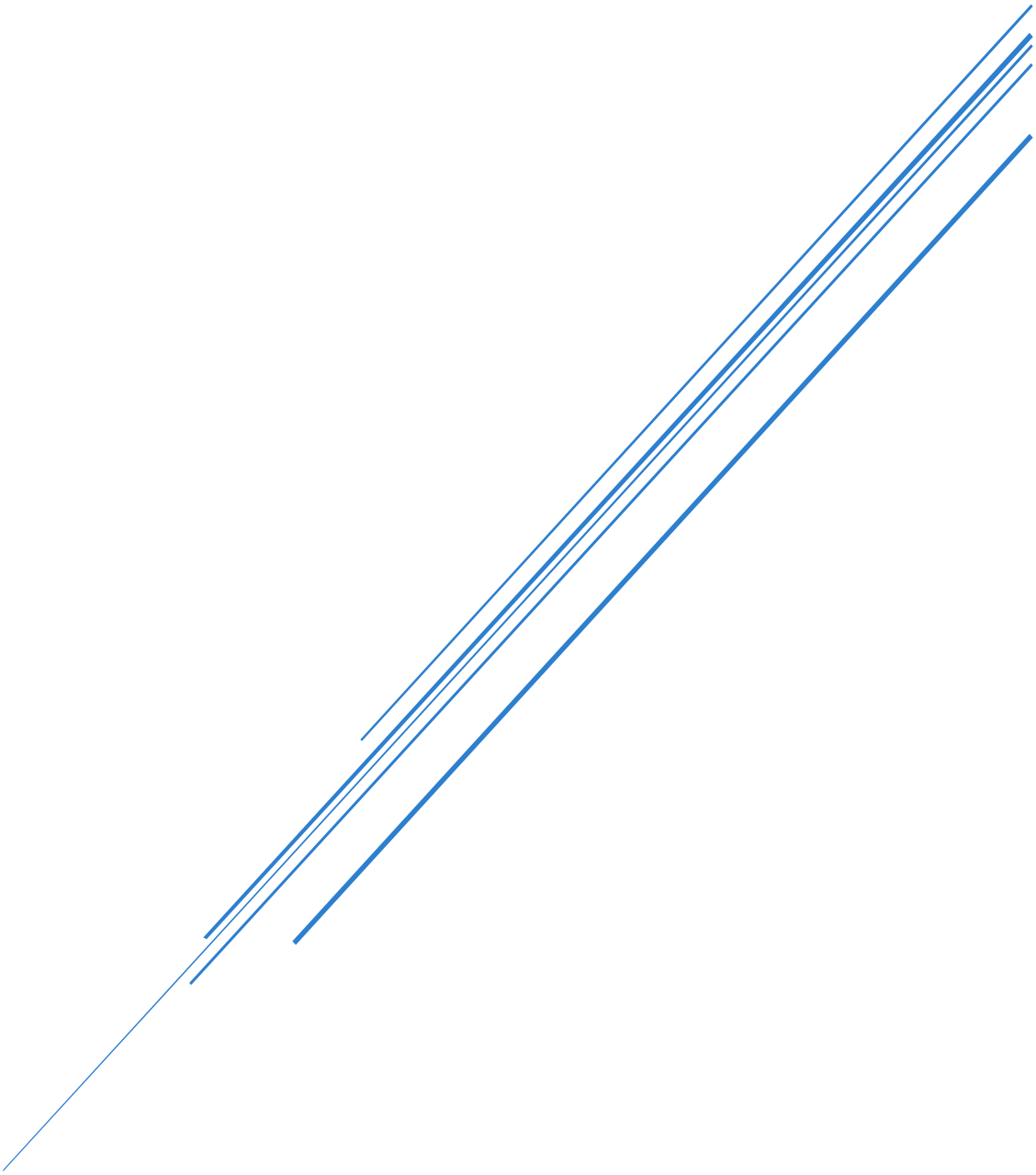
INDIEVAULT

Final Project - COS30043



Rupayan Banerjee
Student ID - 103538229

Table of Contents

Main Functionality

The application is a modern web platform designed to showcase indie games. Users can browse, search, filter, and interact with game listings as well as reorder them using a draggable interface. Registered users have additional interactive features, such as liking games, adding them to a wishlist, and writing reviews. Administrators possess extra capabilities, including adding, editing and deleting games on the fly. Users also have the option to see the games they have wishlisted on their profile. Apart from that, there's a basic home page for welcoming a user, a news page to display news and an about page that has been crafted according to the project brief.

