
UNIVERSITATEA DIN BUCUREȘTI

FACULTATEA
DE
MATEMATICĂ ȘI INFORMATICĂ


SPECIALIZAREA INFORMATICĂ

# Lucrare de licență
# CODEX
# READING COMPANION

Absolvent
Miruna-Bianca Georgescu

Coordonator ştiinţific
Lect. Univ. Dr. Marius-Iulian Mihailescu

București, iulie 2024

# Abstract

Codex is a web application designed for readers. The goal of Codex is to enhance the reading experience by offering users a platform that is more than the average reading tracker. Developed using the ASP.NET MVC framework, Razor Pages and Bootstrap, Codex is a place where users can not only log their reading activity, but also personalize their profiles and join in challenges and buddy reads.

On the platform, users can create accounts and then customize their profiles to their liking by showcasing their favorite books and the badges they earned for their reading. In order to track their reading, users can join the reading challenge and set yearly goals, as well as read over 30 pages everyday to keep their streak going. Furthermore, Codex has a lot of social features that allow users to connect, fostering a sense of community. These features include Buddy Reads and Friends Quests, alongside the usual reading tracker features such as the ratings and reviews.

Through all its features, Codex aims to encourage people to make reading part of their daily life and share their reading journeys with others.

# Rezumat

Codex este o aplicație web concepută pentru cititori. Scopul Codex este de a îmbunătăți experiența de lectură, oferind utilizatorilor o platformă care este mai mult decât o simplă listă de lectură. Dezvoltată cu ajutorul cadrului ASP.NET MVC, Razor Pages și Bootstrap, Codex este un loc unde utilizatorii nu numai că își pot înregistra activitatea de lectură, dar își pot personaliza profilurile și se pot alătura provocărilor și lecturilor în echipă.

Pe platformă, utilizatorii își pot crea conturi și apoi își pot personaliza profilurile după bunul plac, prezentându-și cărțile preferate și insignele pe care le-au câștigat în urma lecturilor. Pentru a-și monitoriza lectura, utilizatorii se pot alătura provocării de lectură și își pot stabili obiective anuale, precum și citi peste 30 de pagini în fiecare zi pentru a-și menține streak-ul. În plus, Codex are o mulțime de caracteristici sociale care le permit utilizatorilor să se conecteze, contribuind la sentimentul de comunitate. Aceste caracteristici includ Buddy Reads și Friends Quests, alături de caracteristicile obișnuite ale unei aplicații pentru evidența lecturilor, cum ar fi evaluările și recenziile.

Prin toate funcțiile sale, Codex își propune să încurajeze oamenii să facă din lectură un obicei zilnic și să împărtășească cu ceilalți experiențele lor.

# Table of Contents

# List of Figures

# Chapter 1: Introduction

## 1.1 Type of Work and Subdomain

Codex is a platform that aims to bring readers together by offering the typical book management functionalities, but also coming with fun challenges for readers, interesting statistics, features that allow you to read along with friends and more. The project naturally falls in the broad category of web application development, blending technology with literature.

As far as subdomains, Codex is an intersection of literature, social networking and technology, intending to accompany readers through their reading journeys.

By using the ASP.NET MVC framework, the project takes web application development practices and addresses the specific needs of readers.

## 1.2 Overview

Codex is an application developed by an avid reader for other readers. The project aims to blend the world of literature with web development by creating and designing a platform that is more than just a list of books. Using the ASP.NET MVC framework, Razor pages and Bootstrap, the application has a responsive design, becoming an easy-to-use, interactive companion for users. Codex focuses on enhancing the user's experience by providing a personalized approach to book tracking, social interaction and interesting statistics.

## 1.3 Purpose and Motivation

As I said above, Codex is an application developed by an avid reader for other readers. The idea of the project was born from my love of reading and also from my quest for the perfect app to track my reading progress, wanting a more personalized experience than the apps already available provided.

My love for books is deeply rooted in the social aspect of reading, so, for me, sharing books and reviews with friends is the most rewarding aspect of my reading journey. I wanted Codex to have a buddy reading feature exactly for this reason. Being able to share your reading journey with a friend, while reading the same book at the same time and leaving reviews along the way was at the top of my priority list when envisioning Codex.

As someone motivated by numbers and interested in creating good, lasting habits, I came up with the idea of implementing streaks as a way to help build the habit of picking up a book and reading everyday.

My love for numbers and more specifically for statistics is evident from the implementation of various statistics and yearly recaps.

Additionally, challenges are a big part of Codex since they offer an exciting way to push yourself beyond your limits.

So, in summary, I challenged myself to develop Codex as a platform that encompasses all the things I love, beginning with my greatest passion: books!

## 1.4 Objectives

1. Creating an Account
    - Highlighting favorite books
        - Users can showcase their favorite books on their profile, making it customisable
    - Reading progress
        - Users can see what books they read during the current year and track their progress when it comes to the reading challenge they set for themselves at the beginning of the year
    - Reading Taste and  Badges
        - Users earn badges that reflect their reading taste or accomplishments
        - Users can chose to display these badges on their profile
2. Book Tracking Aspect
    - Reading Challenge
        - Users can set a personal reading challenge and track their progress every year
3. Connection Aspect
    - Friend Quest
        - Users can participate in friends quests with themed challenges that they can complete with the help of friends
    - Streak
        - Users are rewarded for reading a certain amount of pages everyday
    - Buddy Read
        - Users can read a book along with friends
        - Users can leave timestamped comments or annotation as they read, that become visible to their buddies after they pass that point in the book
4. Review Aspect
    - Thoughts on the Book
        - Users can leave reviews visible to all the members offering their thoughts on the book they read
    - Star Ratings
        - Users can leave star ratings for the books they read
        - The average star rating gets updated with every review
5. Yearly Recap Feature
    - Year in Review Summary

    - Every user gets a personalized summary of their reading statistics for the year, such as the total number of books read, average star rating, most-read genres, and any completed reading challenges..

## 1.5 Preliminaries

### 1.5.1 Technologies and Frameworks used

- ASP.NET MVC Framework
    - For the back-end I decided to go for the ASP.NET MVC framework because of the flexible, but still well-structured architectural pattern. The Model-View-Controller (MVC) architectural pattern offers a clear separation between the parts of the application, making it so much easier to plan and structure the implementation of the app. This way, the development process was a lot less overwhelming and the code was easier to organize.
- Razor Pages
    - As far as front-end, I chose Razor Pages since they are part of the ASP.NET Core framework. Razor Pages provide a page-based programming model that offers an easier and more productive way of building a user interface, so it was the ideal choice for my project.
- Bootstrap
    - For styling the Razor Pages, I integrated Bootstrap. Bootstrap is an easy to use open-source toolkit for developing with HTML, CSS and JS and it is really helpful in developing a responsive UI  by providing such a large variety of components that are really easy to incorporate in a project.

### 1.5.2 Key Concepts

- Web Application Development
    - Web Application Development has the user at its core and it is a really important aspect of today's technology. Being focused on user-centric design, it blends together some of the most interesting and important aspects such as front-end and back-end development, user interface design and database management.
- Intuitive Design
    - Since a web application has a user at its core, responsive design is incredibly important, so making sure a user can easily navigate the app regardless of the screen size of the device was a priority.
- Literature and Socialization
    - Codex aims to combine literature and socialization by encouraging users to read alongside friends and engage with the community. In order to achieve

this, Codex has a diverse set of features including reading quests and buddy reads, which will be expanded on in later chapters.

## 1.6 Needs Satisfied by Codex

I mentioned earlier that Codex was born from the search for the perfect reading application. Most reading applications only focus on the tracking aspect and I felt there was so much more a web application for readers could offer. The following is what Codex aims to offer its users:

1. An easy way to keep track of their reads

   Codex offers the classic reading tracker experience by allowing users to organize their reads into shelves, add reviews and ratings, keep track of progress on the books they are currently reading, set a yearly reading goal and see their statistics at the end of the year.
2. A personalized profile

   Codex is not just a reading tracker, it's also a way in which users can share their reading taste with others through their personalized profile. Users can choose to display their favorite books on their profile, as well display badges awarded to them based on their reading taste.
3. An incentive to read daily

   The steak system is a great way to encourage users to turn reading into a habit. Through streaks users can track how many consecutive days they've read more than 30 pages.
4. A way to interact with friends

   Interaction is at the core of Codex. Users can check out others' profiles and follow each other. The Buddy Read feature allows friends to read books together and share thoughts that only become visible to others when they've passed that point in the book. Additionally, users can join Friends Quests where they get a reading goal and aim to read that amount together in the course of a week, fostering teamwork and a sense of shared accomplishment.

## 1.7 Similar Applications

### 1.7.1 Goodreads

Probably the most popular reading tracker, Goodreads is a website that allows users to search for books, create shelves and share their thoughts on their most recent reads. Its most notable features include book recommendations, community interaction, the reading challenge, a yearly recap and statistics.

- Book Recommendations

    - On Goodreads, users can input their favorite books and genres and their algorithm takes that data combined with their activity on the website like reviews, what books they are currently reading and recommends other books users may enjoy.
- Community Interaction
    - Members are encouraged to share their thoughts on books through review and ratings. Users can follow each other and see what people they follow have been reading recently on their timeline. They can also follow authors , celebrities or other public figures to see what they're reading or what their favorite books are. Another notable feature is the book club, where users can join either existing book clubs or create their own. Here they can read a chosen book together and discuss it.
- Reading Challenge
    - At the beginning of a new year, users can join the reading challenge and set a reading goal for the year. Once joined, Goodreads keeps track of the books users read throughout the year. Users can manually mark the books as "read" or use the "currently reading" status to update their progress. As well as tracking, Goodreads also motivates their users to keep up with their goal by offering insights into how they relate to it, telling them if they are ahead, behind or on schedule.
- Yearly Recap
    - The end of year recap goes hand in hand with the reading challenge. At the end of every year, goodreads generates a personalized recap highlighting the books the user read, their average rating, average page count, longest and shortest book read, most popular and least popular book read and other insights.
- Statistics
    - Users don't have to wait for the yearly recap, during the year they can still get some interesting statistics on their reading habits. The Stats page provides detailed information, such as the number of books and pages read annually, the publication years of the books read and a monthly breakdown of the number of books and pages read. Using graphs to visually represent the statistics, users can see their progress over time.

### 1.7. 2 The StoryGraph

Maybe less popular than Goodreads, The StoryGraph is another social cataloging website for readers. The platform helps users to keep track of their reads, but also has other notable features, like mood-based recommendations, reading with friends, reading challenge, half and quarter stars, up-next in queue, yearly statistics among others.

- Mood-based Recommendations
    - The StoryGraph doesn't stop at genres when organizing books, it also takes into account the mood of the book, so when users are looking for a recommendation they can also specify the mood of the next book they are looking to read.
- Reading with Friends
    - The Reading with Friends is one of the features the platform is most well-known for in the reading community. Users can read books along with friends and leave live reactions to specific parts of the book. To keep things spoiler-free users can’t see others comments until they also pass that part of the book.
- Reading Challenge
    - Just like Goodreads, The StoryGraph also has a reading challenge feature. Users can join the challenge and track the progress on their reading goal for the year.
- Half and Quarter Stars
    - The StoryGraph differentiates itself from Goodreads by allowing users to rate books using half and quarter stars. This way users can express their thoughts on a book through more accurate ratings.
- Up-next in Queue
    - The Up-next in Queue feature allows users to plan their next 5 reads. So when a user finishes a book, they know what to read next.
- Yearly Statistics
    - Similarly to Goodreads, users get a personalized end of year wrap-up. Here users can see how many books and pages they read throughout the year in total and split by month, what moods they gravitate to and a breakdown of all the books read by every mood, the pace of the books they read, a breakdown of the books by number of pages and by whether they are fiction or nonfiction, a breakdown by genre, their most read authors of the year, the average rating and also a graph split by each rating.

## 1.8 What Codex Brings New

Although Goodreads and The StoryGraph were big inspirations for my project, Codex introduces some new features that personalize the reading experience even more.

To begin with, Codex allows users to personalize their profile by showcasing their favorite books. Users can choose which books out of their five stars reads to display.

Codex also allows users to express their personality by displaying badges on their profile. Badges are earned based on the user's reading preferences, so when a user reads a certain amount of books in a certain genre or by a certain author they earn a badge.

Codex aims to encourage users to make reading a habit, so with this in mind I developed the streaks feature. Through streaks users can track how many consecutive days they read more than 30 pages, motivating them to make time to read every day.

Furthermore, Codex introduces the Friends Quest feature. Friends Quests are a fun way for users to work together towards a common goal. When joining a Friends Quest, users get assigned a random page goal and they read as much as possible in order to reach that goal.

## 1.9 Structure of the Thesis

The thesis presents an in-depth analysis of the development and functionality of the Codex web application. In the following chapters and subchapters, I'm addressing key points to provide a comprehensive understanding of the application and its purpose.

### Chapter 1: Introduction

The first chapter is an introduction to the project, setting the stage for the thesis by outlining the type of work and subdomain. Along with that, it provides an overview, detailing the purpose and motivation behind the development of the web application, followed by the specific objectives that I aimed to achieve through this project. There is a preliminaries section that discusses the technologies and frameworks used as well as a section about similar applications to Codex and what Codex does differently from those applications.

### Chapter 2: Technologies Used

This chapter details the technologies I used while developing the application. Each technology has its dedicated subchapter discussing the details, providing a deeper understanding of their roles and benefits in the project. The main technologies covered are ASP.NET Core MVC, ASP.NET Identity, Entity Framework Core and Razor Pages.

### Chapter 3: System Architecture

This chapter delves deeper into both front and back-end development, explaining how each of the technologies was used in order to create the final version of the web application. It aims to provide more insight into the development process of Codex, explaining how all the theoretical aspects from the previous chapter were implemented into the project. Although it may seem similar to chapter two, in this chapter the focus is shifted from theory to practice, explaining how controllers and models, as well as model mapping, migrations, database design, Identity, Razor Pages and Bootstrap all came together to shape Codex.

### Chapter 4: Application Features

As the title of the chapter suggests, this chapter highlights all the features of Codex from the perspective of the user. The main focus here was to explain in detail every feature implemented during development. All the features were grouped into categories and some were grouped further into subcategories for organizational purposes. The chapter starts with User Experience explaining all the features implemented in order to help the user sign-up and log-in and also the profile customization aspects. Following this there is a chapter about books, shelves and reviews which gives some insight into how books are managed within Codex, leading seamlessly to the reading challenge and yearly recap features. In the end, there is a section about the connection aspect with features such as the following and follower system, streaks, friends quests and buddy reads, that are meant to motivate users to engage with other users and with the platform.

### Chapter 5: Conclusions and Future Perspectives

The final chapter is dedicated to the conclusions and future development perspectives of Codex. It presents the development experience, the impact on the users and the goal of the thesis as well as providing some outlines of future development ideas that could continue the evolution of Codex.

# Chapter 2: Technologies Used

## 2.1 ASP.NET Core MVC

ASP.NET is a server-side framework for building web applications, websites and web services introduced by Microsoft in 2002. It supports a number of different programming models, of which I decided on MVC for the implementation of Codex.

The Model-View-Controller (MVC) pattern separates the application into three main components: the Model, the View and the Controller.

