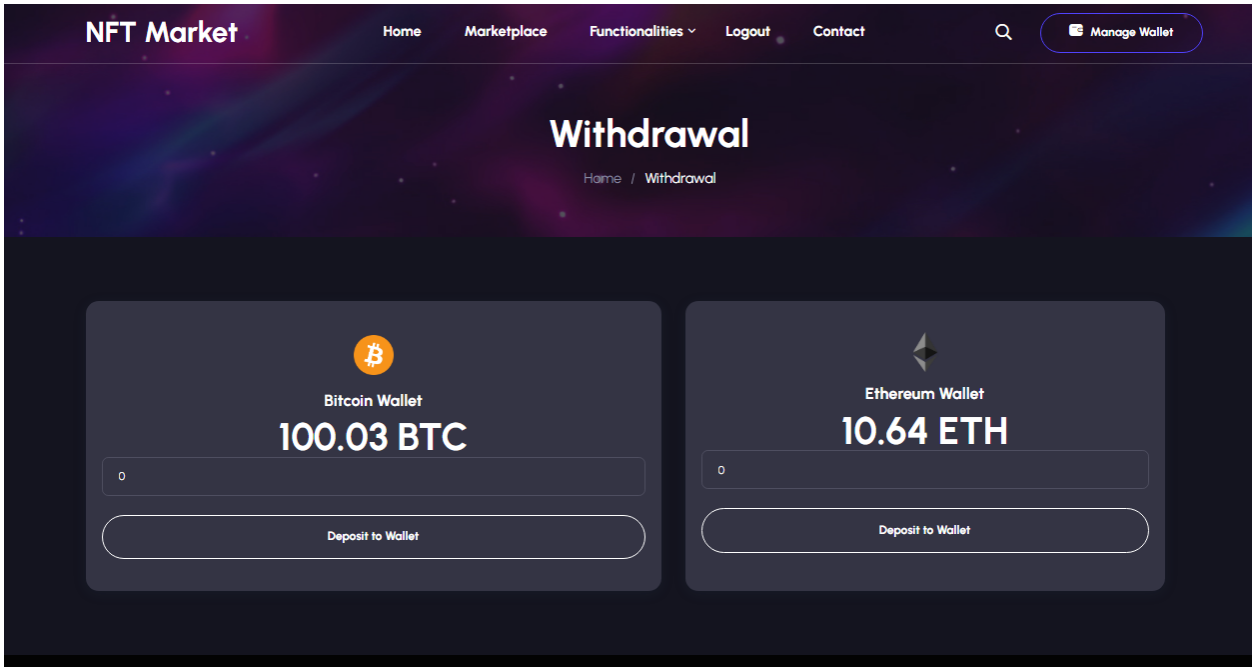


Create Item: Interface to upload your piece of art to the system. Users can set a price for the item along with a title and description.