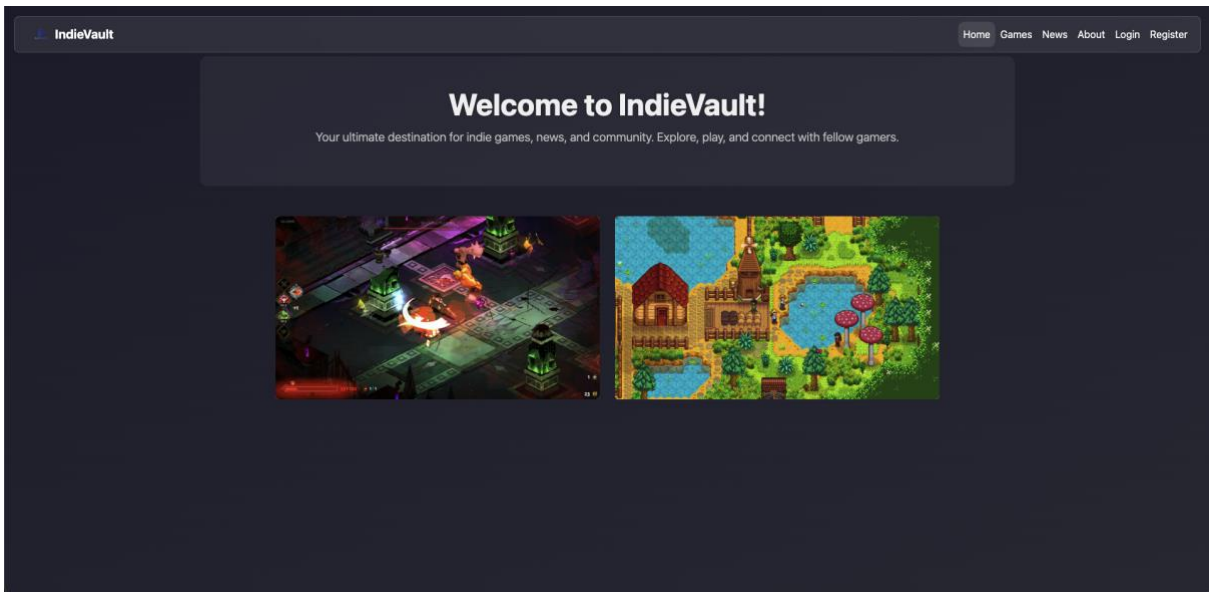


Fig. 1Root page for the website

Technical Components and Tools

- **Vue 3**: Used as the frontend JavaScript framework, providing reactive data binding and component-based architecture. The Vue environment has been created using Vite package manager.
- **Bootstrap**: Implemented for responsive and aesthetically appealing UI components.
- **LocalStorage API**: Employed for client-side data persistence, enabling user interactions and preferences to persist across sessions.
- **Vue Router**: Facilitates client-side routing, enabling seamless navigation between pages.
- **Custom Vue Composables**:
    - **useDebounce**: Provides real-time, debounced search functionality.
    - **useDragReorder**: Enables interactive drag-and-drop reordering using the vuedraggable library.

- **Vuedraggable**: Integrated for intuitive drag-and-drop sorting of game items.
- **JSON Data Storage**: Utilized to simulate backend data for games and news content.

Extended Functionality Description

As mentioned earlier, the website is an indie game hosting platform. I will go over the pages(views) and their features and functions in this section.

- **Home**: The root page which loads up when we launch the website. It is very simple with moderate styling and a big header welcoming the user along with 2 pictures showcasing 2 popular indie games' gameplay. The images pop out a bit when we hover the cursor over them as well. Also, we have a view of the persistent navbar that is translucent and persistent on all views of the website.