- Model
    - This component represents the application data and any business logic or operations performed by it.
- View
    - This component is responsible for what the user sees, so the user interface. It typically contains the HTML and CSS code and uses the Razor view engine to embed .NET code in HTML markup. There is usually little logic in this component and if there is logic, it is related to presenting content to the user.
- Controller
    - Just like the name implies, this component is responsible for interacting with the other components. Controllers handle user interaction, so when they receive a request it is responsible for selecting which models to interact with and which view to render.

## 2.2 ASP.NET Identity

ASP.NET Identity is a framework that facilitates adding authentication and authorization to ASP.NET Core applications. It is a very useful tool for developers, helping manage user authentication and allowing users to create an account with the login information stored in Identity. On top of user authentication, Identity supports the allocation of user roles, which facilitates access control within the application.

## 2.3 Entity Framework Core

In ASP.NET MVC, data storage is done by using an open-source technology called Entity Framework Core.

The framework is an Object Relational Mapper (ORM) for .NET, allowing developers to interact with data from the database. Through using a collection of libraries, Entity Framework Core maps every class from a model to the database.

Entity Framework Core incorporates  Language-Integrated Query (LINQ), which allows developers to integrate any Relational Database Management System (RDMS)

such as SQL Server, Oracle Server and so on. The RDMS stores data in tables which can be accessed and processed using SQL. LINQ facilitates the integration of SQL queries within C# code.

## 2.4 Razor Pages

ASP.NET Core's default and recommended View engine is the Razor View engine. This is a page-based model for building server-rendered web UI in ASP.NET Core.

The main advantage of Razor Pages is that it allows developers to mix C# code and HTML code in a single file. The razor code blocks start with the @ symbol and the Razor View engine is intelligent enough to distinguish between the two types of code and generate the right output.

# Chapter 3: System Architecture

## 3.1 Back-End Development

When talking about Back-End, developers refer to the server-side development of the application, which is the part that stores, processes and delivers web pages to the user's request. So, the back-end handles tasks such as the interaction with the database, the business logic and APIs.

### 3.1.1 ASP.NET Core MVC Framework

As discussed before, Codex uses the ASP.NET Core MVC framework to create a structured and maintainable server-side architecture. Relevant for back-end development are Controllers and Models.

#### 3.1.1.1 Controllers

The controllers in Codex manage the incoming HTTP requests, perform the needed operations and return the appropriate response. For example, the *GenresController* handles all the actions such as creating a new genre, editing a genre, displaying all the genres in a database, displaying just one of the genres in the database and deleting a genre from the database. Each of the controller actions listed before interact with the services layer of the application in order to process the data and return the necessary views so the information can be displayed to the user.

#### 3.1.1.2 Models

Models represent the data and business logic of Codex. They define the structure of the data that will be stored and manipulated in the database. Each model is mapped to a table, with the help of Entity Framework Core, and the properties of a model represent the columns of that table.

For instance, the Genre model represents what a genre looks like in the system's database and, in this case, a genre has the following attributes:

- 'GenreId': this refers to the unique identifier of a genre in the database, being the primary key of the Genres table.
- 'Name': this refers to the name of a genre and it is a mandatory field when creating a new genre

### 3.1.2 Entity Framework Core

I mentioned earlier that Entity Framework Core facilitates the process of mapping each model to its respective table in the database.

#### 3.1.2.1 Model Mapping and Migrations

By using Entity Framework Core, the models are automatically mapped to the corresponding database table. Entity Framework is installed using NuGet and supports the code-first technique which allows developers to write the classes through which the database is automatically generated.

In the example of the Genres model, Entity Framework Core automatically maps the two properties to individual columns, part of the genres table.

Another aspect of Entity Framework Core is that it supports migrations. Migrations offer a way to update the database as the project evolves and grows. As the development process advances and new entities in the database are needed, the command *Add-Migration Migration-Name*, followed by *Update-Database* is used and a corresponding migration is created containing the necessary updates to keep the database schema in sync.

In order to determine what updates are necessary, Entity Framework Core compares the current model to the snapshot of the old model, determines what the differences are and generates the new migration file. Once the migration is added, Entity Framework Core keeps track of it by adding it to a special migration history table.

#### 3.1.2.2 ApplicationDbContext

The *ApplicationDbContext* class manages the connection to the database and tracks the changes to the models. It inherits from DbContext which is part of Entity Framework Core.

The ApplicationDbContext class includes all the configurations and methods necessary to manipulate and save data to the database. The class contains the information needed to manage all the entities in the database, as well as define the many-to-many relationships and foreign keys.

#### 3.1.2.3 Connecting to the  Database

The connection to the database can be established in two ways in ASP.NET Core: with or without dependency injection. For Codex, I preferred the approach of using dependency injection.

Dependency Injection simplifies the management of the database connections by automatically providing instances of the needed services at runtime. In order to establish the connection I first defined the *ApplicationDbContext* class I talked about earlier. After that I configured the connection by adding the connection string to the appsettings.json file. The file looks like this now:

Following this step, I registered the DbContext in the *Program.cs* file.

In the end, for every controller I injected the *ApplicationDbContext* through the constructor.

### 3.1.3 Database Design

For Codex, I created a local database. I created it using SQL Server LocalDB which is a lightweight version of SQL Server Express.

The Codex database was designed to manage and relate the different entities involved in the application efficiently. I represented the design through two diagrams: the Entity-Relationship diagram and the conceptual diagram.

#### 3.1.3.1 Entity-Relationship Diagram

As the name implies, the Entity-Relationship Diagram illustrates the relationship between entities in the database. More specifically, they are used to show how the entities relate to each other within the system.

Entity-Relationship diagrams use a defined set of symbols, such as rectangles for the entities, which are connected by lines that represent the relationships. These diagrams are inspired by the grammatical structure, so entities are nouns and relationships are verbs. Each relationship is defined by its cardinality, which indicates the number of instances of one entity that can be associated with an instance of another entity, including the minimum and maximum. These cardinalities include one-to-one, one-to-many and many-to-many.

Below, figure 3.1 shows the Entity-Relationship Diagram of the Codex database. It visually represents how different elements such as the books, shelves, users and so on are connected to each other. The entities are represented by rectangles and the relationships between them are illustrated through undirected arcs. The primary key of each entity is marked with a PK on its left. Additionally, the minimum cardinality for each relationship is indicated in parentheses, while the maximum cardinality is written next to it, without parentheses.

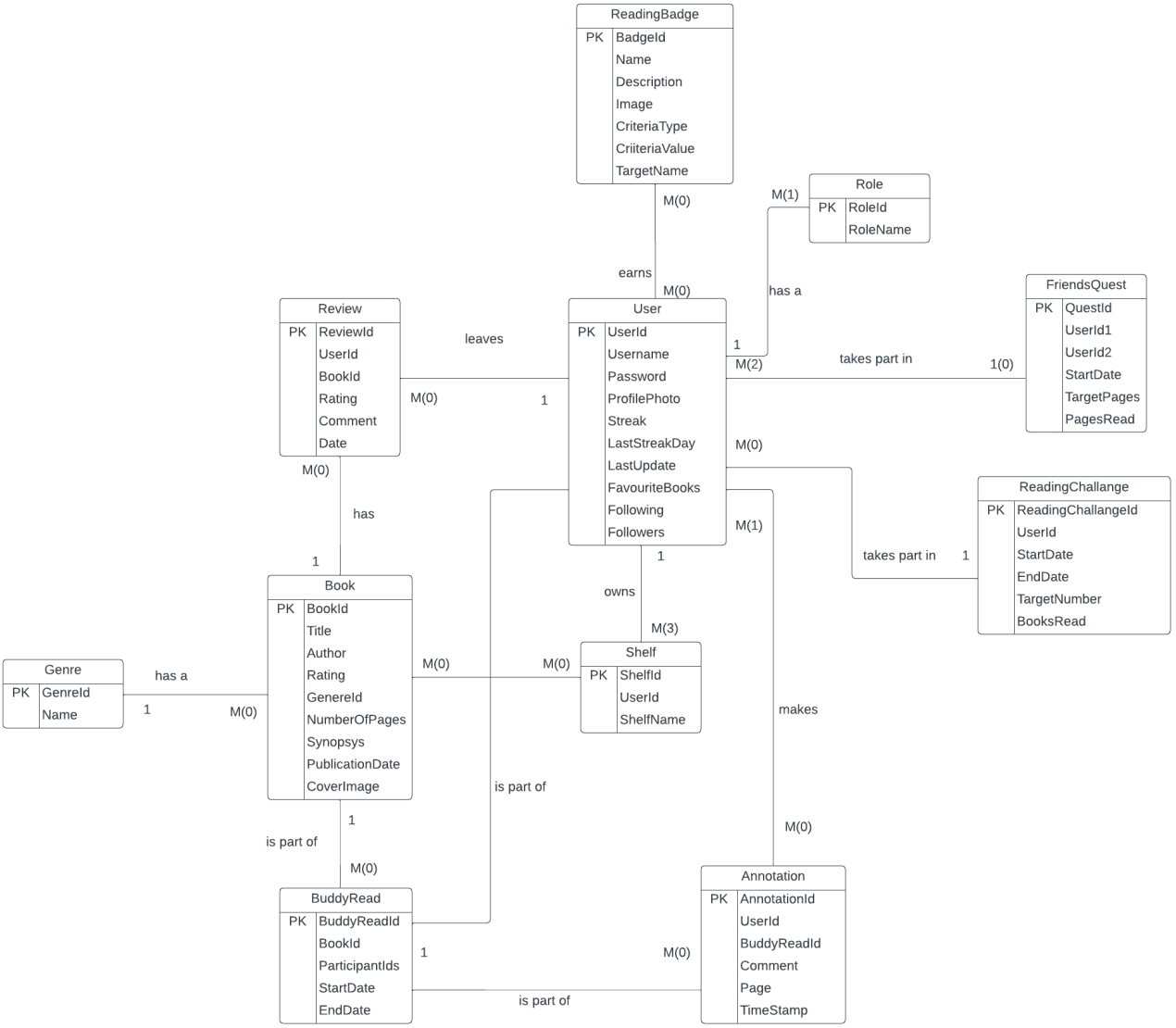


*Figure 3.1 - The Entity-Relationship diagram of the database*

### 3.1.3.2 Conceptual Diagram

The Conceptual Diagram provides a more high-level view of the database structure, focusing on the overall organization and the key entities and relationships within the system. Unlike the Entity-Relationship Diagram, the Conceptual Diagram doesn't focus on the specifics of the relationships and their cardinalities. In order to handle many-to-many relationships, the Conceptual Diagram introduces intermediary tables, called *junction tables,* which split the many-to-many relationship into two one-to-many relationships.

Below, Figure 3.2 represents the Conceptual Diagram of the Codex database. Similar to the Entity-Relationship Diagram, this diagram also highlights the main entities,

such as books, users, shelves, reviews and so on, but also depicts four new tables. These tables appeared as a result of splitting the many-to-many relationships into the two one-to-many relationships. More specifically, between the ReadingBadge table and the User table there was a many-to-many relationship since a user could earn more than one badge and the same badge could be earned by more than one user. This led to the creation of the BadgeEarned entity, a junction table, which serves as an intermediary table used to resolve the many-to-many relationship. The same process applied for the many-to-many relationships between Users and Roles, Books and Shelves, and Users and BuddyReads, resulting in the creation of three additional tables: UserInRole, BookOnShelf and BuddyReadParticipant.

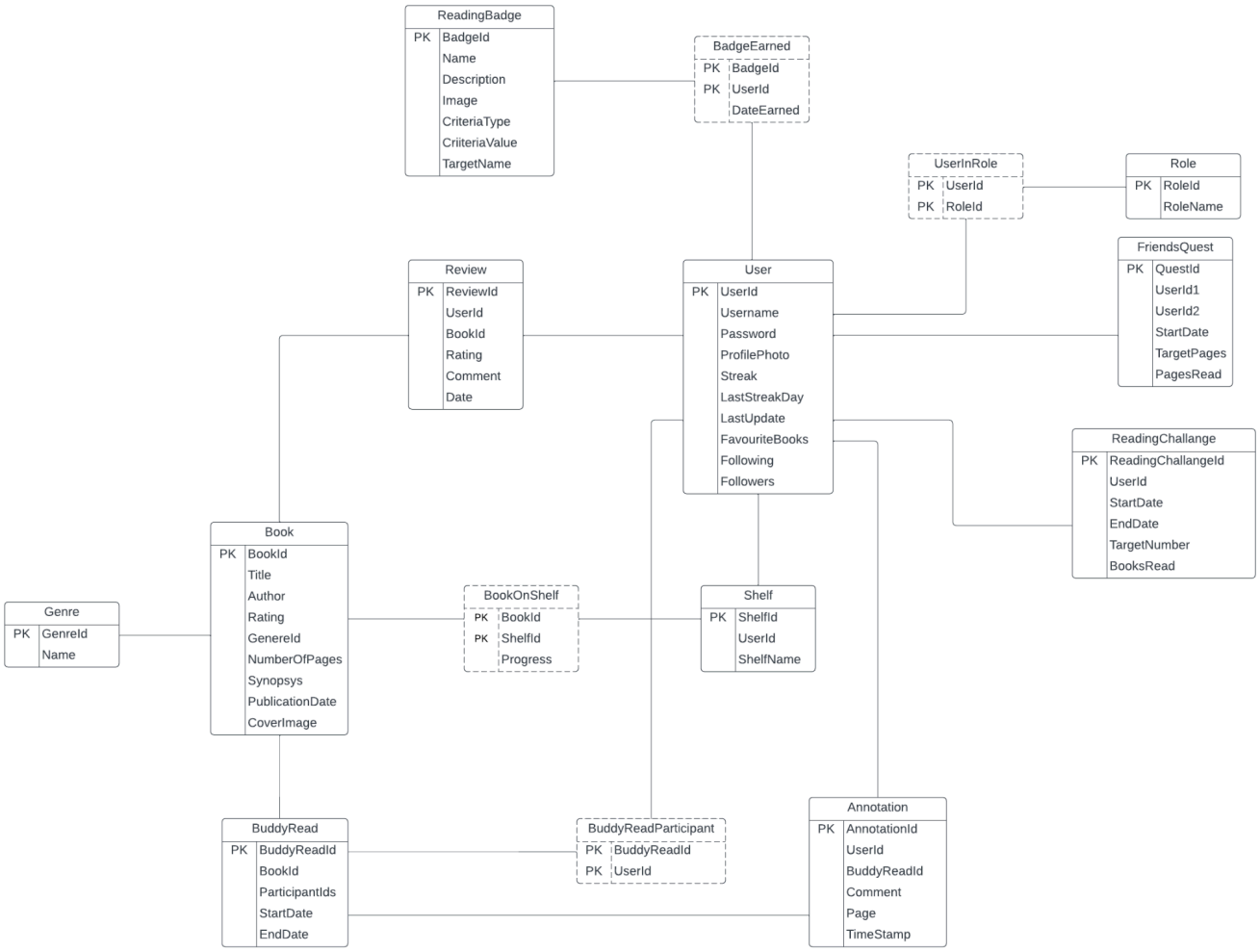


*Figure 3.2 - The Conceptual diagram of the database*

### 3.1.4 ASP.NET Identity

As I previously said, ASP.NET Identity is the framework that facilitates the user authentication and authorization. In order to have features such as role management, user

authentication and secure access control, I opted to use Identity in the development of Codex.

The first step in order to achieve this was to make sure that when creating the Codex project the *Individual Accounts* authentication option was selected. This automatically sets up the project to include the essential Identity components, which means that it creates the needed database tables, such as User.

