Computer Science 343

Project 2

Members:

*Louis De Jager [24908134@sun.ac.za]*

*Hardus Viviers [24917540@sun.ac.za]*

*Stefan Russouw [24715034@sun.ac.za]*

*Jurianne Schreuder [24695351@sun.ac.za]*

*Migael Van Wyk [24716812@sun.ac.za]*

Contents

[Introduction 1](#_Toc148391049)

[User-Interface 1](#_Toc148391050)

[Login page 1](#_Toc148391051)

[Home page 2](#_Toc148391052)

[Note Management Side Bar 2](#_Toc148391053)

[Main text area 2](#_Toc148391054)

[Top features 2](#_Toc148391055)

[Use Case Diagram: 3](#_Toc148391056)

[API 4](#_Toc148391057)

[User Authentication 4](#_Toc148391058)

[Database operations (CRUD) 4](#_Toc148391059)

[Real-Time web sockets 4](#_Toc148391060)

[Database 5](#_Toc148391061)

[Description of design patterns for the client and API 5](#_Toc148391062)

[Client: 5](#_Toc148391063)

[API: 6](#_Toc148391064)

[Contribution Summary: 6](#_Toc148391065)

# Introduction

In this project, we've developed an innovative Markdown note-taking web application called "Anvil." Anvil boasts real-time collaboration capabilities, allowing multiple users to seamlessly work together via WebSockets. Our user interface is powered by React.js and elegantly styled with Tailwind CSS, ensuring a user-friendly and visually appealing experience. Additionally, we've employed Node.js in conjunction with Express for our API. This document provides a comprehensive overview of our application and a high-level insight into the design patterns employed in both the client and API components.

# User-Interface

## Login page

The login page is composed of two distinct screens: a login screen and a registration screen. The login screen provides users with input fields for their username and password, along with convenient features like a "Remember Me" option for persistent login sessions and a "Forgot Password" function for password retrieval.

On the other hand, the registration page offers users the ability to create a new account with essential information, including a unique email address, username, and password. Additionally, users can customize their profiles during the registration process.

# Home page

## Note Management Side Bar

Within the home page, a user-friendly left sidebar stands out with two primary tabs: "My Files" and "Shared Files." You can easily initiate a new file creation at the bottom of this sidebar and assign it to a specific category. The top section of the sidebar hosts a practical search and filtering function for your notes. Furthermore, the individual notes themselves offer convenient options to share and delete content. When you venture into the "Shared" tab, you'll find an overview of notes that have been shared with you.

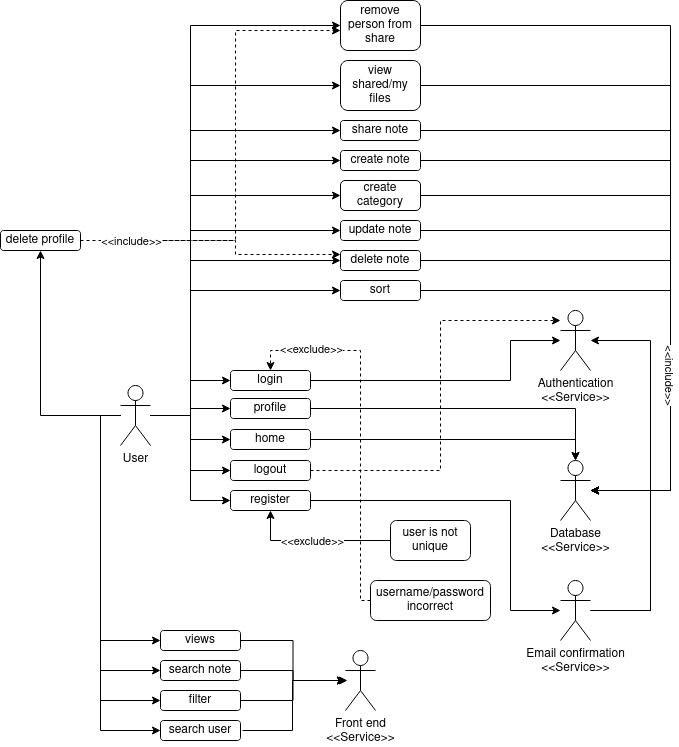
## Main text area

Adjacent to the sidebar, you'll discover the central hub for markdown and note editing. This area is divided into three dedicated tabs. The first tab is dedicated to editing and viewing your markdown text, while the second tab offers a live preview of the results generated by your markdown script. The third tab ingeniously presents your markdown text and the corresponding results side by side, allowing for a dynamic and real-time experience as you create your markdown content.

## Top features

Located above the main text area, you'll find a suite of top-notch features. Here, you can easily monitor which users are currently viewing and editing the note being displayed. Additionally, a "Profile" button awaits, providing access to two essential pages: "Edit Profile" and "Delete Profile." The "Edit Profile" page empowers you to update your profile picture, username, email, and password, ensuring that your personal information remains up to date and secure.

## Use Case Diagram:



API

## User Authentication

We have implemented a robust and secure login system that includes username and password verification. Users have the ability to create a new account with their unique username and password. Moreover, our system provides account verification and password reset functionalities. User data, including creation dates, is securely stored in the database for future reference.

To ensure the utmost security, passwords entered during the signup process are hashed using the bcrypt library. This hashing process employs the Blowfish algorithm and a unique generated salt to further encrypt the password, making it nearly impossible for unauthorized access.

For returning users, the system seamlessly retrieves the encrypted password and uses the same algorithm to decrypt it, granting them access to their accounts without compromising security.  
  