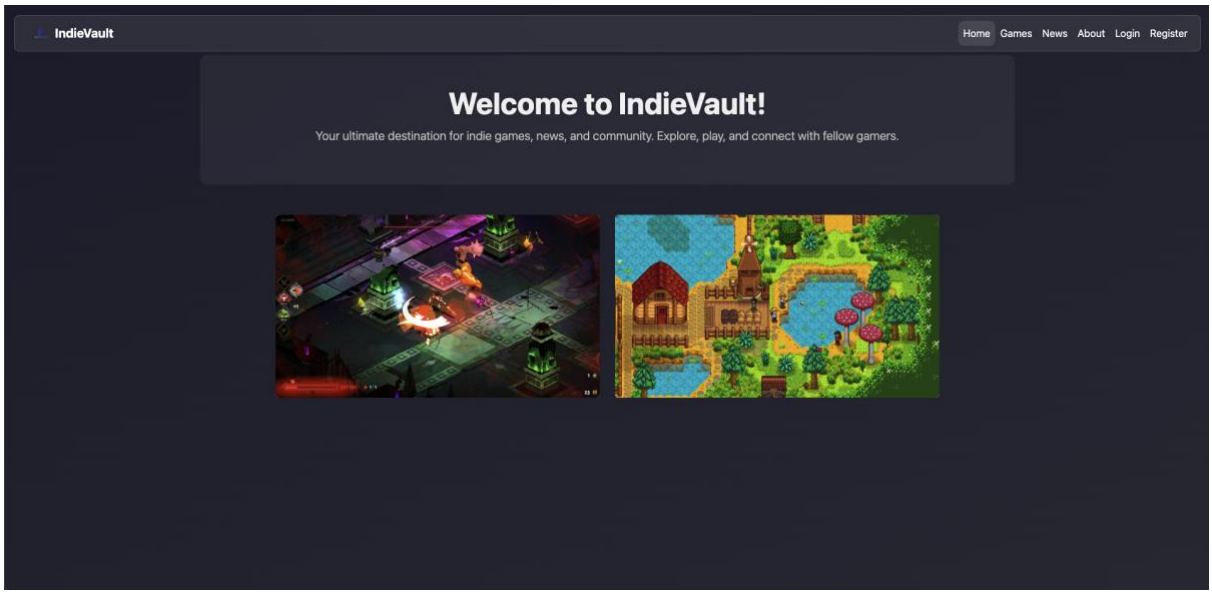


Fig. 2 Home View

- **Games**: This is the most code-heavy part of the whole website. This page is a combination of a lot of different things. It is essentially there to display all the games with their cover arts that it extracts out of a pre-defined json file and then stores the data in the local storage to make it dynamic. Users can search through the games with the included search bar as well as filter them out based on their genre. They can also re-order the games by simply dragging their cards. There's also pagination included. Once logged in, regular users are also able to like any game they want as well as add any to their wishlist. They can also leave a review if they want and everything would stay persistent. Admin users are additionally able to add new games into the mix, alongside having the power to edit and delete any existing ones. The search bar here features debounced search which waits till the user stops typing to then show the results saving resources in the process. All these changes

are made persistent by storing them in the browser local storage via unique arrays and retrieving the data from them later.