Following this step, I set up the *ApplicationUser* class. This class inherits from IdentityUser and contains some more information necessary for Codex users.

Afterwards I configured Identity in the *Program.cs* file in order to set up the user interface, cookies and the necessary mechanisms for authentication, as well as role management which I enabled by using *AddRoles*.

Next I added some seed data to populate the database with some initial roles and users. For this I created a class with a suggestive name, *SeedData.cs*.

Finally, the last step was making some changes to the registration form (R*egister.cshtml*) in order to allow the user to choose a name they want to use on the site and a photo.

## 3.2 Front-End Development

While back-end development refers to the server-side portion of a web application, front-end development refers to the creation of the client-side portion of the application. This involves designing and implementing the user interface, which is the part of the application that the users interact with. In order to develop an intuitive, responsive and visually appealing experience for the users, in the development process of Codex I opted to use Razor Pages and Bootstrap.

### 3.2.1 Razor Pages

Razor Pages are a page-based programming model in ASP.NET Core that are dynamically rendered on the server to generate the page's HTML and CSS response. When it comes to Codex user interface development, razor pages manage various aspects such as displaying book details, user profile pages and so on.

Taking the Index page of the Genres as an example, the *@model* directive at the top of the file specifies the type of data the page will receive from the controller. For this page, the page expects to receive a collection of Genre objects, because this page displays all the genres in the database.

Moving on, the integration of the HTML tag <h2> displays the heading for the page which lets the user know that they are viewing the genres in the database.

After this, the @if block checks if there's a message stored in *TempData* and if so displays it inside a Bootstrap-style alert box.

Next, I used a for each loop to iterate through the collection of genres passed to the page from the back-end. This way each genre is displayed in its own card that includes the name and two buttons. The buttons send to the *Edit* and *Delete* methods from the controller. In order to achieve this the *href* attribute of the Edit and the *action* attribute of the *Delete* form use Razor syntax (*@genre.GenreId*) to dynamically generate the appropriate URL based on the specific genre ID.

In the end, there is a button for adding a new genre, that sends to the *New* method of the *Genre Controller*. Below, Figure 3.3 shows the end result of what the Index page looks like, displaying all the genres in the Codex database in individual cards.

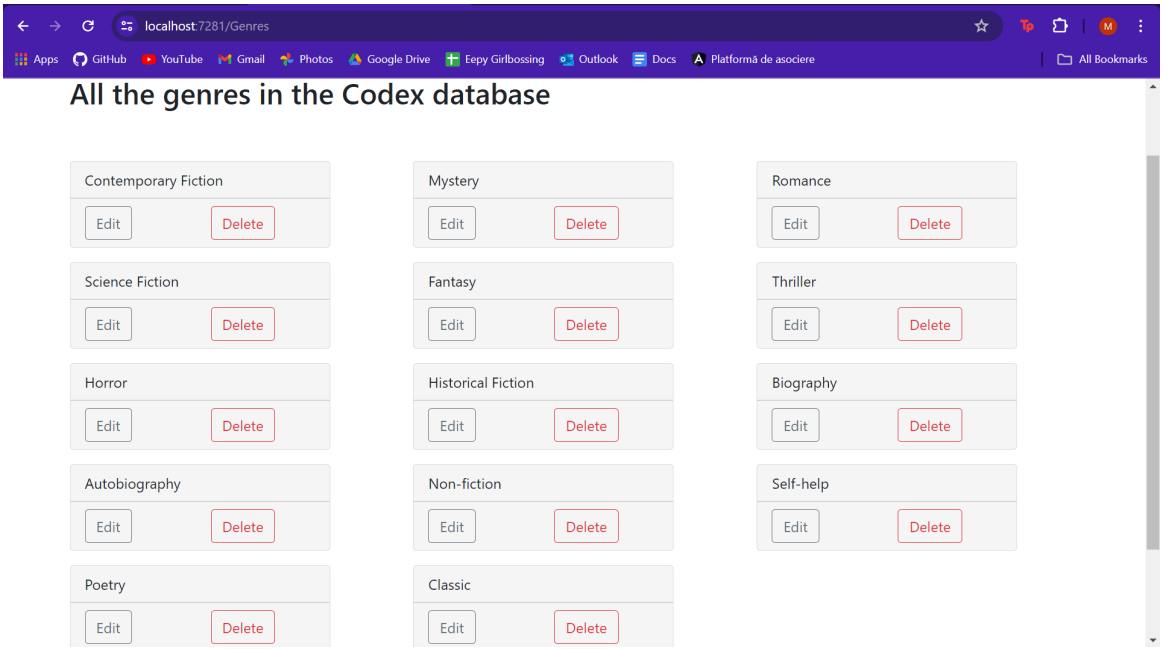


*Figure 3.3 - The Index page for Genres*

## 3.2.2 Bootstrap

Bootstrap is a front-end framework that simplifies development by providing pre-designed components and responsive grid layouts. In the process of designing Codex's user interface, I used Bootstrap for a variety of different purposes from the grid layouts and cards to buttons, icons and alerts.

### 3.2.2.1 Bootstrap Grid Layouts

Bootstrap's grid system allows for the creation of responsive layouts by using containers, rows and columns. For Codex, I used the grid system to structure the pages dynamically. In the above example of the Index page illustrated in Figure 3.3, I implemented the page using the grid layout, more specifically I used this system to organize the cards that contain the names of the genres in a way that adjusts based on the user's screen size.

The *row* class creates a flex container that allows the alignment of the genre cards. Then, the *col-md-4 mb-3* class specifies that each of the cards should take up 4 columns on a medium to large screen and allow for a bottom margin of 3 units for spacing. By using this format, I made sure that the cards are displayed in a three-column layout making it easy for the user to see all the cards in an organized way that is not overwhelming. The three-column layout is applied on the medium to large screens, but if the screen size is smaller it is adjusted appropriately.

#### 3.2.2.2 Bootstrap Cards

Cards are really versatile components offered by Bootstrap and are really useful when it comes to displaying information in a clean and organized manner. Although they seem simple at first glance, they offer a lot of customization possibilities.

A card has three main parts: the header, the body and a footer. Alongside these parts it is possible to add other details like titles, links, images and list groups.

When developing Codex, cards were a big part of the user interface because of the customization possibilities they offer and the way they work seamlessly with a grid layout to create a simple and clean page.

In the Index page of the genres illustrated in Figure 3.3, for example, I used cards to display the genres in the database along with buttons for each of the genres.Here the card's header contains the name of the genre and the footer contains the two buttons for editing and deleting a genre. In this example the body is not explicitly shown.

#### 3.2.2.3 Bootstrap Buttons

Bootstrap buttons are essential for the user's experience. They come with a variety of built-in styles for different purposes making it easy to create visually appealing buttons for an application.

In the development of Codex I exclusively used bootstrap buttons for the various actions. For instance, in the genres Index page, shown in Figure 3.3, I used the outline buttons for Edit and Delete.

#### 3.2.2.4 Bootstrap Icons

Bootstrap Icons provide a set of vectors that can be integrated in any application. In order to add Bootstrap Icons to an application, first they need to be linked in the CSS file head of the _Layout.cshtml document, by adding the following link *<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.2/font/bootstrap-icons.css" />*.

In Codex's user interface, an instance when I used bootstrap icons is in order to show a little fire icon next to the number of days a user had consecutively read more than

30 pages. Figure 3.4 illustrates how the fire icon is integrated in the user's profile view to show the streak count.

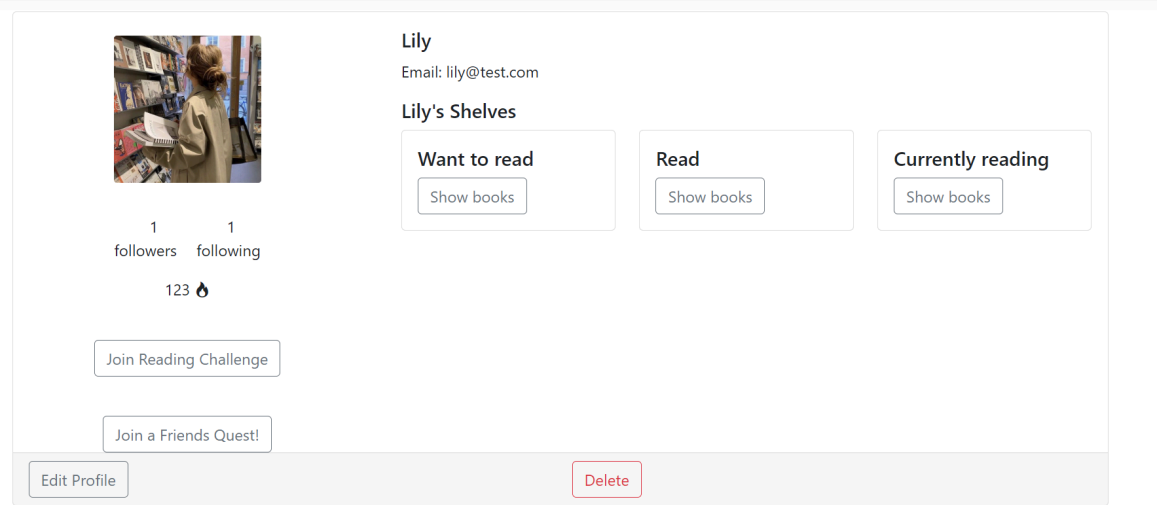


*Figure 3.4 - The profile of a user with a 123 day streak*

### 3.2.2.5 Bootstrap Alerts

Bootstrap Alerts are an important part of user interface development since they offer a convenient way to display important messages to the user. Just like with buttons, Bootstrap offers a variety of styles when it comes to alerts too.

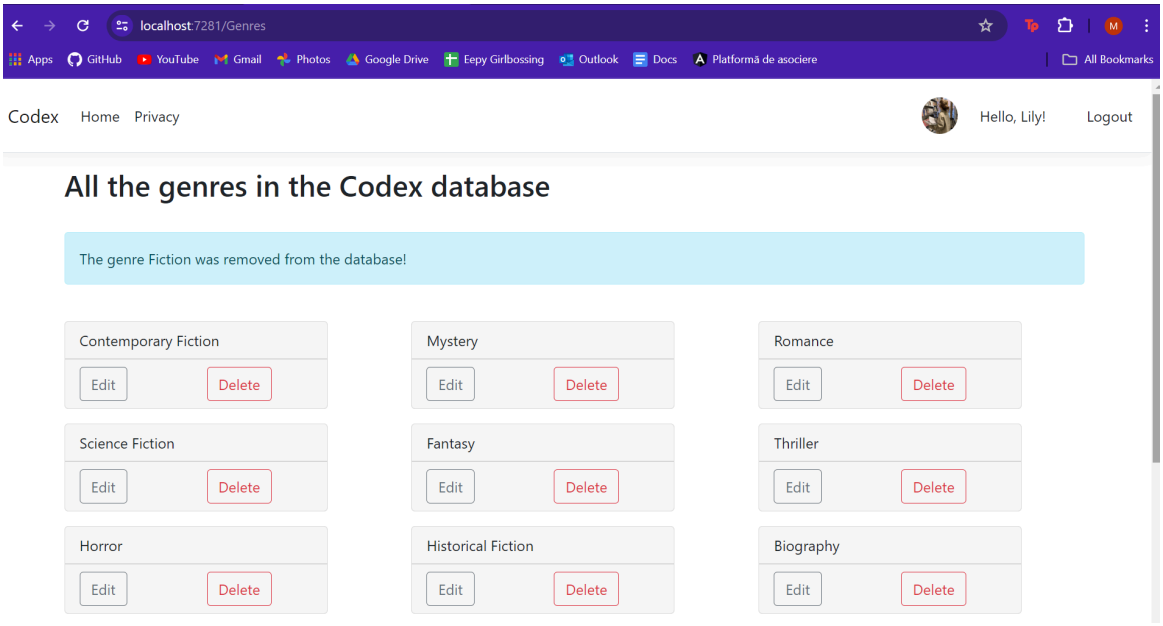


*Figure 3.5 - The Genres Index view, illustrating what alerts look like to the user*

In Codex, I used alerts as a way to notify the user of important events or information. For instance, Figure 3.5 showcases the Genres Index page where I used the alerts to display the message in the *TempData,* a temporary storage mechanism.

# Chapter 4: Application Features

## 4.1 User Experience

Creating a unique and engaging user experience was my primary focus when planning and developing Codex. In a market full of various book-related applications, I wanted to make sure that Codex sood out through its features that are both intuitive and appealing to users. In order to achieve this, I incorporated elements like the sign-in and sign-up feature that allows users to make their own accounts and the extensive profile customization options such as the ability to highlight their favorite books and the badge system.

### 4.1.1 Types of Users

Codex had four main types of users: Members, called simply users, Admins, Editors and Not Signed In Users.

#### 4.1.1.1 Users

When I refer to users, I refer to the regular members of Codex that have created an account and can explore and enjoy the content on the platform. These users can use all features such as profile customization, browsing books, organizing them on shelves, adding reviews, participating in challenges etc.

#### 4.1.1.2 Admins

Admins have the highest level of control and access within Codex and are the people who manage the platform and ensure it runs smoothly. Admins can add, remove or edit user accounts, books, genres etc.

#### 4.1.1.3 Editors

Editors help admins manage the platform. Even though they don't have as many privileges as admins, they can edit and delete books, genres etc.

#### 4.1.1.4 Not Signed In Members

These are the visitors who explore Codex without creating an account or signing in. These people can only browse books, but not leave reviews. They can’t join challenges and they don't have access to features like badges and streaks.

### 4.1.2 Sign-in and Sign-up Features

In order to have access to an account users must first create an account and then sign-in into said account. These features were designed with simplicity and security in mind and the Identity framework played a major role in the implementation.

#### 4.1.2.1 Creating an Account

After clicking on the register button, the new user is greeted with the sign-up form that requires minimal user information in order for them to be able to create an account. Users need to fill in their name, e-mail address, choose a password and, optionally, they can choose a profile picture by providing a URL. If they chose not to add the picture at the beginning they will have the default profile picture until they choose to change it.

Figure 4.1 illustrates what the register page  for Codex looks like.