JWT After Authentication

We employ a robust JWT library for handling user authentication. After a user successfully signs in, a JWT token is generated using a secret key known only to the backend. This token contains the user's ID and an expiration date, which is particularly useful for "remember me" functionality.

## Database operations (CRUD)

Our application relies on SQL statements and relational tables, which are categorized into 'users,' 'notes,' and 'shared.' The core database operations we perform are Create, Read, Update, and Delete (CRUD) of data. To enhance security, we have implemented query sanitization to prevent SQL injection attacks for the 'notes' table. Additionally, we enforce restrictions on user input for 'users,' disallowing HTML tags in usernames.

## Real-Time web sockets

In order to facilitate collaborative note-sharing among users, we have implemented websockets using the Socket.IO library. When users connect to our application, they utilize the frontend Socket.IO client library. Each user is assigned a unique socket ID, and when a user makes edits to a shared note, they are automatically placed in a corresponding socket room associated with their ID. This design ensures that all editors of a particular note are in the same room, allowing for seamless real-time updates to be broadcasted to all users within that room.

# Database

Our database is hosted in a serverless PostgreSQL environment called Neon. Neon enables us to execute SQL queries and visually examine tables. Moreover, we have carefully normalized our tables to adhere to the third normal form (3NF), which helps in reducing data redundancy and maintaining data integrity.

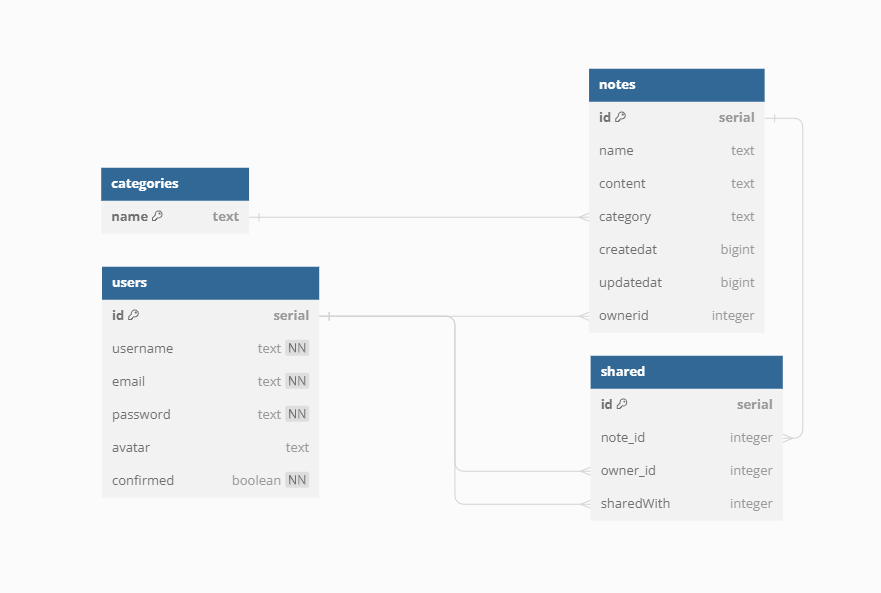


Figure 2

# Description of design patterns for the client and API

## Client:

We have adopted a component-based architecture for the client, which makes our system more maintainable and scalable. This design pattern allows us to break down the user interface into smaller, manageable components.

To facilitate real-time data synchronization, we've implemented a state management pattern. This pattern enables us to efficiently share notes across different users and ensure that updates are synchronized in real-time.

## API:

Our API follows a RESTful design pattern, which leverages HTTP methods such as GET, POST, and DELETE to perform various database operations like create, read, update, and delete. This approach makes our API user-friendly and predictable.

Real-time collaboration between users and their notes is achieved through WebSocket integration. This allows for immediate communication and collaboration in a synchronous manner.

Database normalization is a key aspect of our design. It minimizes data redundancy and enhances data integrity, ensuring our data is organized and efficient.

# Bonus Features

Account Registration Verification via Email: We've implemented a robust email system that ensures the security and authenticity of our users. When you register, you'll receive an activation link in your email, which you can click to verify your account. This extra layer of security helps protects your information.

Password Reset Functionality: For those times when you forget your password, we've got you covered. Our system includes a user-friendly password reset feature. Just follow the prompts, and you'll be back in your account in no time.

Automated Testing with GitLab CI/CD: We take quality seriously. Our application is equipped with a suite of unit tests integrated into the GitLab CI/CD system. This means that any changes or updates to the application are thoroughly tested, ensuring a stable and reliable experience for our users.

Public-Facing Server Hosting: To make our application accessible to a wider audience, we've hosted it on a public-facing server. This means you can access our services from anywhere, without the need for complex setups or VPNs.

# Contribution Summary:

*Contributions consisted of a lot of pair programming*:

Louis De Jager [24908134@sun.ac.za]: Louis contributed more on the back-end side. Worked with the API. As well as SQL queries and web sockets. Also did the login and authentications (JWT).

Hardus Viviers [24917540@sun.ac.za]: Worked and created with the GUI and was managing the tailwind and CSS. Created majority of the components. Created S3 Bucket for externally storing avatars online.

Stefan Russouw [24715034@sun.ac.za]: Stefan also worked on the back-end side. Also worked with API and created routes. Integration of front-end and back-end. Created the forgot password bonus feature.

Jurianne Schreuder [24695351@sun.ac.za]: Helped on database creation. Wrote the report. Created CI pipelines/tests for testing the program and finding bugs.

Migael Van Wyk [24716812@sun.ac.za]: Created the database and SQL queries for creation and design of database using Neon. Also created test data queries and deployed front-end and back-end to render.