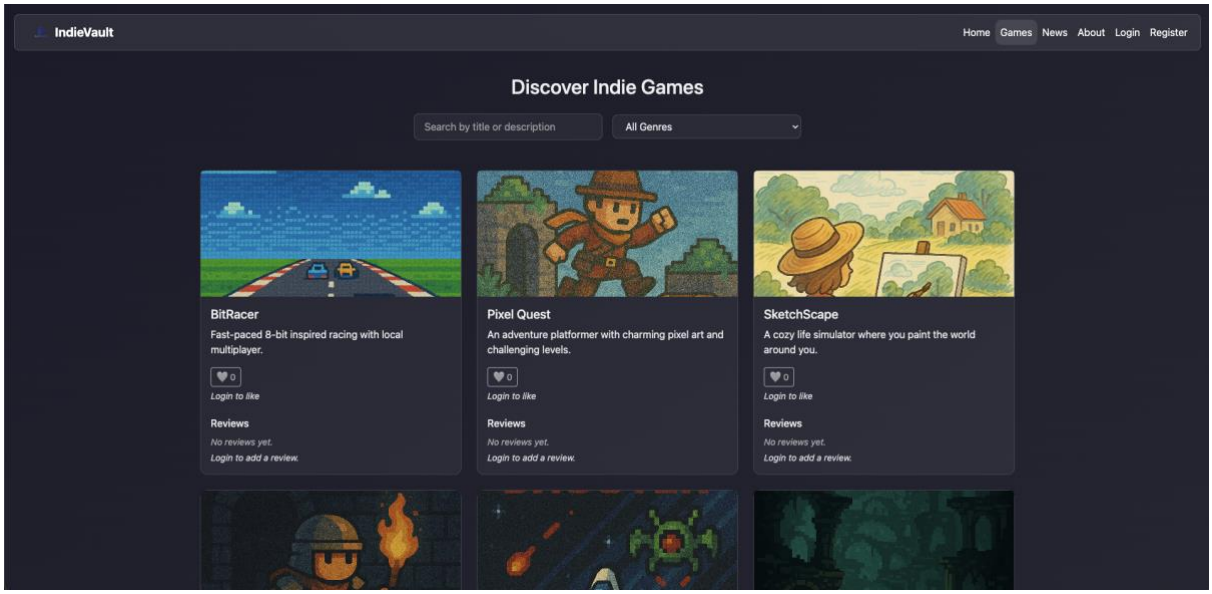


Fig. 3 Games View

- **News**: This is a simpler page, which again included the same debounced search feature and the pagination options. This page loads a pre-defined set of news articles from a json file and displays them to the user. Users are obviously able to scroll through the pages and search the news based on any field of their choice.

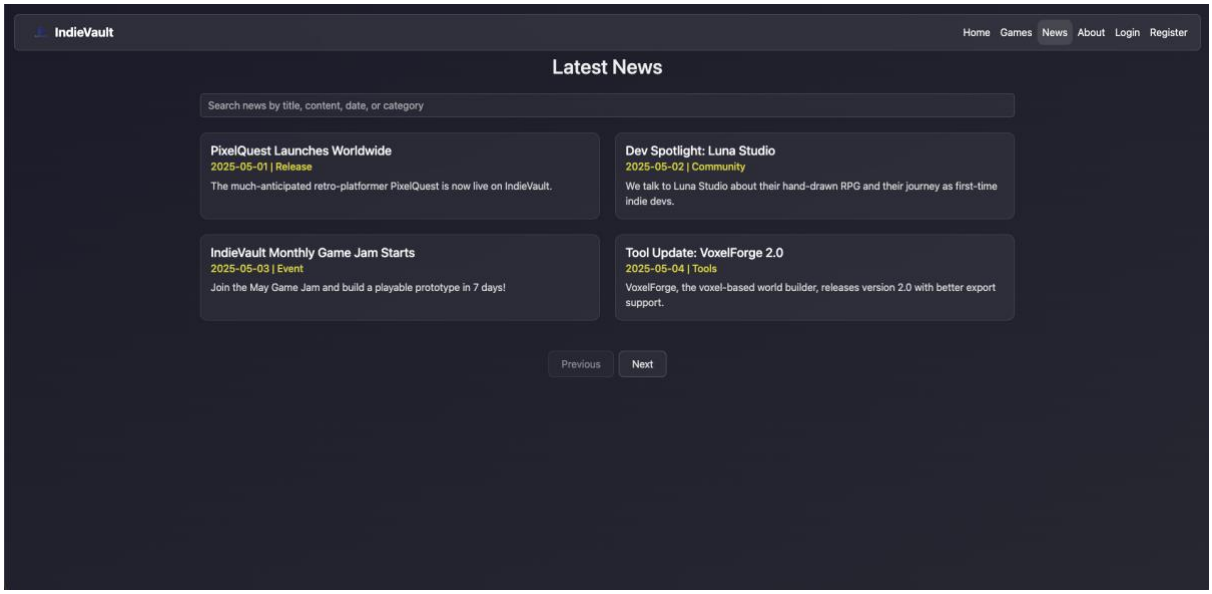


Fig. 4 News View

- **About**: This is more of a vue feature testing page where there's a simple form that asks the user for their name and dynamically shows them a greeting message. It also has 3 radio buttons to show the poster of the user's favourite indie game.