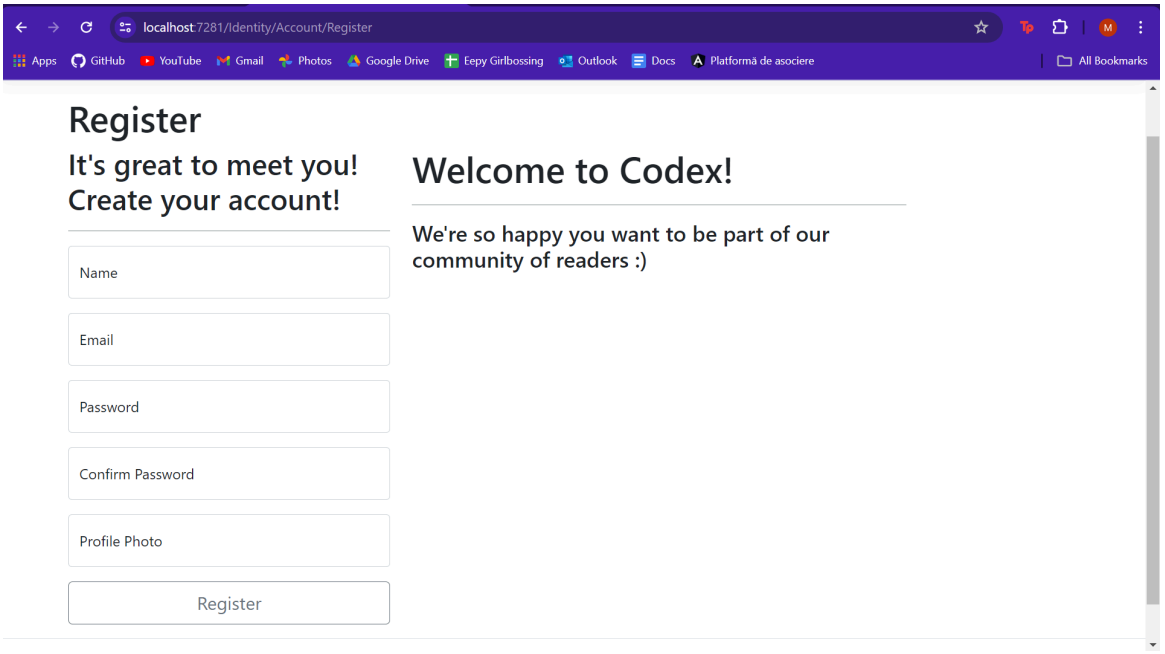


*Figure 4.1 - The Register page*

#### 4.1.2.2 Sign-in

Once a user has created their account, they can simply log in by clicking on the login button. On the login page the user is greeted by a form where they have to fill in their email and password, as shown in Figure 4.2, and based on these they get access to their account.

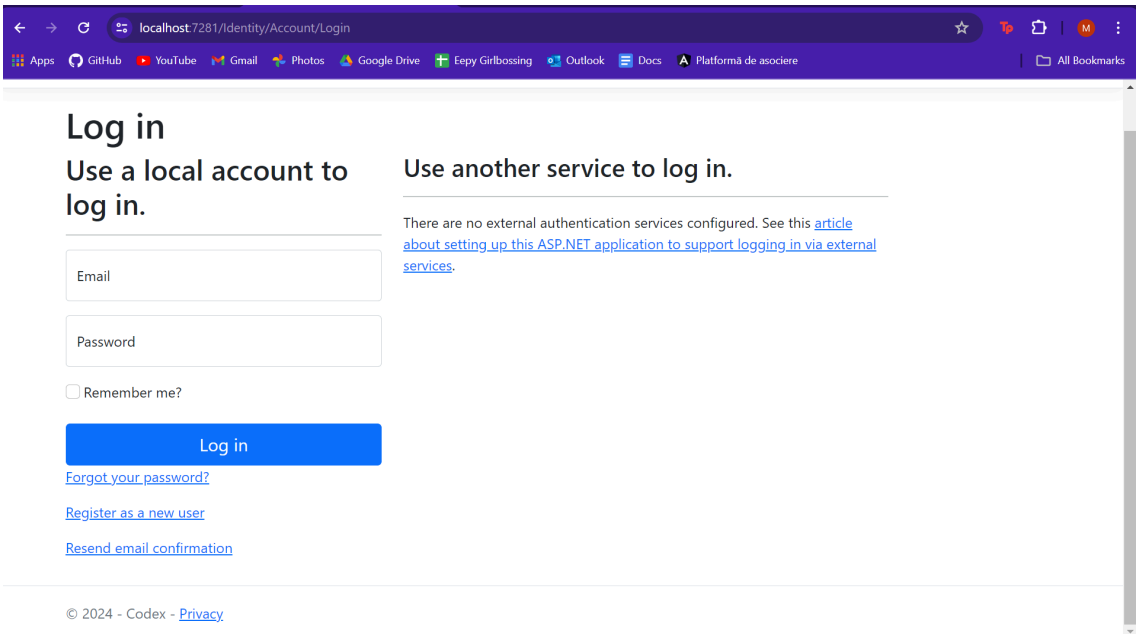


*Figure 4.2 - The Login page*

### 4.1.3 Profile Customization

As I have stated before, one of the key features of Codex is profile customization. When developing Codex, I wanted to offer users a little more creative freedom over their profiles and allow their profiles to showcase the readers' reading preferences.

#### 4.1.3.1 Favorite Books

One standout feature of Codex is the ability to highlight favorite books on the user's profile. Under the user's shelves the user can choose what books they would like to display in a carousel under their favorite books heading. Figure 4.3 shows what the carousel looks like once a user has chosen their favorite books.

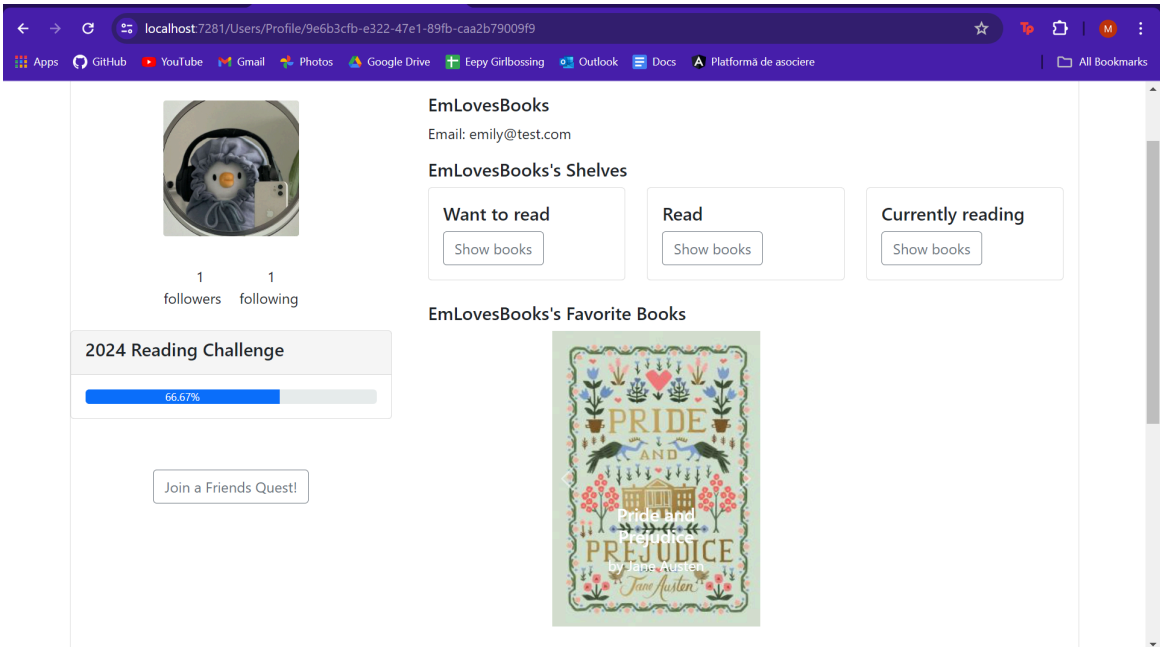


*Figure 4.3 - The favorite books carousel in the user's profile*

In order to implement this feature in the user's model I added a collection of favorite books, along with a list of the options for their favorite books.

These options are populated in the controller and are made up of all the books the user rated 5 stars.

This list is then sent to the view and, while editing their profile, the user can choose which books they find most representative of their reading taste and display them on their profile, as shown in Figure 4.4. Once they save the changes, the books will be displayed on their profile.

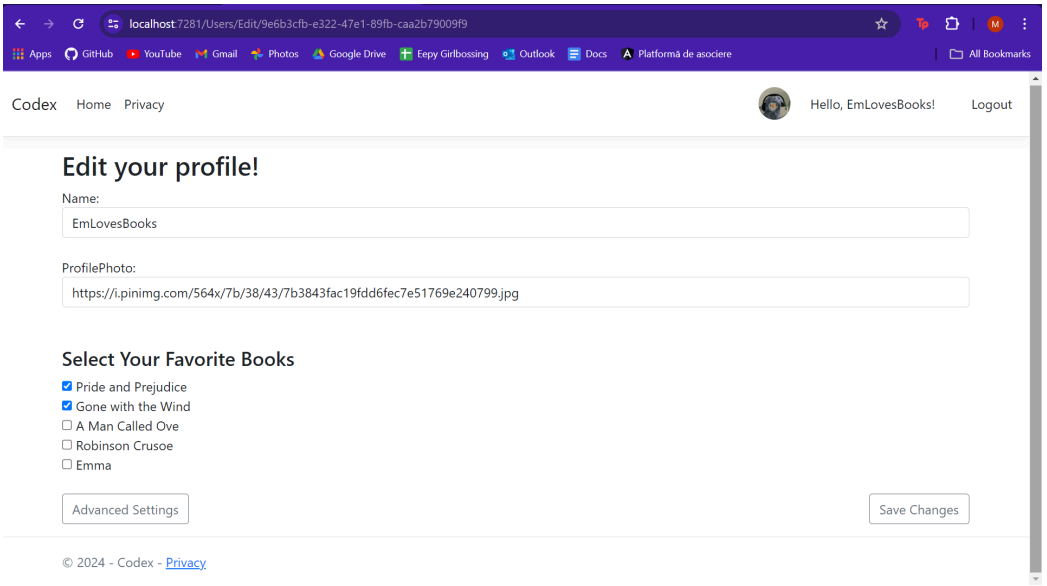


*Figure 4.4 - The Edit profile page*

### 4.1.3.2 Reading Badges

Reading badges are a great incentive for users to read more, but also a great way for them to showcase their reading taste. Once a user earns a badge it gets automatically displayed on their profile underneath their favorite books, as shown in Figure 4.5.

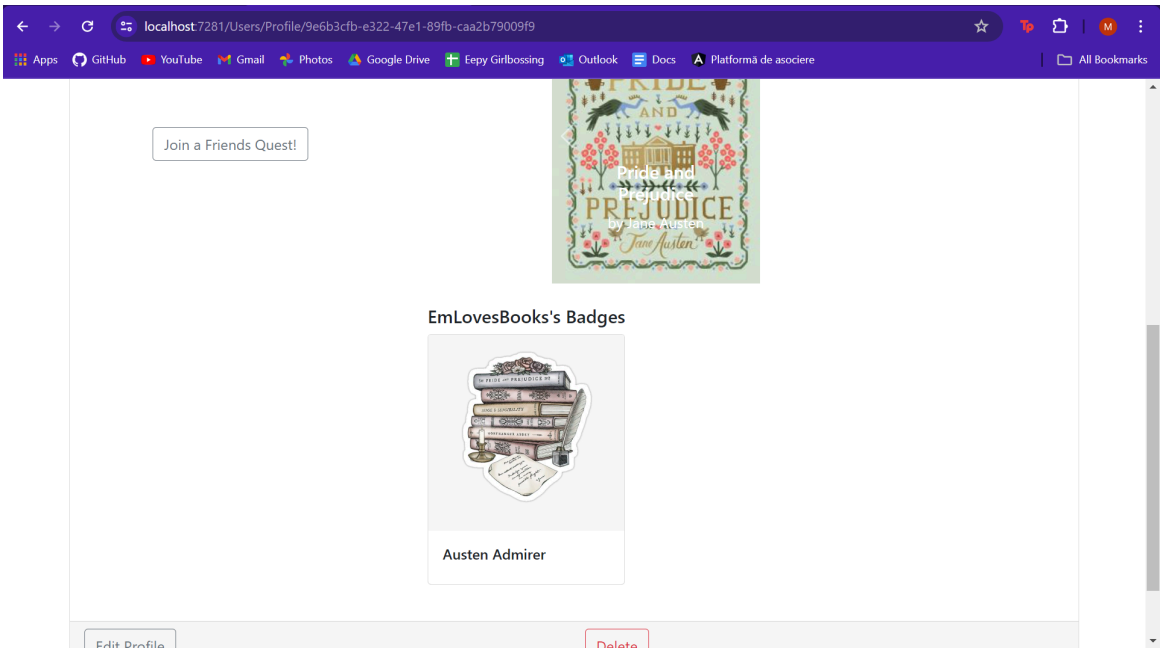


*Figure 4.5 - The badges display on the user's profile*

Admins are responsible for reading badge management. They have access to all the badges in the database and they can edit or delete them, as well as add new badges.

Figure 4.6 depicts the Index page for the badges, which only admins have access to.

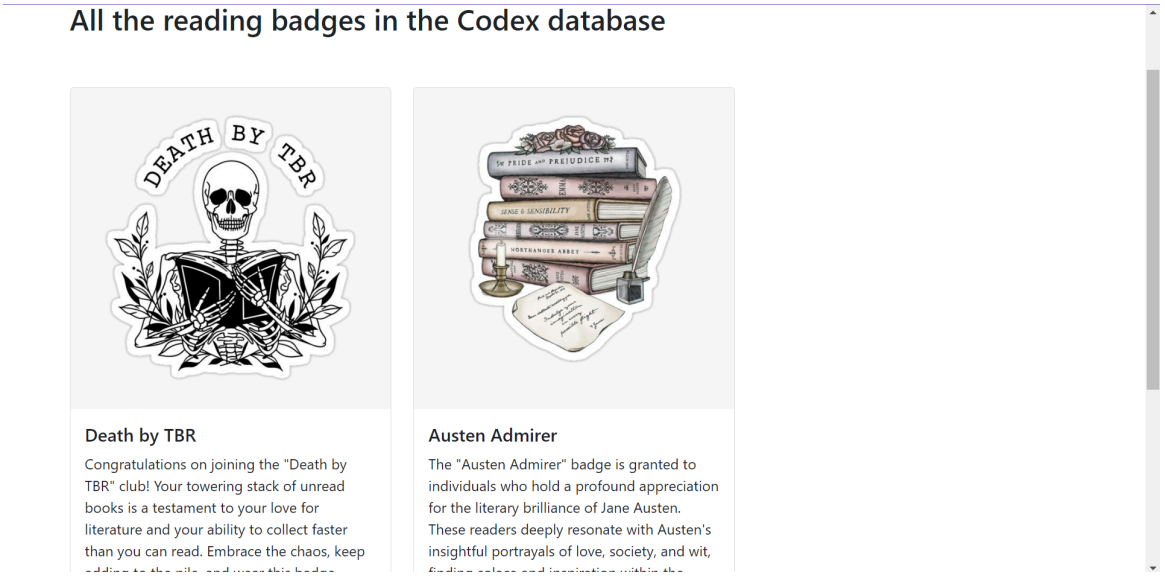


*Figure 4.6 - The Index page for the badges*

In order to add a new badge to the database, the admin has to fill out a form providing information about the badge. As shown in Figure 4.7, first they need to enter the name of the new badge, a description, the url of the image for the badge, then they need to select the criteria type, the target number of books and some extra information in some cases.


*Figure 4.7 - The New page for the badges*

Types of Badges

To make it easier to manage, but also to award the badges, I categorized the badges into four main types, represented by enums in the code. This approach allowed me to add, edit and delete badges without needing to hardcode the criteria checks, making the system flexible and scalable. Each one of the badge types has a certain criteria the user needs to meet in order to earn the badge.

1. Books Read

   This badge type is awarded based on the number of books a user has read. There is no other criteria than for them to have read that number of books. In order to add this type of badge in the system, the admin chooses the type from the drop down list and then adds the number of books.

2. Books To Read

   This badge type targets the number of books a user wants to read. In order to be awarded this type of badge a user must have on their want to read shelf the amount of books specified. For admins the process of adding this type of badge to the system is similar to the previous type: they choose the type form the dropdown list and then choose the number of books.