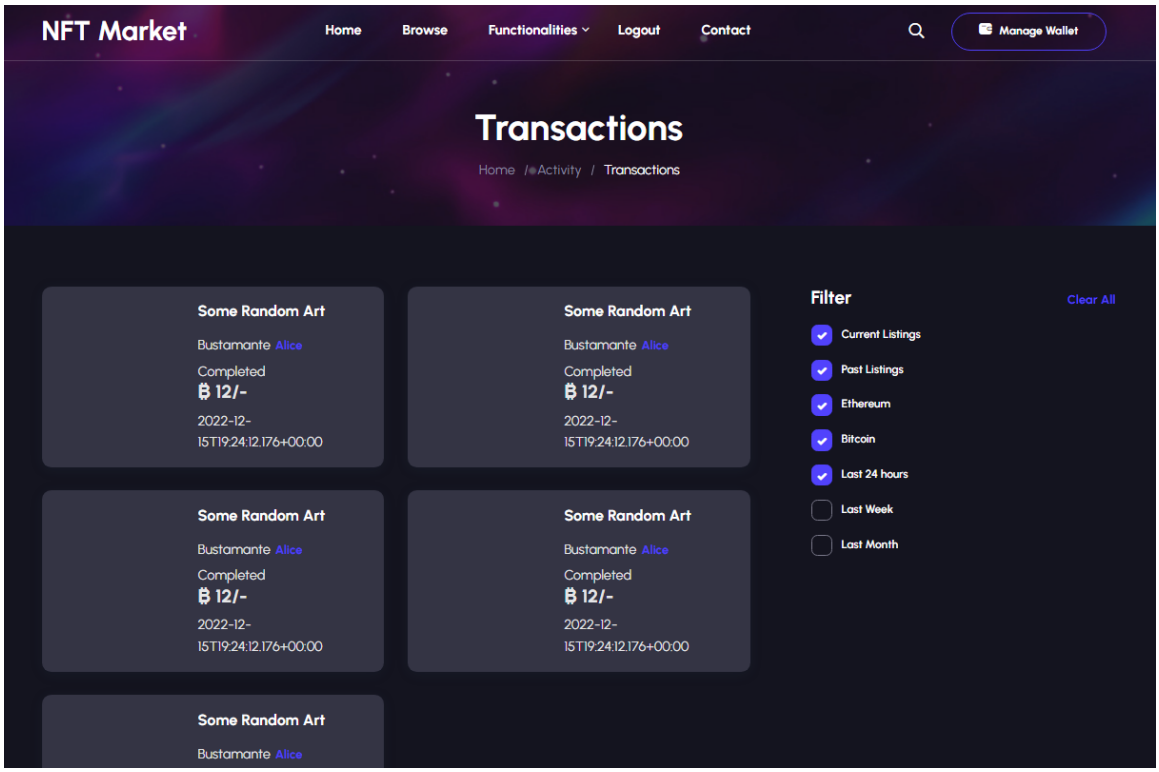


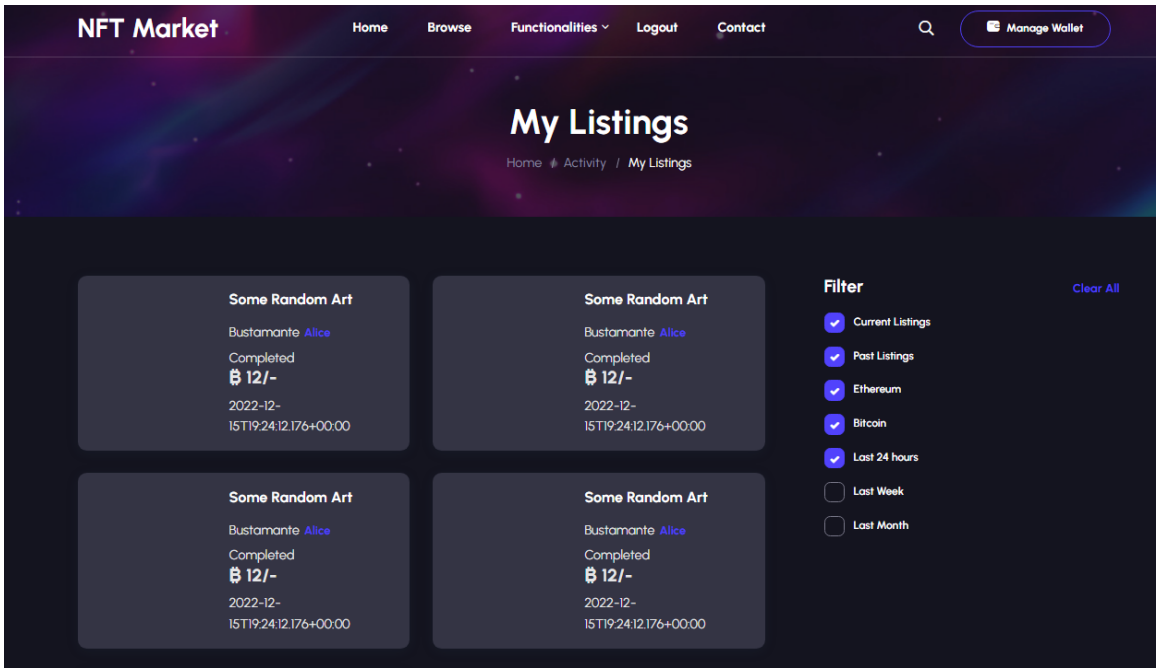
Deposits and Withdrawals: Users are provided with two wallets each out of the box. One for BTC and another for ETH. These wallets can be recharged or discharged of their balances with this interface.