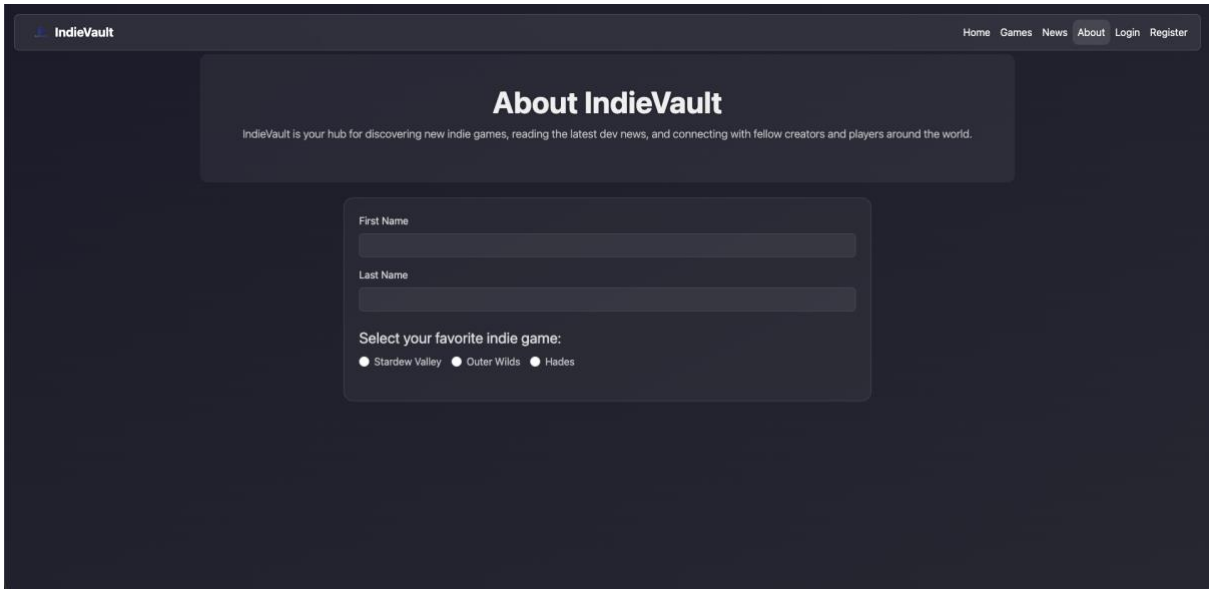


Fig. 5 About View

- **Login**: This is the page which allows users, who already have an account setup, login to the website to use the other features. It relies on an array that holds all the user details who sign up for the website. Once logged in, it redirects to the home page and the navbar buttons for login and register give way to a profile button displaying the user's name and a logout button.