3. Author Count

   Users can earn this type of badge for reading a certain number of books by a particular author. Author count badges target fans of a certain author. In order to add this kind of badge to the system, an admin must first choose the type from the

drop down list, then choose the number of books a user has to read in order to earn the badge and, finally, add the name of the author in the extra information field.

4. Genre Count

   Genre Count badges work similarly to author count badges. They are made for fans of a certain genre and they must read a certain number of books in that genre in order to earn the badge. For admins the process of adding one of these badges in the system is also similar to the author count badges. First, they need to select the type, then add the target number of books and, in the end, in the extra information field fill in the genre.

## 4.2 Organizing and Reviewing Books

At its core, Codex is still a system for organizing and reviewing books. The system is designed to provide users with a simple way to manage their reading lists and share their thoughts through reviews and ratings.

### 4.2.1 Books

Codex is built around books and the database of books is at the heart of the project.

Figure 4.8 shows the index page. Here, books are displayed in a visually appealing format, with three books per line and nine books in total per page. This layout is designed to ensure that users can easily browse the titles without feeling overwhelmed. Each book is presented in a card that contains the cover image, the title and author of the book, the average rating and a “Show details” button. Clicking the button, the user is redirected to the show view for the book that contains more information about the book.

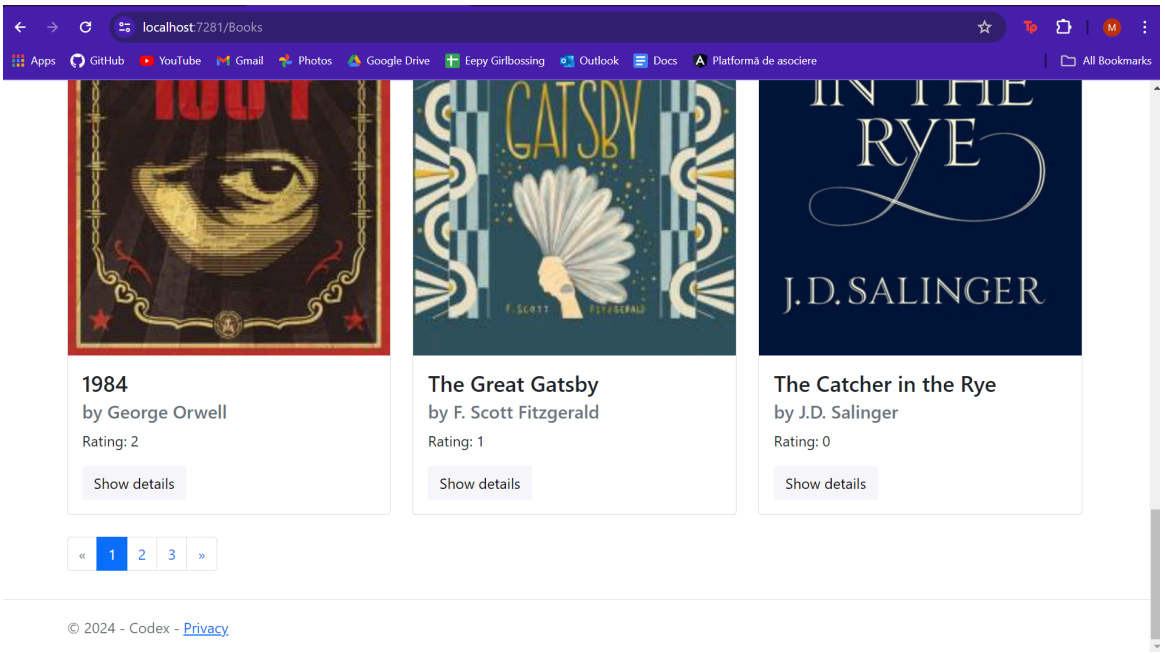

*Figure 4.8 - The pagination on the Books Index page*

The detailed view provides comprehensive information about the book, including the information previously displayed in the index page. Along with that information, the show view also displays the book's synopsis, genre, number of pages and publication date. Furthermore, from this view users can select a shelf they want to add the book on and also read and add reviews, as depicted in Figure 4.9.

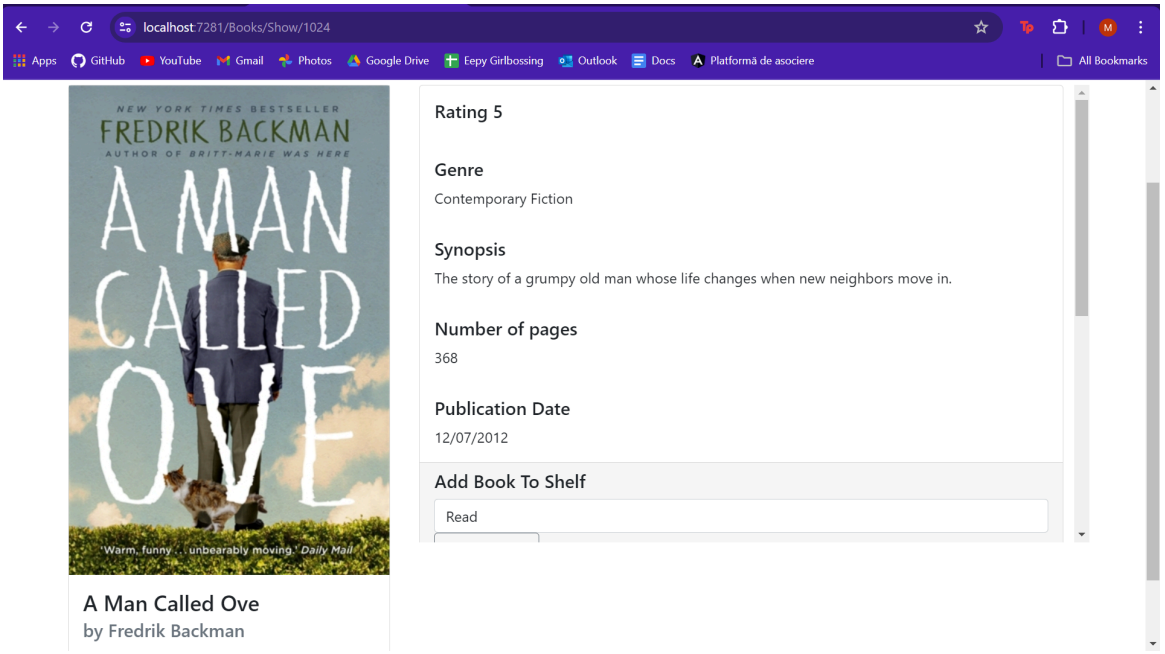


*Figure 4.9 - A book's Details page*

## 4.2.2 Shelves

Shelves offer users a structured way to organize their books. When creating an account a user gets assigned three default shelves: read, want to read and currently reading. As shown in figure 4.10, these shelves are displayed on their profiles with buttons that facilitate access to the books on the shelves.

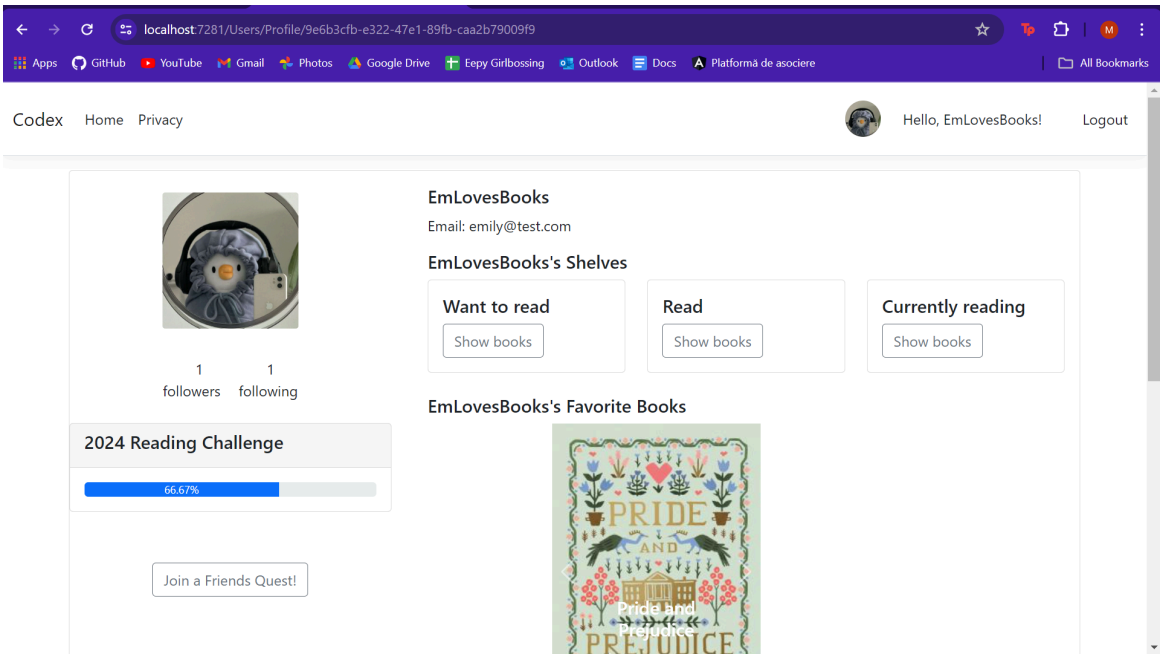


*Figure 4.10 - The shelves displayed on a user's profile page*

### 4.2.2.1 The Want to Read Shelf

The want to read shelf contains the books a user has decided they might want to read one day. The show view of this shelf displays the books three per row in cards, just like in the books index view.

Figure 4.11 illustrates how books are displayed on a user's want to read page.

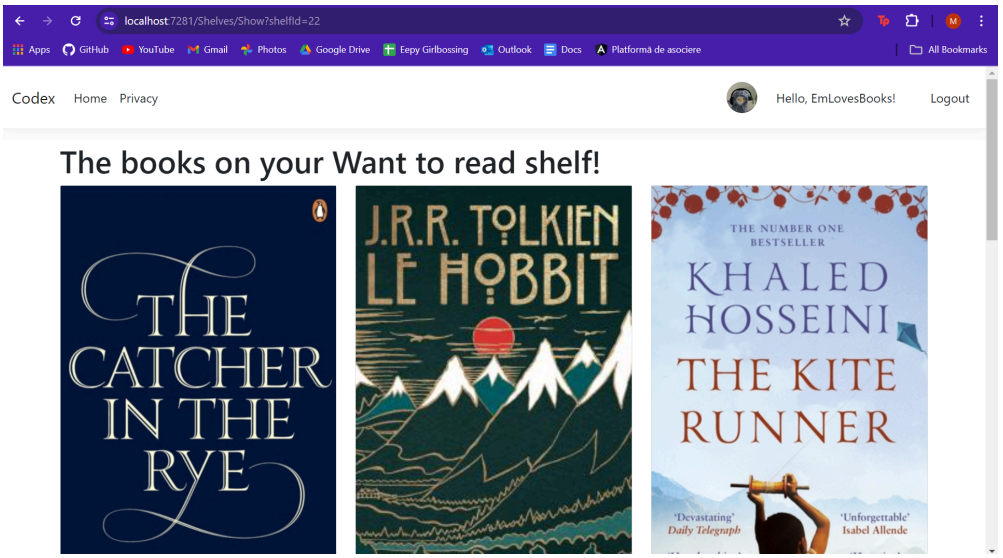


*Figure 4.11 - The Want to Read shelf*

#### 4.2.2.2 The Read Shelf

As the name says, the read shelf contains all the books a user has read. Figure 4.12 showcases the show view of a user's read shelf. This page's layout is similar to the want to read shelf, displaying the books in three cards per row.

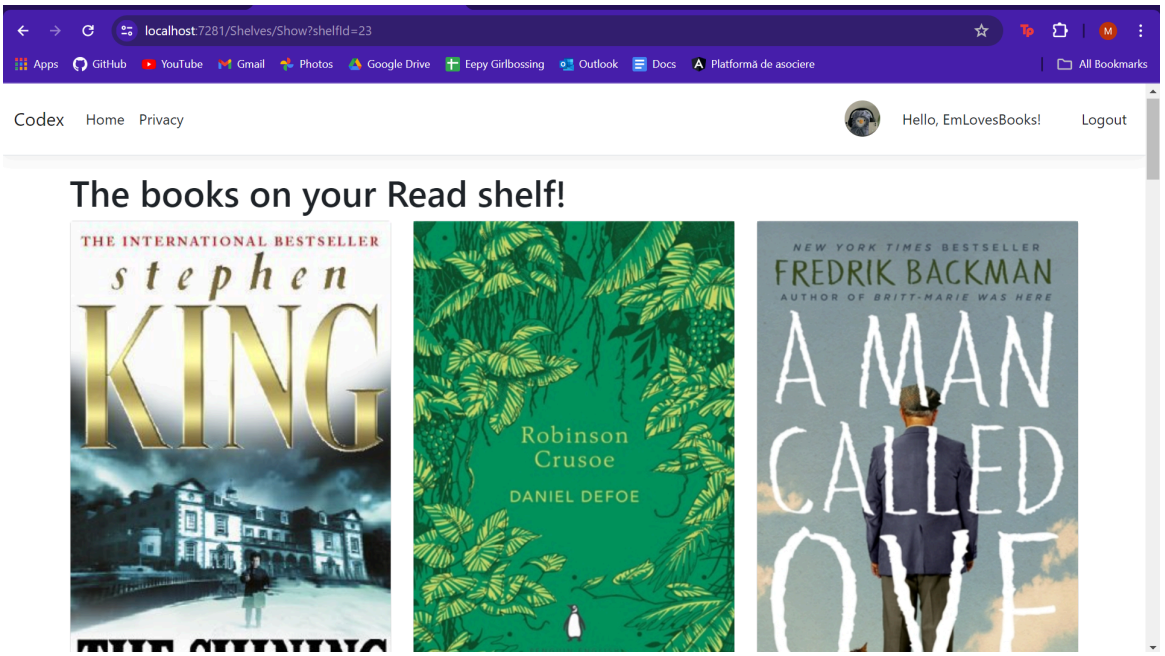


*Figure 4.12 - The Read shelf*

#### 4.2.2.3 The Currently Reading Shelf

When a user starts reading a book, they add it to their currently reading shelf. Here they can keep track of their progress through the book.

Figure 4.13 represents a user's Currently Reading shelf. Similarly to the other shelves, books are displayed maximum three in a row. What sets this page apart from the others is the Update Progress button, as well as the progress bar that serves as a visual representation of how far into the book the user is.