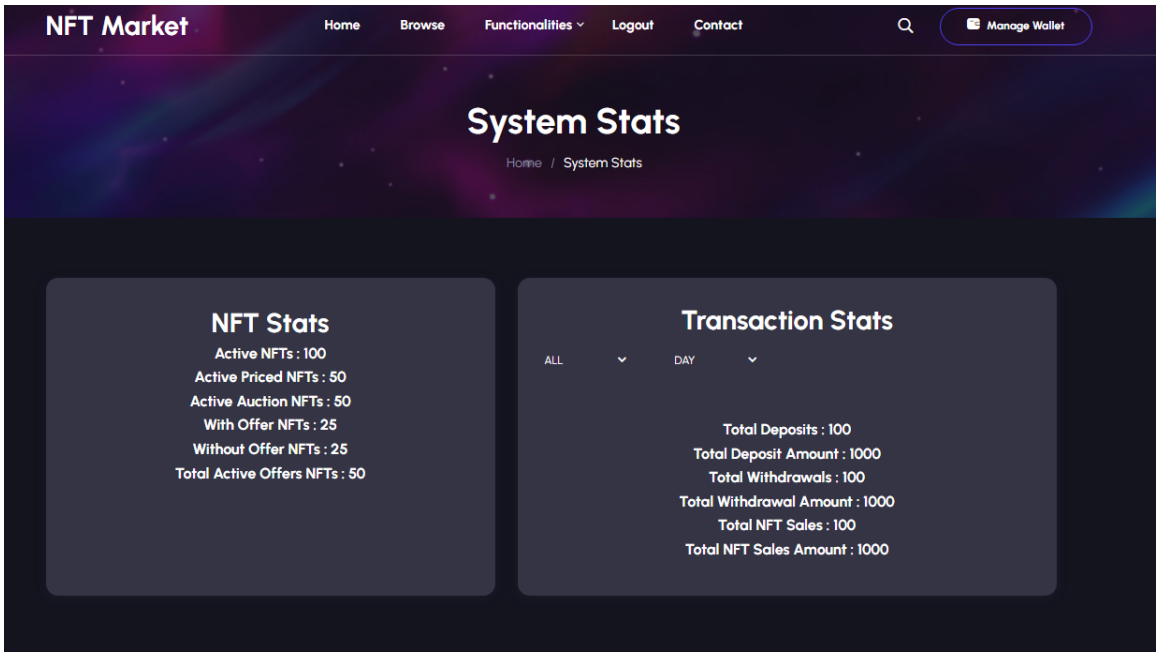


My Transactions and My Listings: The user can view their items on sale and their past transactions. We support filters to provide precise information that the user is looking for.





System Stats: Our application provides a public interface to view the total system statistics view the System Stats page.



