**Outline**

In this project I have built a web app using REST APIs. This project implements user authentication, a home page with links to other pages, there is an about page, it uses an api from a publicly available website to store data in a database, and provides search functionality from the database. This app allows users to view random fun facts and to reroll it so they can keep viewing fun facts. They can also save fun facts as long as they are logged in, and view them later. The app makes use of a publicly available API to retrieve and display these facts. The search function allows users to search by a key term to display all their saved facts with that key term.

**Links and Login**

Application URL: <https://www.doc.gold.ac.uk/usr/691/>

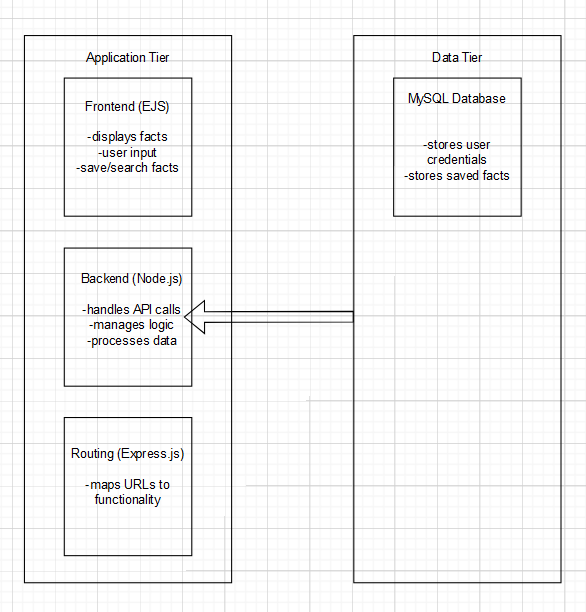
Git repo link: <https://github.com/albngo/dawportfolio.git>

Username: searcher