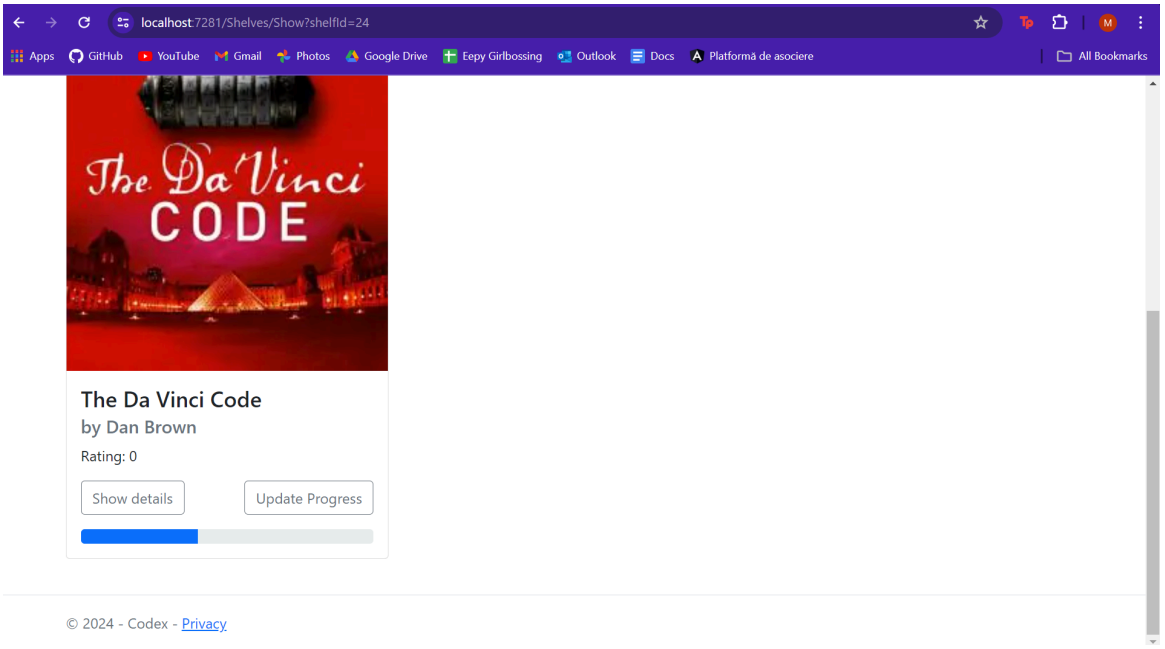


*Figure 4.13 - The Currently Reading shelf*

In order to update their progress, the user clicks the button and then adds the page they got to into the input field, as shown in Figure 4.14. Once they click save the progress is updated and the progress bar shows how close they are to finishing the book.

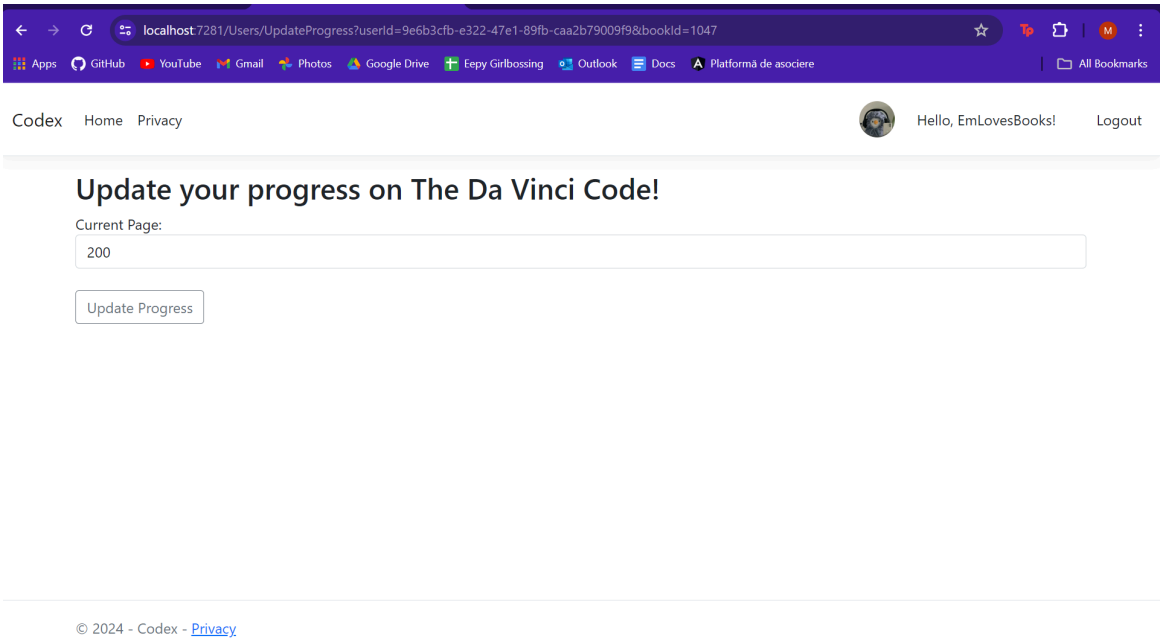


*Figure 4.14 - The update progress view*

### 4.2.3 Reviews and Star Ratings

Reviews and star ratings are fundamental features of Codex, allowing users to share thoughts and insights into the books they read, contributing to the sense of community the platform fosters.

As shown in Figure 4.15, reviews are displayed on the book's details page, right under the option to select a shelf to add the book to. There, every review has its own card where the rating and the comment is shown along with the name of the user and the date and time when the review was posted. Other than this information, the user that posted the review can also see two buttons that allow them to edit and delete their reviews if they choose to.

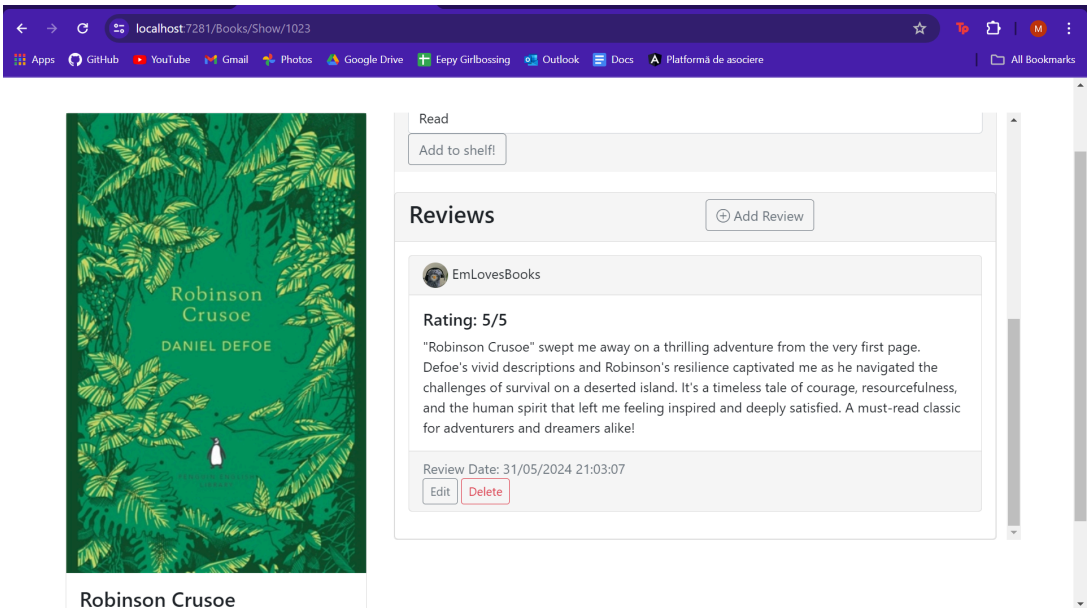


*Figure 4.15 - The reviews on the details page of a book*

To add a new review a user just clicks the add review button and is redirected to the form where they can fill in the details, as shown in Figure 4.16. It's not mandatory for the user to include a comment, but in order to be able to post a review they must include a star rating.

Star ratings work on a scale from one to five stars with one being the lowest rating a user can give a book and five being the highest. As of right now, Codex does not offer the option of half ratings.

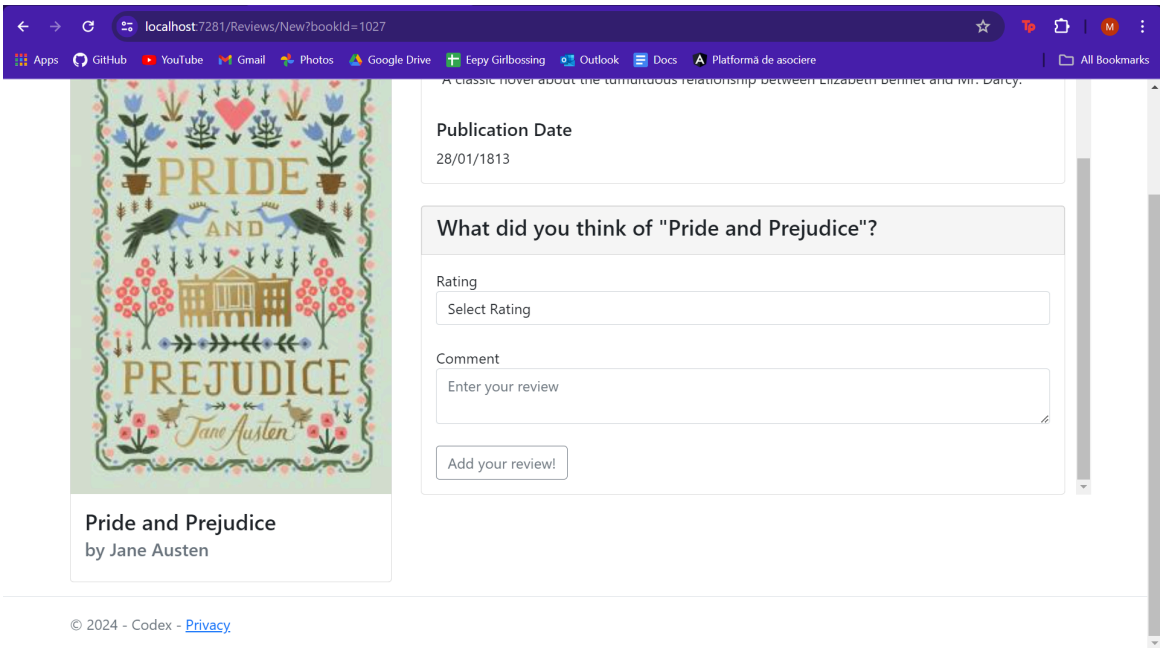


*Figure 4.16 - The process of adding a new review*

## 4.3 Book Tracking and Statistics

Book tracking and statistics are features meant to encourage users to read more and engage with the platform at the same time. By providing visual representations of their progress and insights into their reading habits, these features are meant to help users set goals, track their achievement and reflect on their reading journeys.

### 4.3.1 Reading Challenge

The reading challenge is meant to motivate users to read more throughout the year. At the beginning of each year, users can join a new reading challenge where they get to set a target number of books they aim to read. Each time they finish reading a book, it automatically counts towards their goal providing a great incentive to keep going.

As Figure 4.17 shows, if a user has not joined the challenge, they have a button on their profile that allows them to do so.

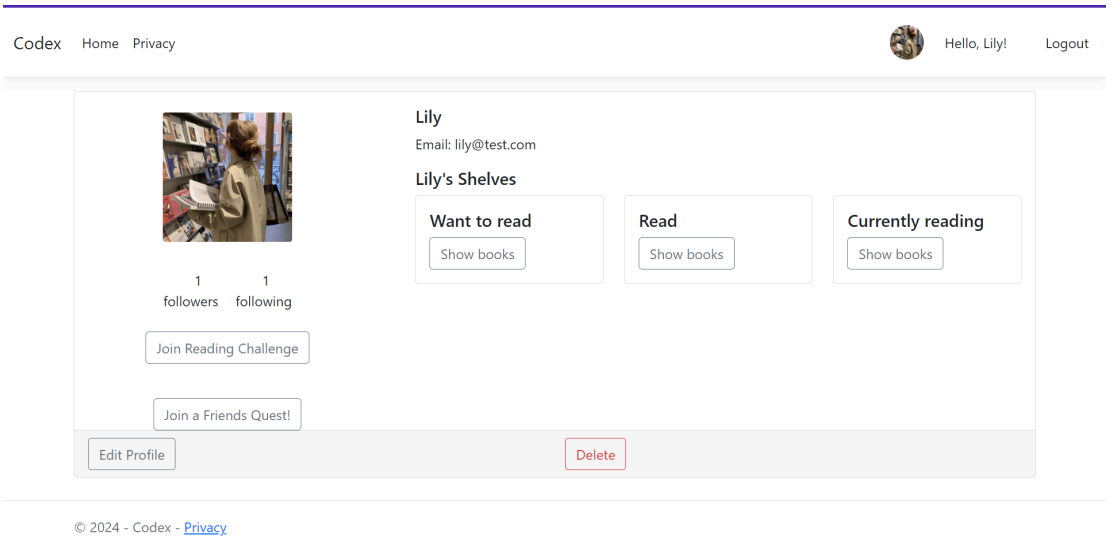


*Figure 4.17 - The profile page of a user who has not joined the reading challenge*

Once a user clicks the button, they are redirected to form where they get to choose their goal for the year. This form is pictured in Figure 4.18.

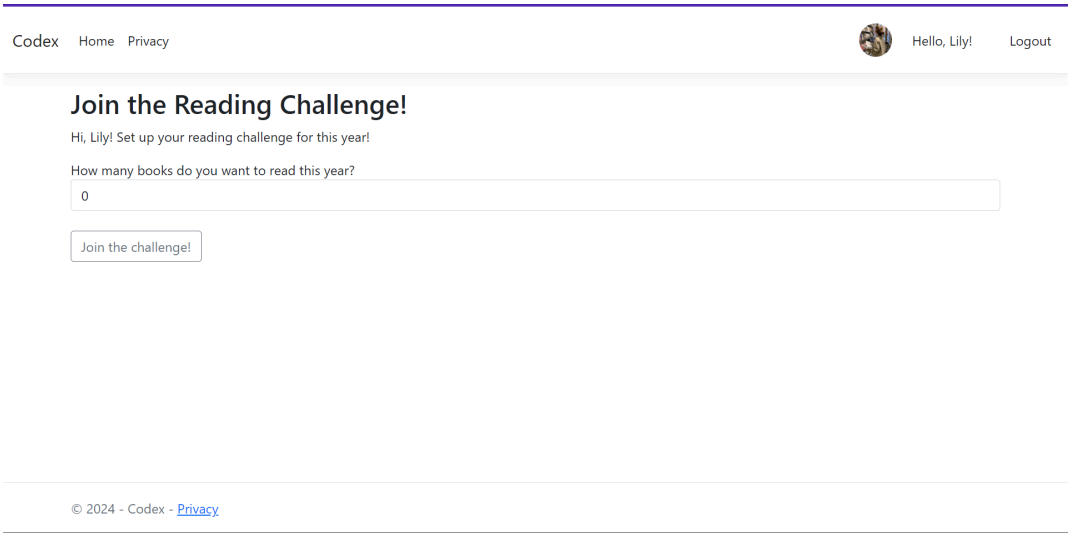


*Figure 4.18 - A screenshot of the process of joining the reading challenge*

Once the reading challenge is set up, on the user's profile there is a progress bar displayed with the user's progress towards their goal, as shown in Figure 4.19. Every time a user moves a book from their currently reading shelf to their read shelf, the book counts towards their reading challenge and they can see the progress bar fill up.