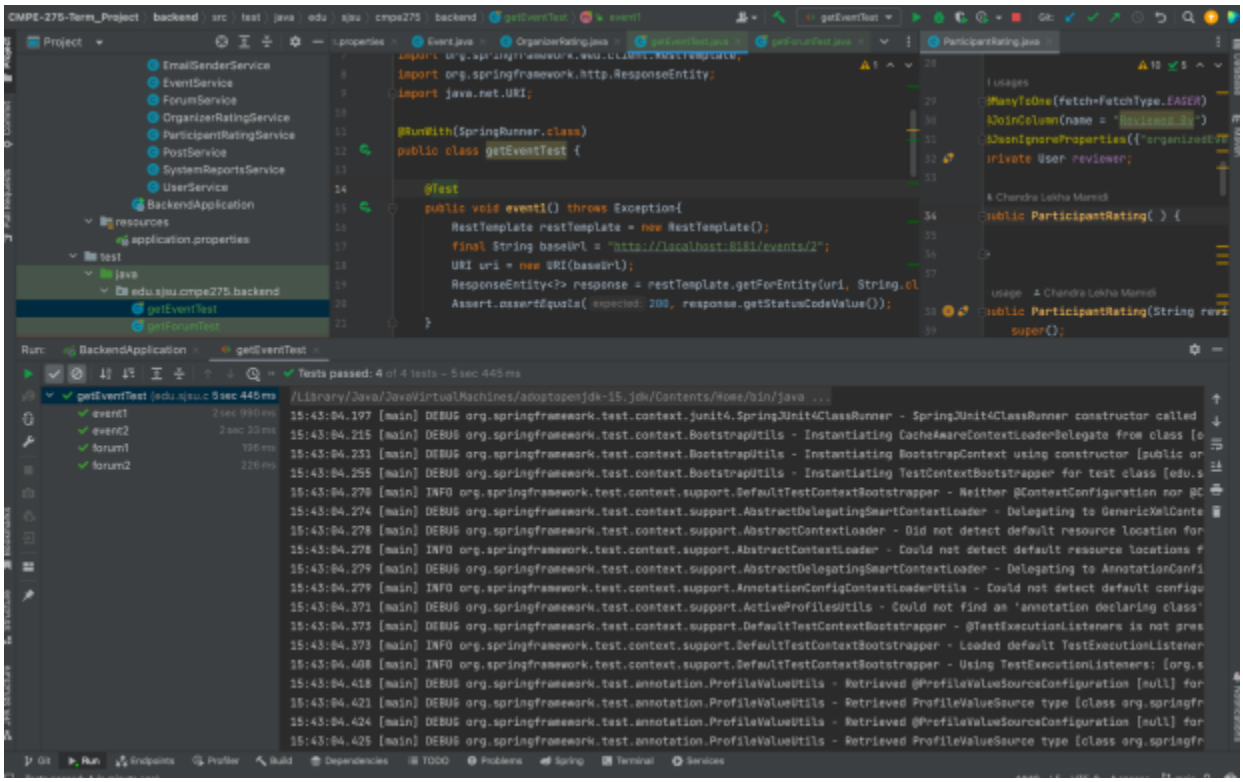
7. Testing Plan - Execution and Results

Our test plan is divided into different api components. The testing script is written in JUnit tests and performed manual testing for frontend interfaces that were hard to access. Following are the results of the same.

#Test Case	Test Case	Expected Result	Observed Result
Login and Sign up			
	Sign up for user with email and password	Verification email sent and redirect to login	Pass
	Sign up for user with Google Auth	Sign in the user directly	Pass
	Login for user with email and password	Go to system dashboard	Pass
	Login for user with Google Auth	Go to system dashboard	Pass
	Use incorrect credentials for login	Show error message as a toast	Pass
	Perform Sign up with invalid form data	Shows error message as a toast	Pass
	Perform login with invalid form data	Show error message as a toast	Pass
Marketplace			
	Perform Buy without login	Shows error for invalid access	Pass
	Perform Buy without sufficient balance in wallet	Shows error for insufficient balance	Pass
	Perform Buy with logged in account and sufficient balance	Buy successful redirects to my transaction	Pass
	Perform Buy with logged in account but insufficient balance	Show error for insufficient balance	Pass
Deposit and Withdraw			
	Perform Deposit in wallet	Show the updated wallet balance	Pass
	Perform Withdrawal from wallet	Show the updated wallet balance	Pass

My Transactions and Listing			
	Perform buy and visit my transactions	Show the newly added transaction	Pass
	Perform sell and view my listings	New sell offer is visible in my listings	Pass

JUnit Test Execution:



8. Future Improvements

- The Buy flow can be enhanced with a separate payment flow instead of a popup transaction flow.
- Interface to add connect more wallets to support multiple crypto currencies and transactions.
- Real time conversion between crypto currencies to support transactions across multiple crypto currencies and NFTs.
- Adding real time block chain support instead of mock implementation.

9. Lessons Learned and Takeaways

- Hands-on experience in building and deploying large scale applications using Spring and React.
- Implementing concepts like OOPS and AOP in tandem with each other to create a RESTful application.
- Learned to integrate third-party authentication frameworks with OAuth protocols.
- Scaling the platform with cloud infrastructure provided by AWS

10. References

[1] Spring Boot Documentation - https://docs.spring.io/spring-boot/docs
[2] React JS Documentation - https://reactjs.org/docs/getting-started.html
[3] Google Authentication and OAuth - https://developers.google.com/identity/protocols/oauth2
[4] JUnit - https://junit.org/junit5/docs/current/user-guide/
[5] Java Mail - https://javaee.github.io/javamail/docs/api/
[6] Amazon Web Services Documentation - https://docs.aws.amazon.com