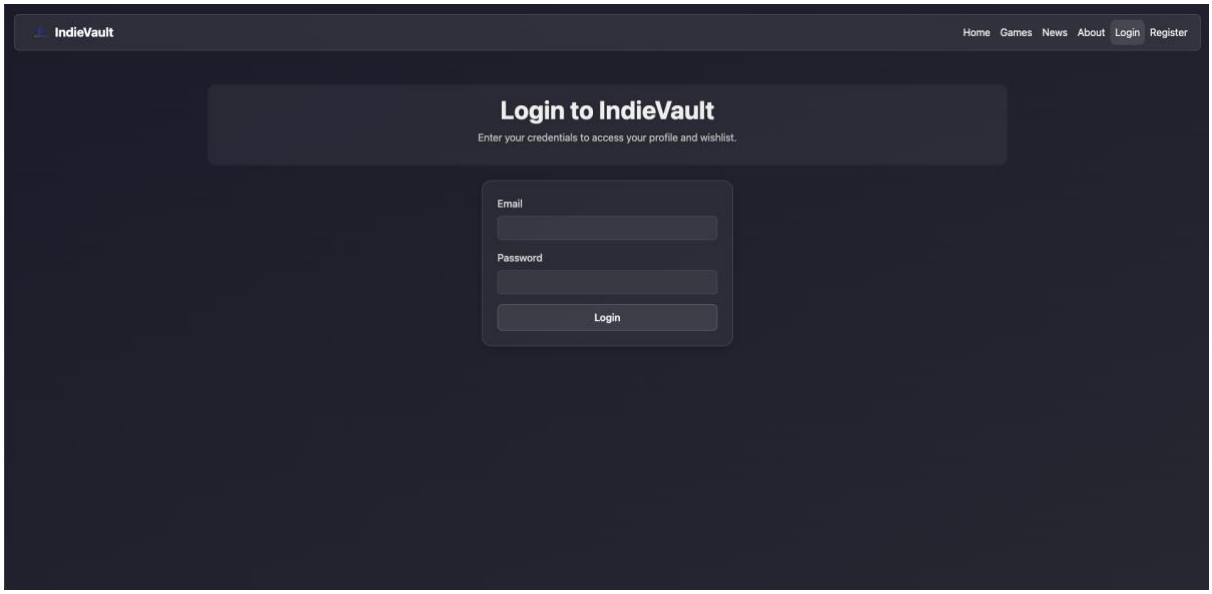


Fig. 6 Login View

- **Register**: This page helps a new user sign up to the website to access all the features locked out to non signed in users, like liking, wishlisting and reviewing games. It also asks the user if they are a regular user or an admin since both accounts have specific level of control over the website. Once the details are fed in the form, which has a decent level of form verification, saves it to the array that stores all the other users' data as well.

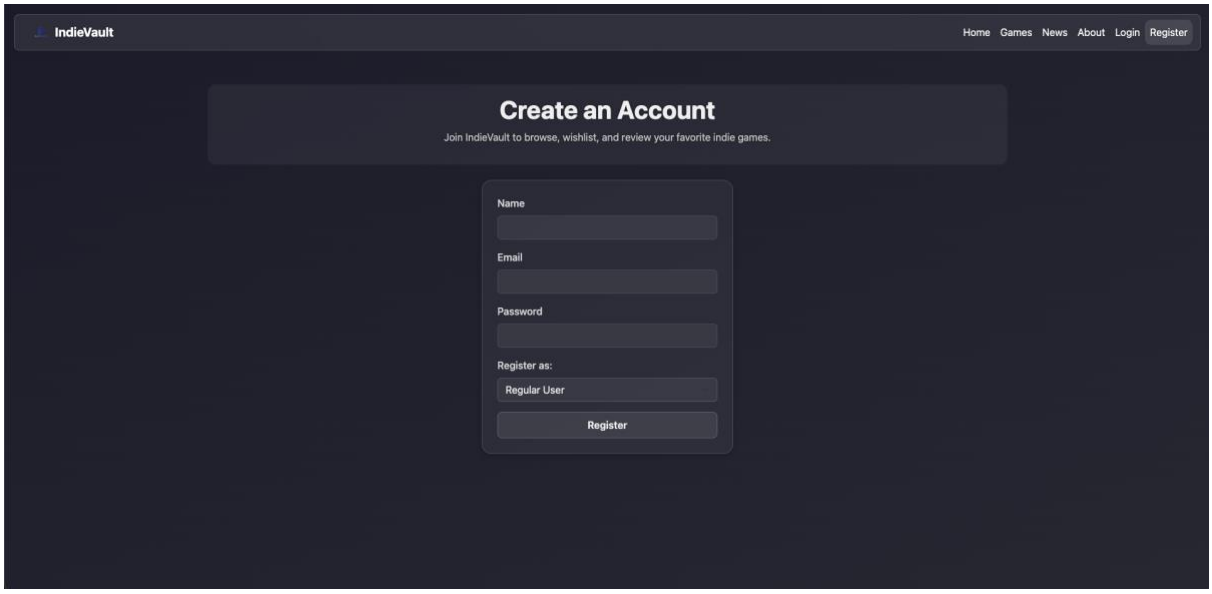


Fig. 7 Register View

- **Profile**: This page can be accessed upon clicking the button showing the username on the navbar. It essentially greets the user and shows them their list of wishlisted games pulling from the array built within the games page.