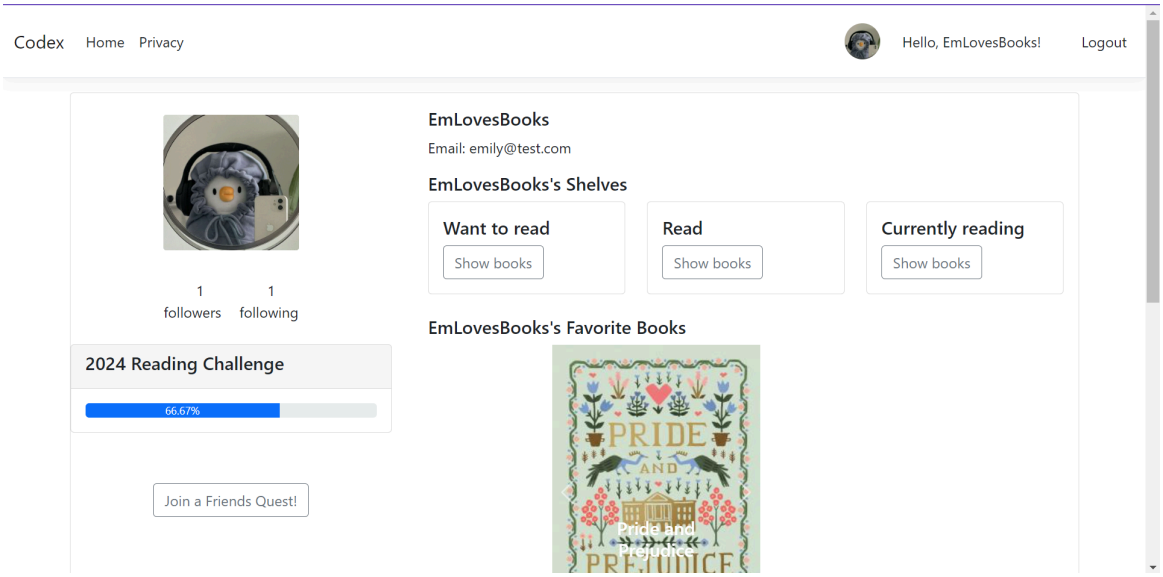


*Figure 4.19 - A screenshot showing the progress bar of the reading challenge in the user's profile*

When clicking on the title of the reading challenge card, the user is redirected to a page that contains a list of all the books they read that year, as well as buttons for editing their goal or deleting the challenge altogether. This page is pictured, below, in Figure 4.20.

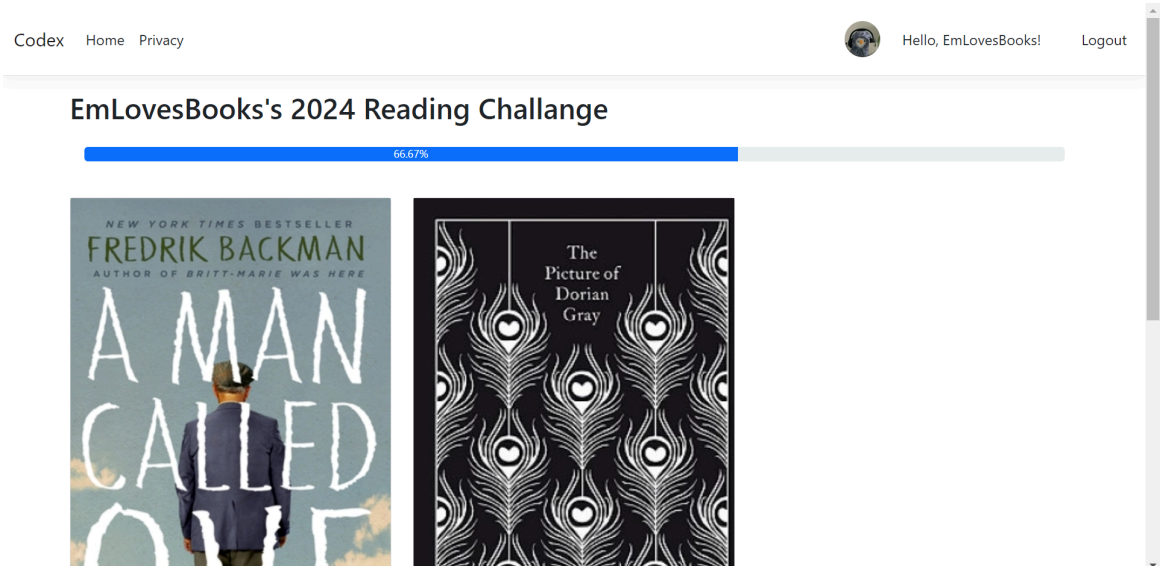


*Figure 4.20 - The Show of the Reading Challenge*

### 4.3.2 Yearly Recap

The yearly recap is designed for the users who participated in the reading challenge. At the end of the year they can see a personalized summary of their reading including detailed information about their reading habits.

Figures 4.21, 4.22, 4.23 make up the yearly recap page, which includes a lot of interesting statistics such as:

- Pages Read: The total number of pages read throughout the year.
- Books Read: The total number of books read throughout the year.
- Shortest and Longest Books Read: The longest and shortest books read are highlighted along with information about the number of pages.
- Average Book Length: The number of pages of an average book read that year.
- Average Rating: The average rating of the year.
- Highest Rated Book: The book that Codex users thought was the best.
- All Books Read: A list of all the titles the user read over the year


*Figure 4.21 - Part of the Yearly Recap page*

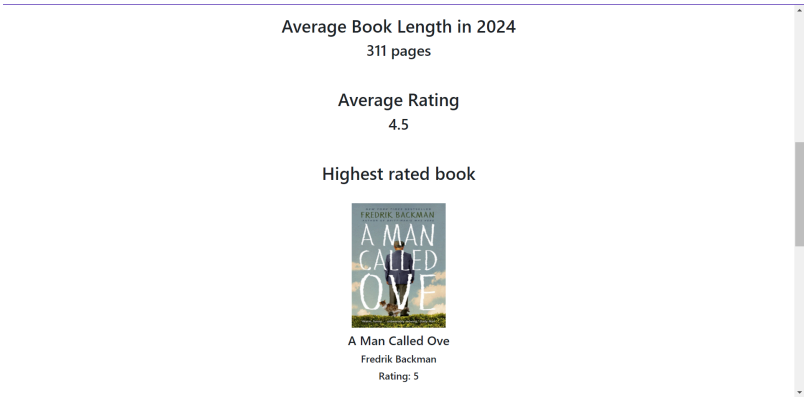


*Figure 4.22 - Part of the Yearly Recap page*

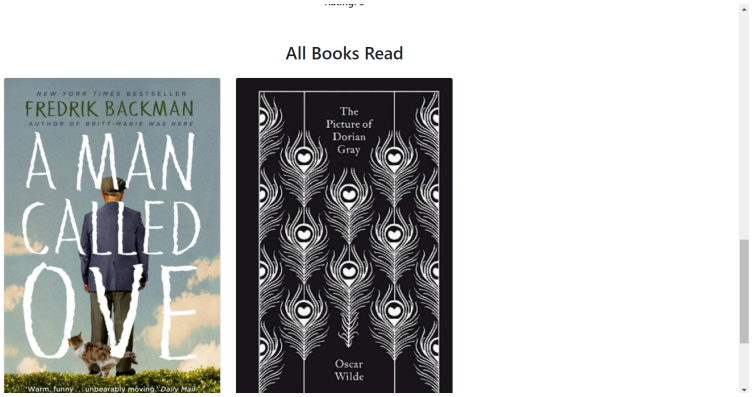


*Figure 4.23 - Part of the Yearly Recap page*

## 4.4 Connections

A big part of Codex is fostering connections between users, because reading is a social habit and it's much better when people can share their experiences. By integrating social features, Codex encourages users to engage with others and share their reading journeys and find motivation to read through shared goals and activities.

### 4.4.1 Friends

Codex allows users to build a network of friends within the platform. Users can follow each other, and when both users follow each other back, they are officially considered friends.

When checking out another user's profile, the current user can see their following and follower count  as well as a button that allows them to follow that user, or unfollow them if they are already following said user. This feature is pictured below, in Figure 4.24.

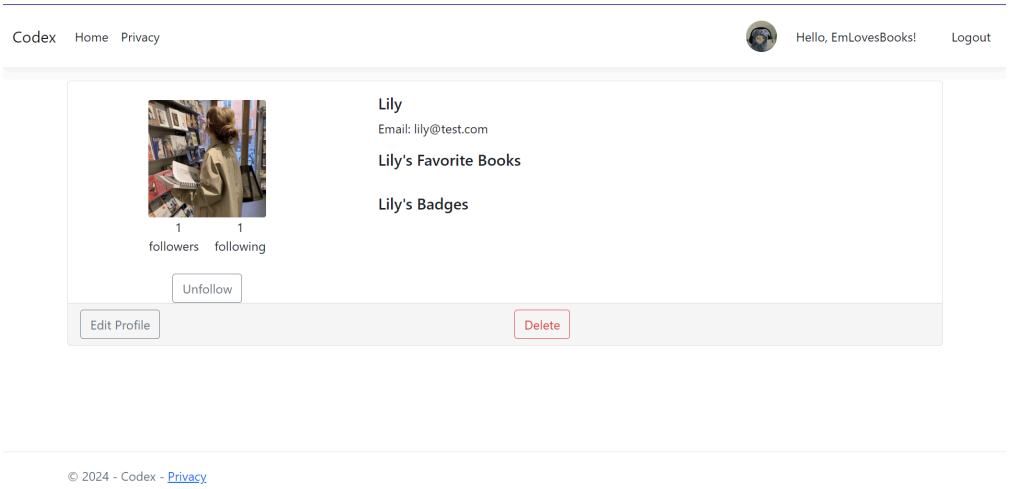


*Figure 4.24 - The profile of a user, showcasing the following feature*

## 4.4.2 Friends Quests

Friends quests are designed to make reading more rewarding and engaging. Every week users can choose to participate in one of these challenges with a friend and they get assigned a random number of pages they have to read in total throughout the week.

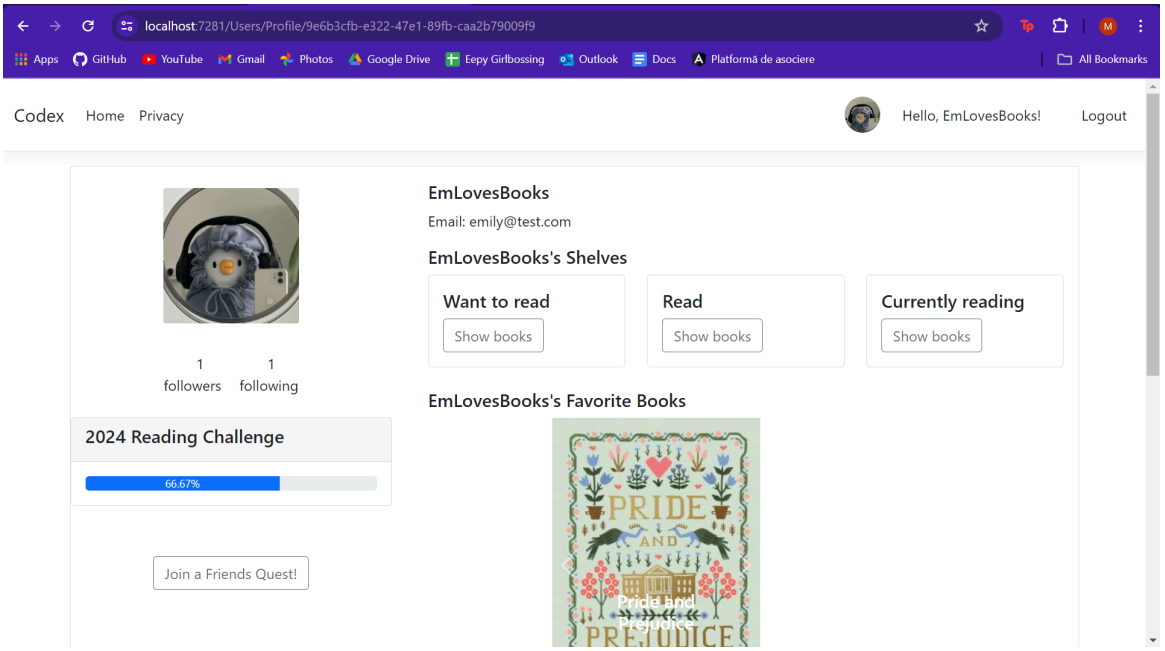


*Figure 4.25 - The profile of a user who has not joined a Friends Quest in the current week*

In order to join a friends quest the user can click the button on their profile page, pictured in Figure 4.25, which is gonna redirect them to another page where they can choose the friend that they want to collaborate with on the friends quest, pictured in Figure 4.26..

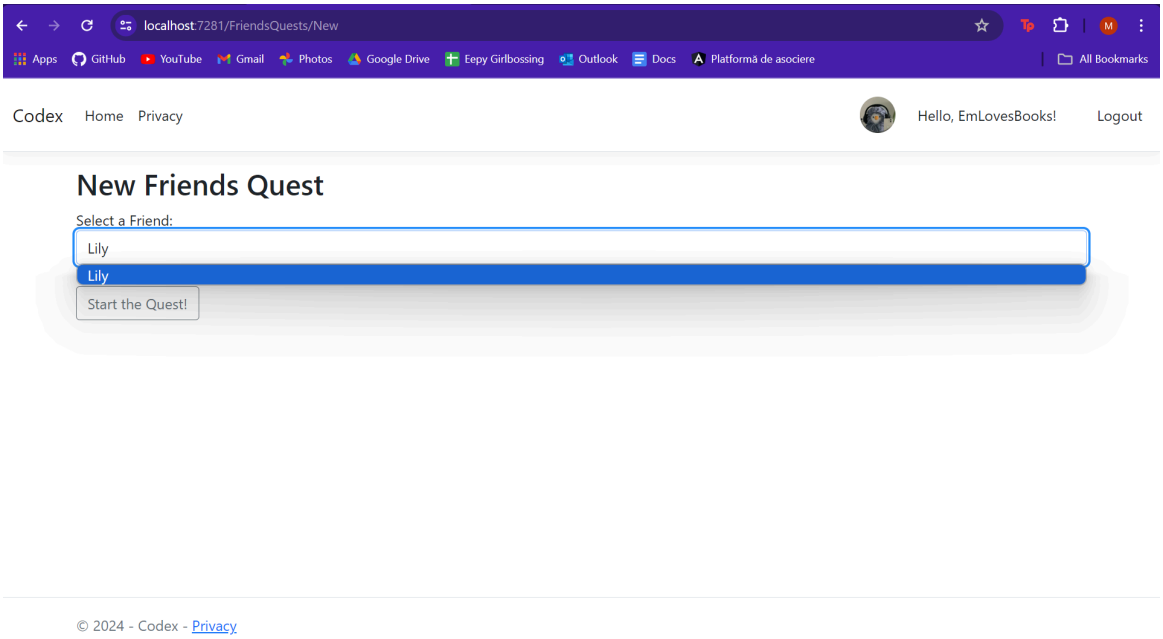


*Figure 4.26 - The process of joining a Friends Quest*

Once the friend is chosen out of the list of available friends for that week the two friends get assigned a number of pages they have to read for the week. The users must work together to reach their target. Whenever a user reads out of a book, they update their progress in their currently reading page and the number of pages they read counts towards the friends quest, as pictured in Figure 4.27.