Password: searcher

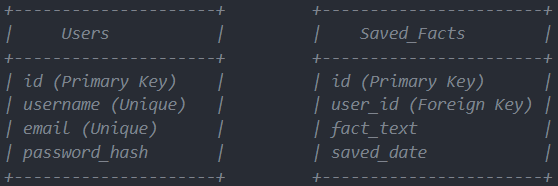
This is a demo account for the app, you can also alternatively register an account but please do not use any real emails or passwords for your safety.

**Architecture**



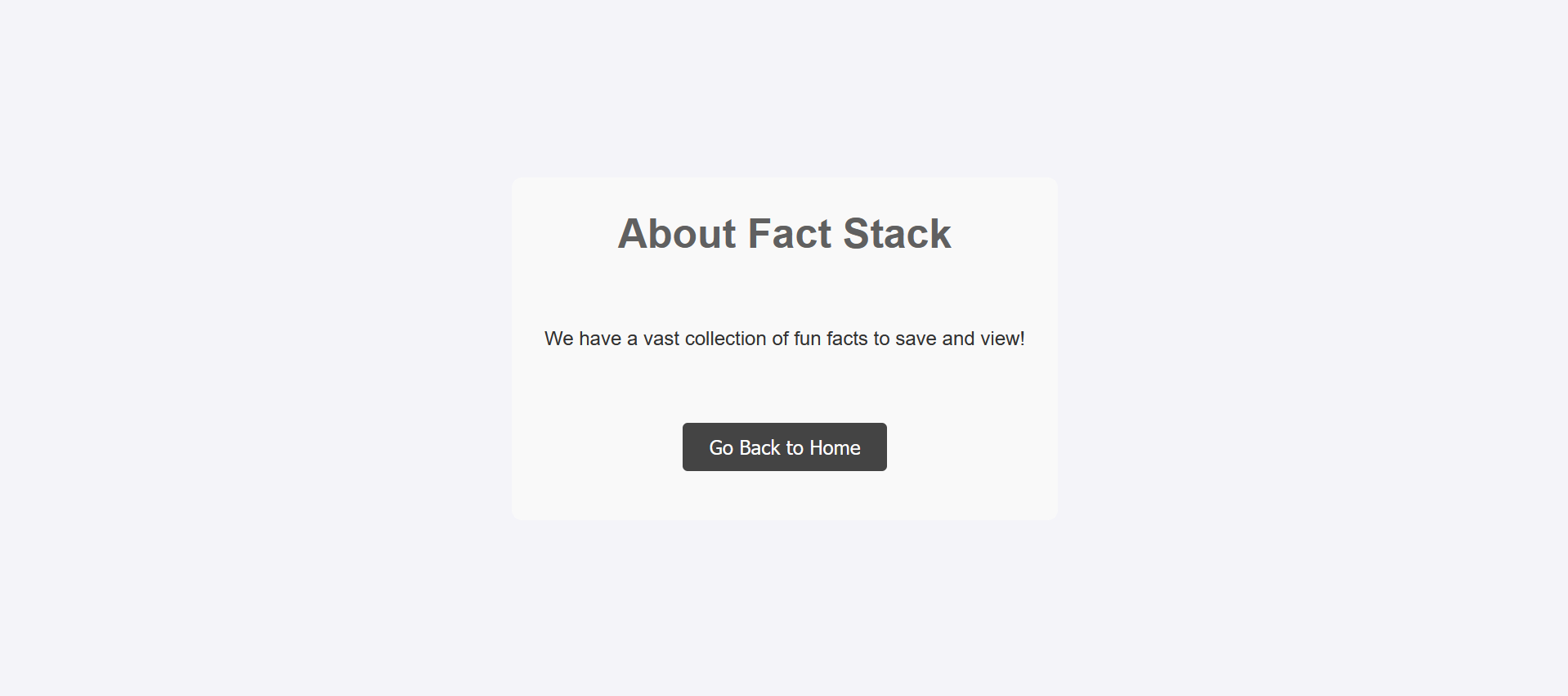