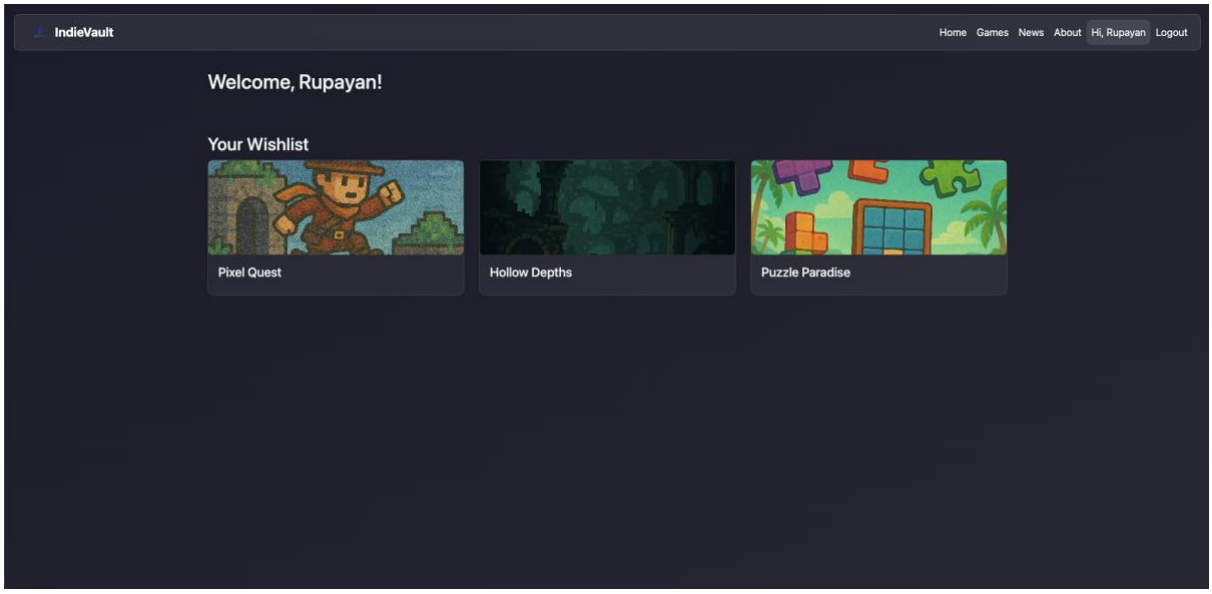


Fig. 8 Profile View

Innovative Features

- **Real-time Debounced Search**: Implements a reactive, user-friendly search feature that reduces redundant processing by debouncing user input, enhancing performance and user experience.
- **Drag-and-Drop Reordering**: Allows users to dynamically manage the order of displayed games directly within the user interface, significantly improving usability and management efficiency.

- **Composable Architecture**: Adopted custom composables to encapsulate complex state logic and enhance code reusability and maintainability.
- **User-specific Wishlist and Likes**: Personalization of user experience by maintaining separate wishlists and liked game statuses tied to individual user profiles.

Challenges Faced and Solutions

There were quite a few challenges that I faced when developing this app. The stage 1 implementation was straight forward and didn't create any issues. With stage 2, the main issue was making the website stay persistent and store the data locally. I had to go through a lot of online tutorials to make sure I am not missing out on anything. There were times when the entire implementation was right, but the website would not load because of a conflict between different script definitions.

Another major challenge was making the like and wishlist status linked to each user so that everyone can only like a game once and wishlist it once. Initially any user would be able to just click on the like button multiple times and be able to rack up that number. This was fixed again by carefully making dynamic arrays in the local storage for every signed-up user, and pulling data from those arrays depending on who is logged in.

The next major challenge was when I was implementing stage 3 features. Integrating drag-and-drop functionality posed unexpected issues with state synchronization, particularly affecting filtering and pagination logic. This was resolved by implementing a writable computed property to seamlessly manage both UI interactivity and underlying state integrity.

Another challenge was the implementation of real-time debounced search. It initially conflicted with pagination, as rapid state changes caused UI inconsistencies. This was effectively managed by encapsulating search logic within a debounced composable, which made the UI experience better.