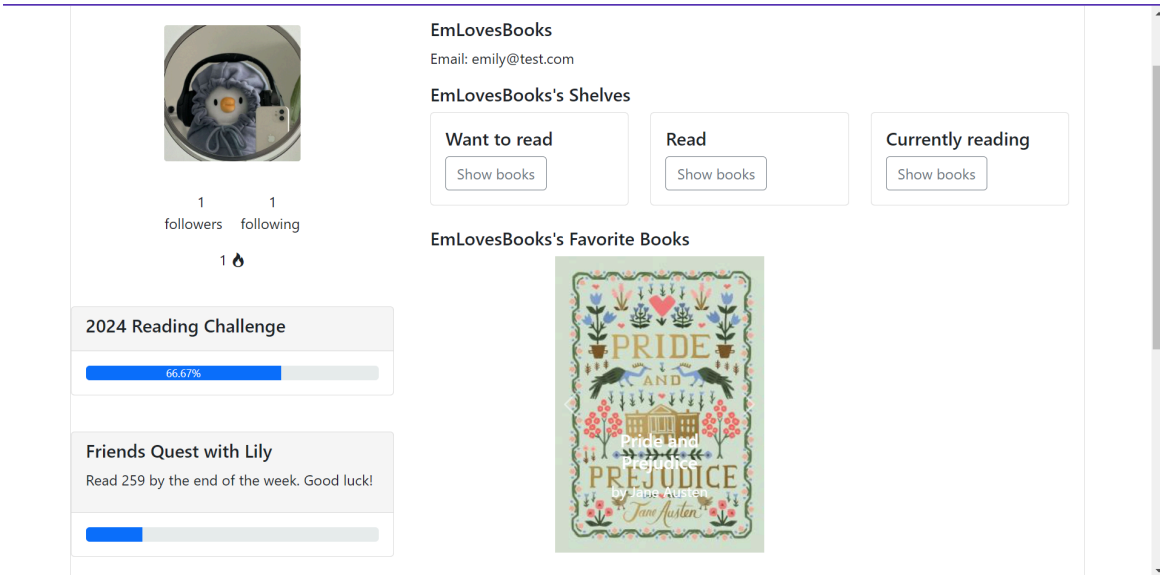

*Figure 4.27- The progress bar of a Friends Quest in a users profile*

### 4.4.3 Streaks

Streaks are another motivational feature of Codex. Streaks count the amount of consecutive days a user has read over 30 pages, offering daily motivation to read daily in order to maintain the streak.

The user's streak count can be found in their profile and it's displayed as the number of days they've consistently read  over 30 pages followed by a little fire icon, as shown in Figure 4.28.

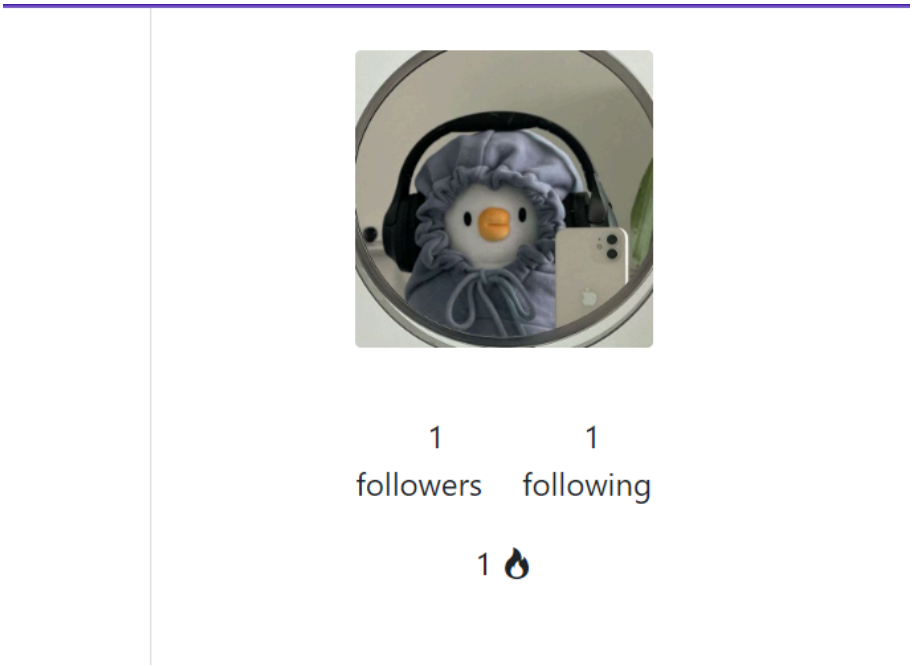


*Figure 4.28- The streaks displayed on a user's profile*

### 4.4.4 Buddy Reads

Buddy Reads are a great way for users to read books virtually along with their friends. This feature allows users to decide on a book to read together and then read in sync while leaving comments with their thoughts on the book for their friends to see once they get to that part of the book.

In order to start a Buddy Read a user chooses the book they want to read and the friends that want to participate in the buddy read. This process is pictured below in Figure 4.29.

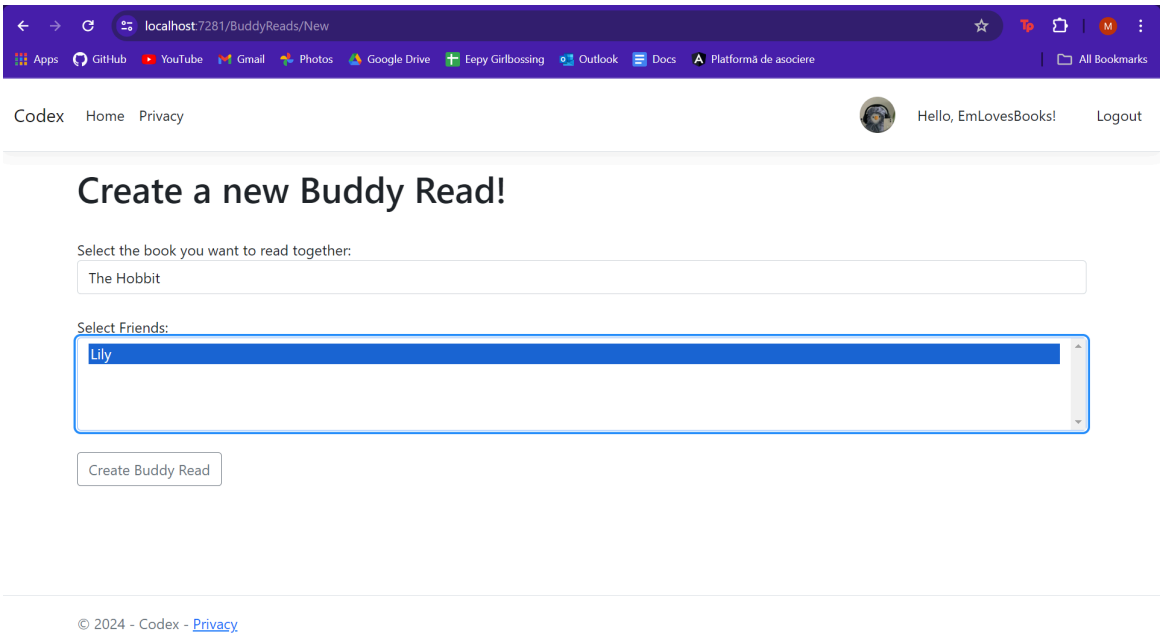


*Figure 4.29 - The process of joining a Buddy Read*

Once the buddy read is started the book is automatically added to the users currently reading page and the friends can see each other's progress on the dedicated page.

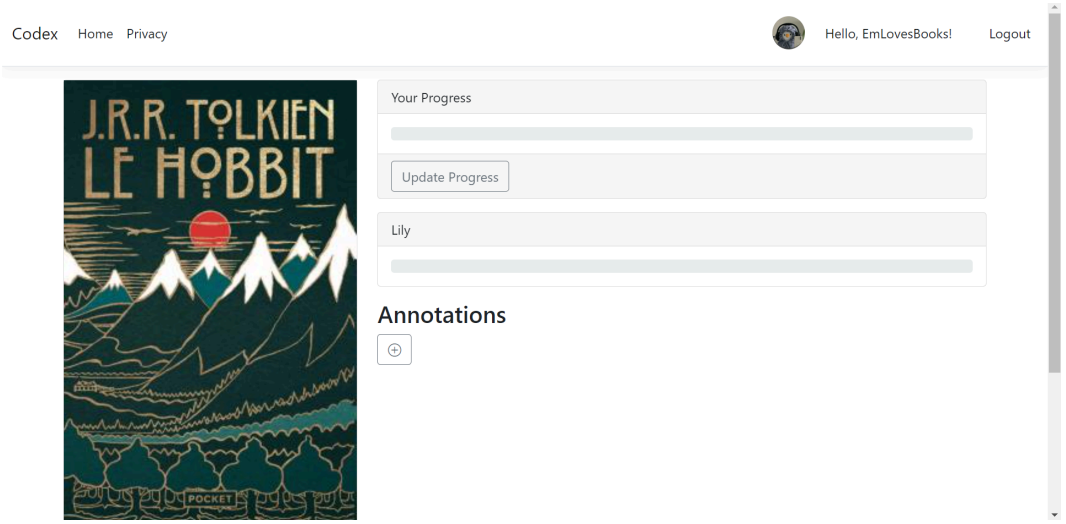


*Figure 4.30 - A screenshot of the Buddy Read page*

Users can see their progress and also update their own from the dedicated page. This is an important aspect because of the annotations feature. The annotations feature allows users to share thoughts on the book without being afraid of spoilers. The annotations only become visible to the other users once they've passed that point in the book too.

First a user clicks on the plus button to add a new annotation pictured in Figure 4.30. This button sends them to a page where they can fill in a form, providing their thoughts but also the page they want to leave the note on.

After they leave the annotation, it will be displayed on the BuddyRead page, just like in Figure 4.31.

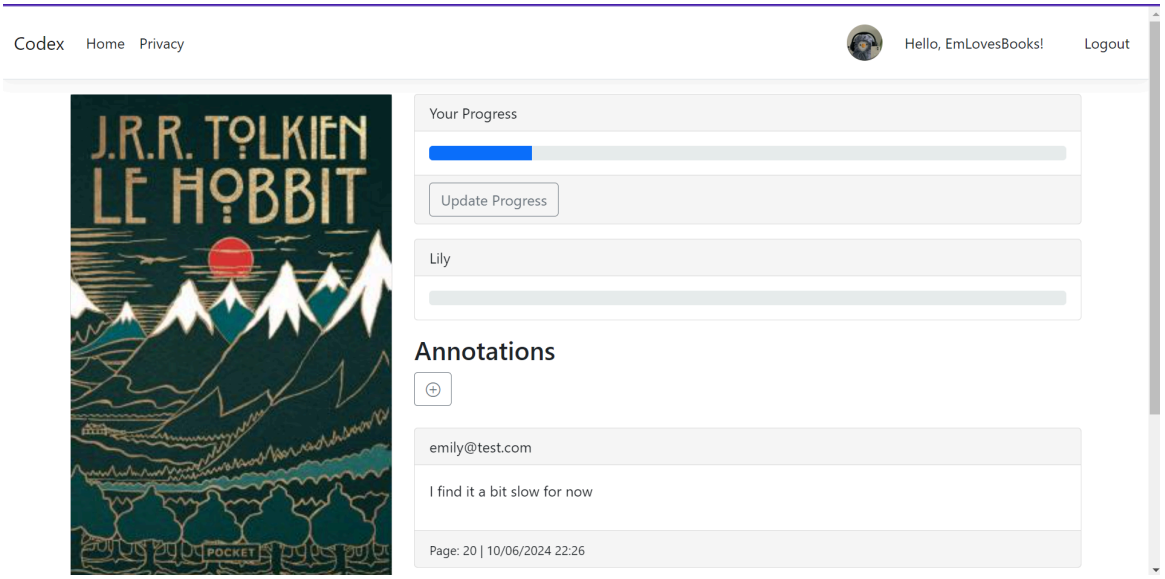


*Figure 4.31 - A screenshot of the annotations on the Buddy Read page*

# Chapter 5: Conclusions

As an avid reader, I have tried multiple applications designed for readers, but I've always found them to be lacking in certain areas. This is what sparked the idea of Codex. With this project I wanted to share my love of reading with others by creating a web application that would go beyond the basic reading tracker functionalities. I wanted an application that would allow me to showcase my favorite books on my profile and also a more fun way to engage with others and read together.

The process of developing Codex was both exciting and challenging. It was really exciting to see my vision come to life, but each part of development came with its own challenges. Despite that, I think the end result is something to be proud of.

I poured a lot of myself into this project and, in return, it taught me a lot. Reflecting on my journey, and by this I am not exclusively referring to the development process of Codex, but also the past three years of studying computer science, I've learned patience and perseverance, I learned to embrace unexpected changes, but most importantly I learned how important it is to work on a project you really are passionate about.

Moving forward, I still have several ideas for future features Codex could include. Even though the project already has so many features, there is still room for improvements. Codex has great potential for further growth and here are some of my key ideas and perspectives for the future of the application: implementing book clubs, more advanced analytics, seasonal challenges, an improved user interface, support for different editions of the same title. This list only has a couple of my ideas, which goes to show how much potential there is in Codex and in its mission. I think that by continuing to adapt to user needs and by bringing innovative ideas to them, Codex could really make an impact on people's reading journeys.

# References


[1] Bootstrap, *Alerts.* Available: https://getbootstrap.com/docs/4.0/components/alerts/. Last accessed: 9 June 2024.
[2] Bootstrap, *Bootstrap Documentation*. Available: https://getbootstrap.com. Last accessed 1 June 2024.
[3] Bootstrap, *Buttons.* Available: https://getbootstrap.com/docs/4.0/components/buttons/. Last accessed: 8 June 2024.
[4] Bootstrap, *Bootstrap grid examples*. Available: https://getbootstrap.com/docs/4.0/examples/grid/. Last accessed 8 June 2024.
[5] Bootstrap, *Cards.* Available: https://getbootstrap.com/docs/4.0/components/card/. Last accessed 8 June 2024.
[6] Bootstrap, *Bootstrap Icons*. Available: https://icons.getbootstrap.com/. Last accessed: 8 June 2024.
[7] GoodReads, *API.* Available: https://www.goodreads.com/api. Last accessed 1 June 2024.
[8] GoodReads, *About Goodreads*. Available: https://www.goodreads.com/about/us. Last accessed 1 June 2024
[9] Lucidchart, *What is an Entity Relationship Diagram (ERD)?*. Available https://www.lucidchart.com/pages/er-diagrams. Last accessed 15 June 2024.
[10] Microsoft, DbContext Class. Available: https://learn.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbcontext?view=efcore-8.0. Last accessed 4 June 2024
[11] Microsoft, *Entity Framework Core*. Available: Overview of Entity Framework Core - EF Core | Microsoft Learn. Last accessed 3 June 20204
[12] Microsoft, *Introduction to Identity on ASP.NET Core*. Available: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio. Last accessed 3 June 2024
[13] Microsoft, *Introduction to Razor Page*s. Available: https://learn.microsoft.com/en-us/aspnet/core/razor-pages/?view=aspnetcore-8.0&tabs=visual-studio. Last accessed 4 June 2024
[14] Microsoft, *Migrations Overview*. Available: https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli. Last accessed 4 June 2024
[15] Microsoft*, Overview of ASP.NET Core MVC*. Available: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0&WT.mc_id=dotnet-35129-website . Last accessed 3 June 2024
[16] Microsoft, *Part 5, work with a database in an ASP.NET Core MVC app*. Available: https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/working-with-sql?view=aspnetcore-8.0&tabs=visual-studio. Last accessed: 6 June 2024
[17] Microsoft, *What is ASP.NET Core?*. Available: https://dotnet.microsoft.com/en-us/learn/aspnet/what-is-aspnet-core. Last accessed 2 June 2024

[18] Ragupathi, Mugilan T. S..*Learning ASP.NET Core MVC Programming.* Packt Publishing, 2016.
[19] The SoryGraph, The StoryGraph | Because life's too short for a book you're not in the mood for. Available: https://www.thestorygraph.com/. Last accessed 2 June 2024.