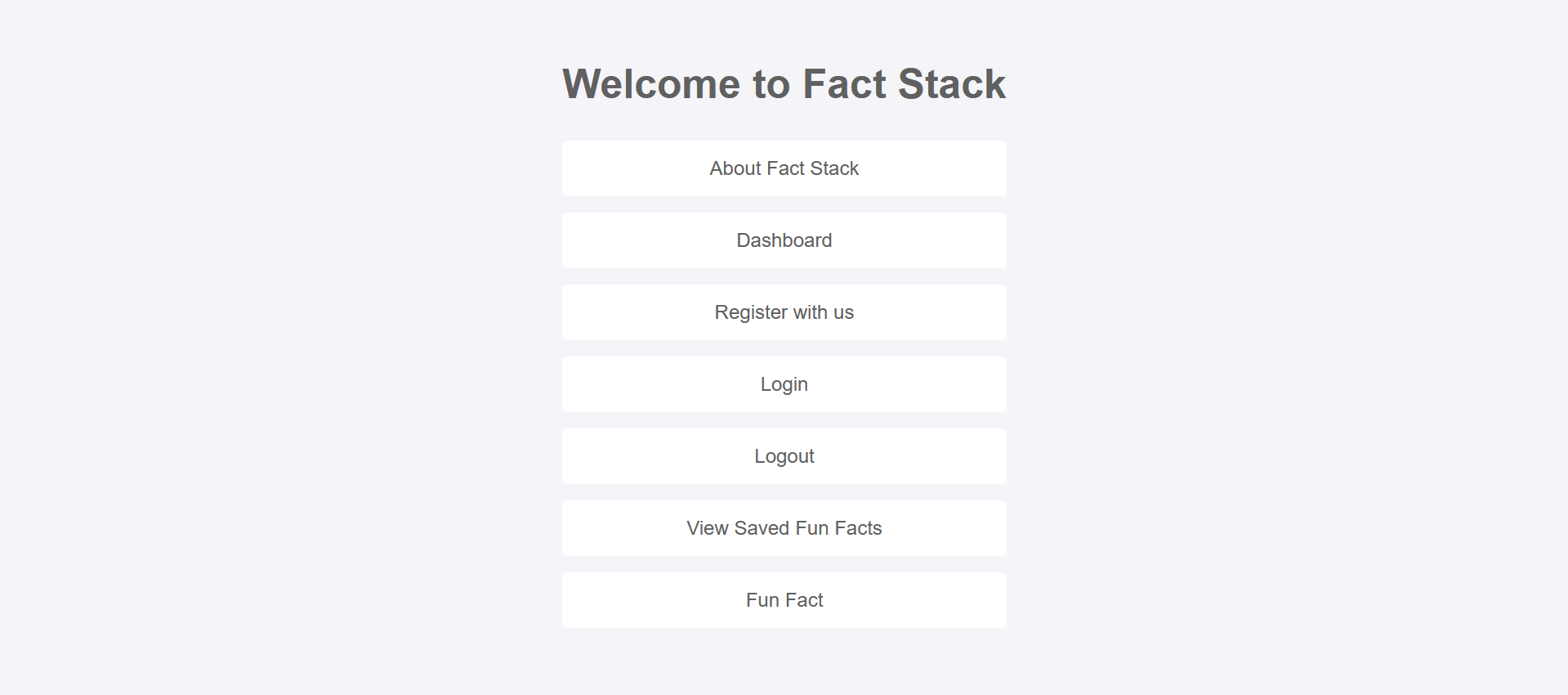
The application is made up of primarily two tiers: Application and Data. The application tier is made up of the frontend which is handled by ejs; backend which is handled by Node.js; and the routing which is handled by Express.js. Now the data tier is handled by a MySQL database that communicates with the application tier effectively and efficiently and stores user credentials and saved facts.

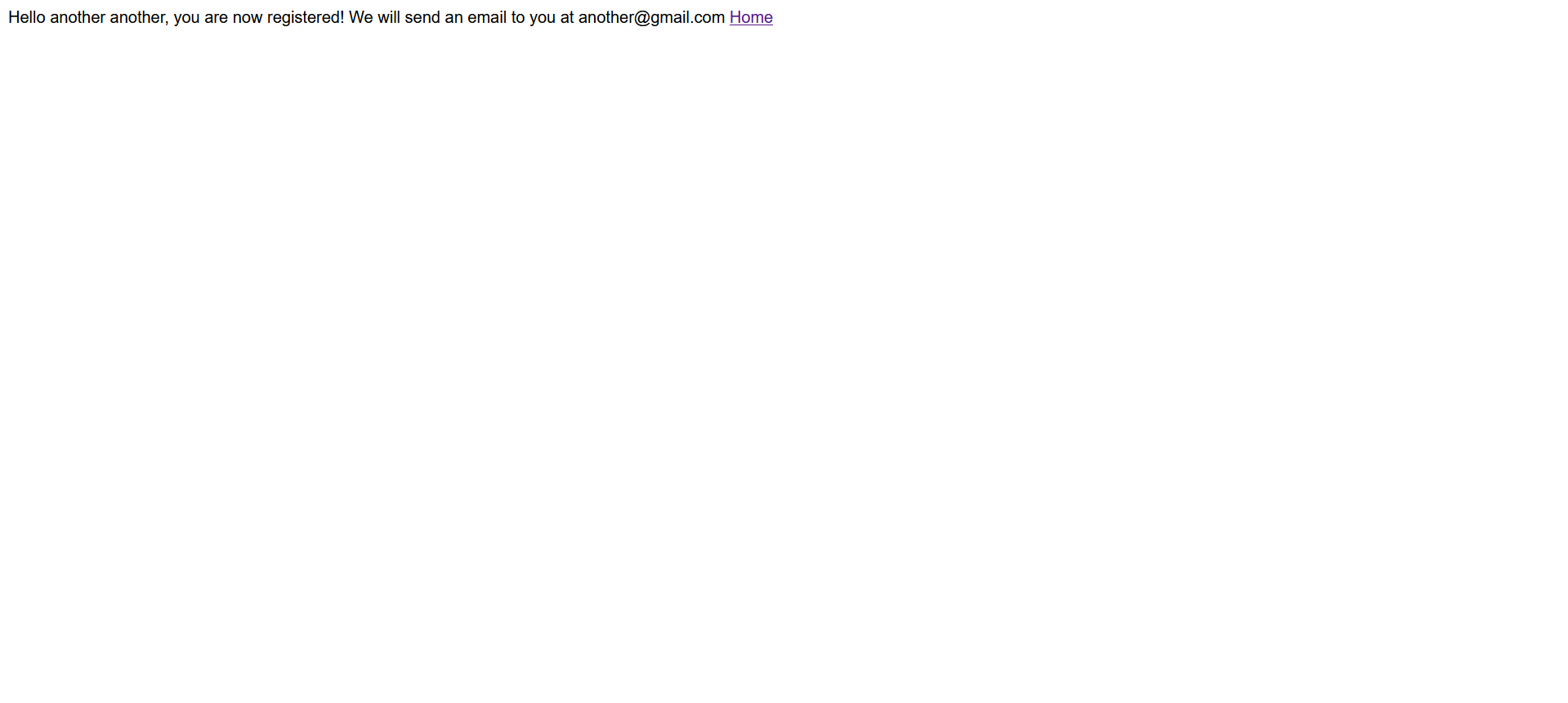
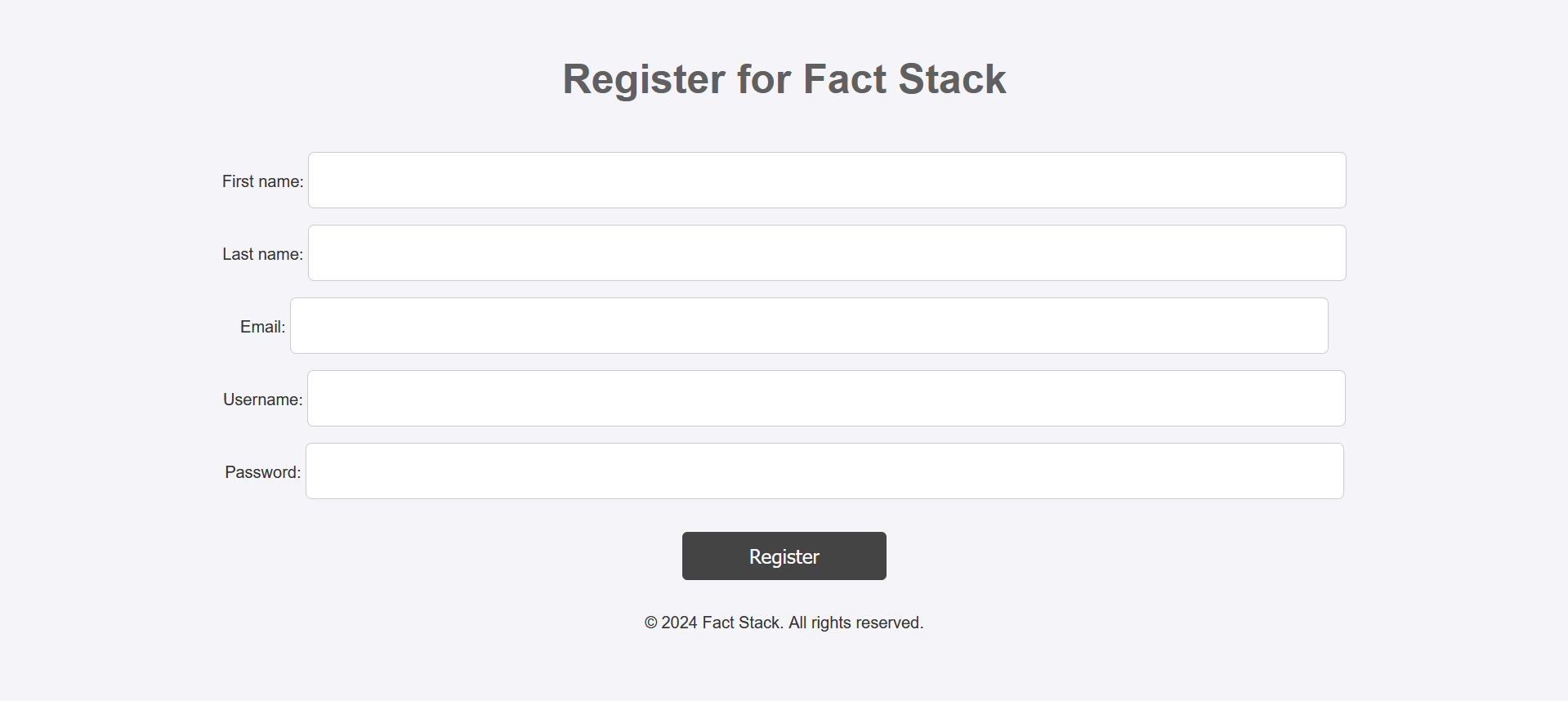
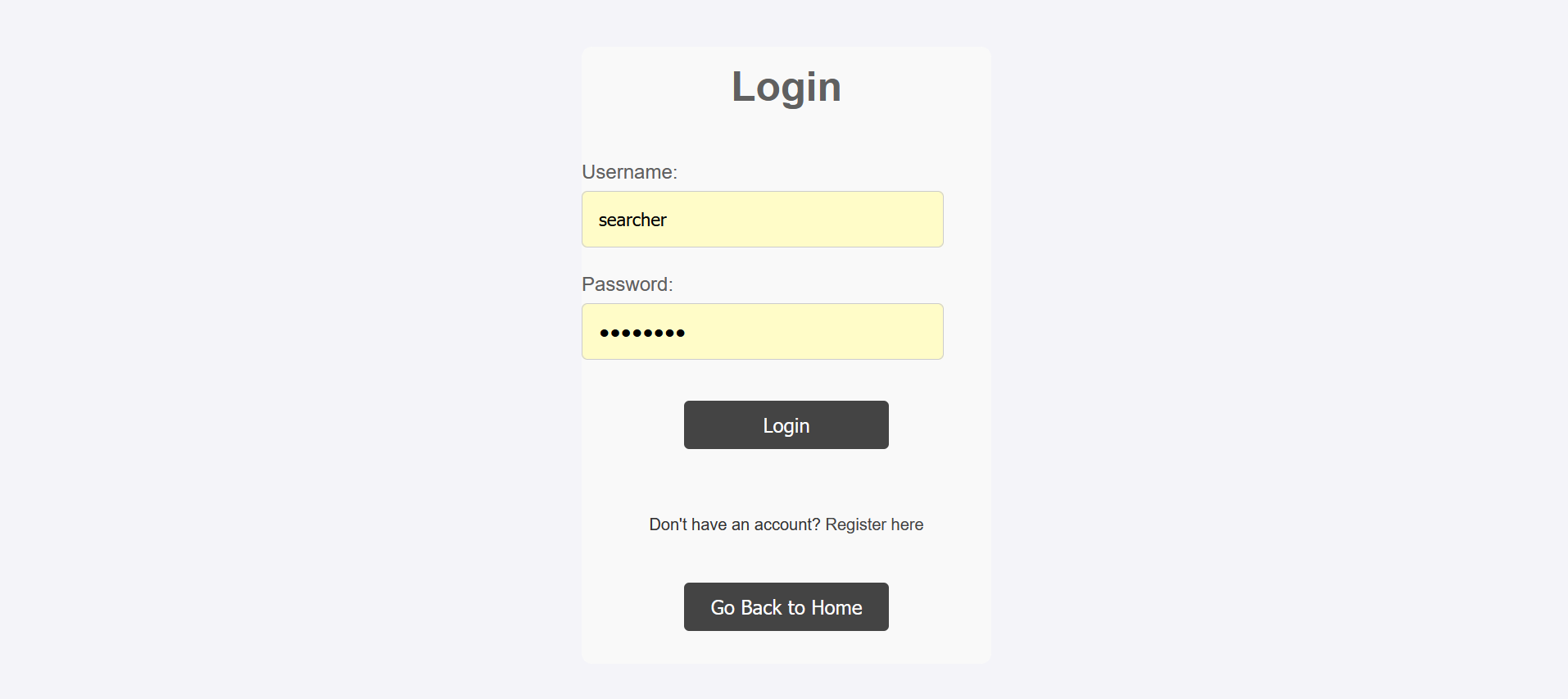
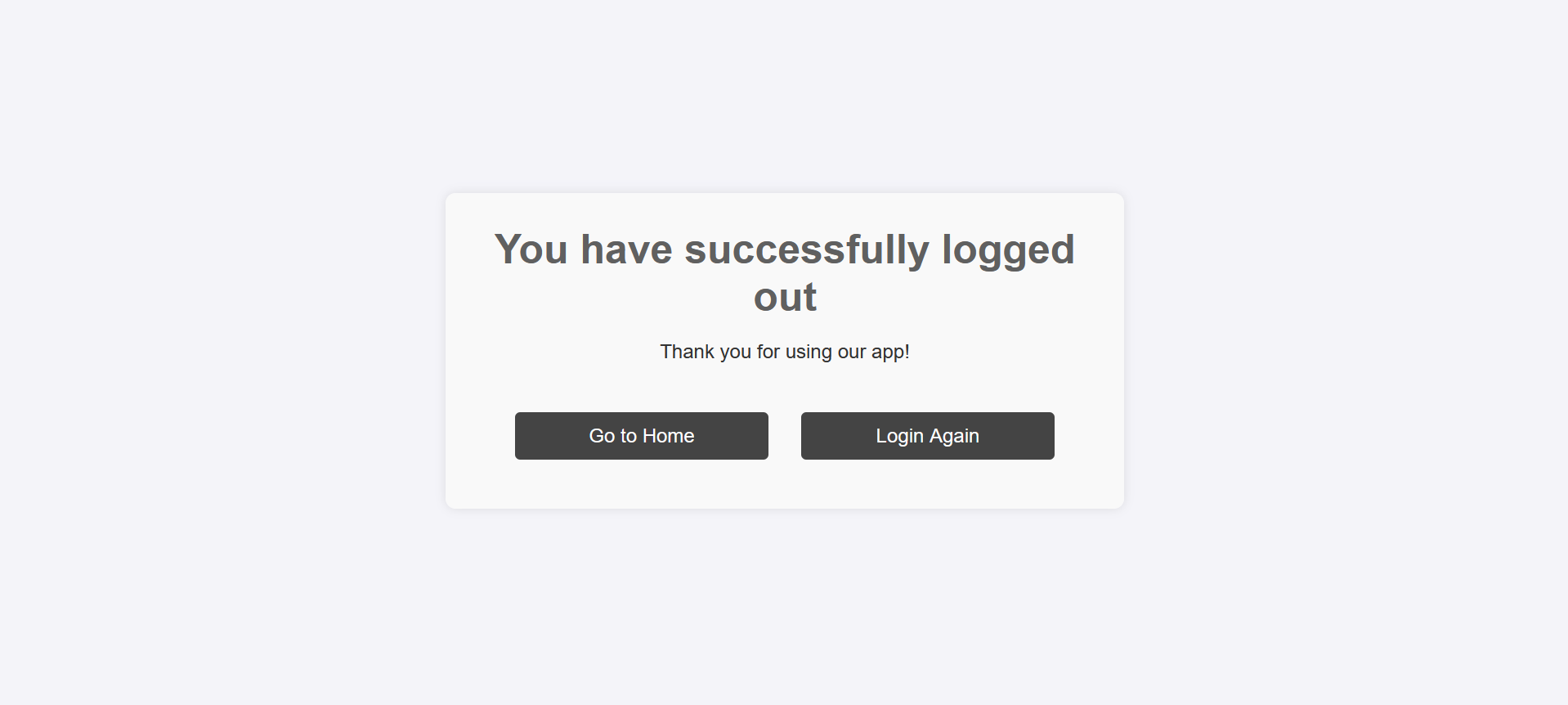
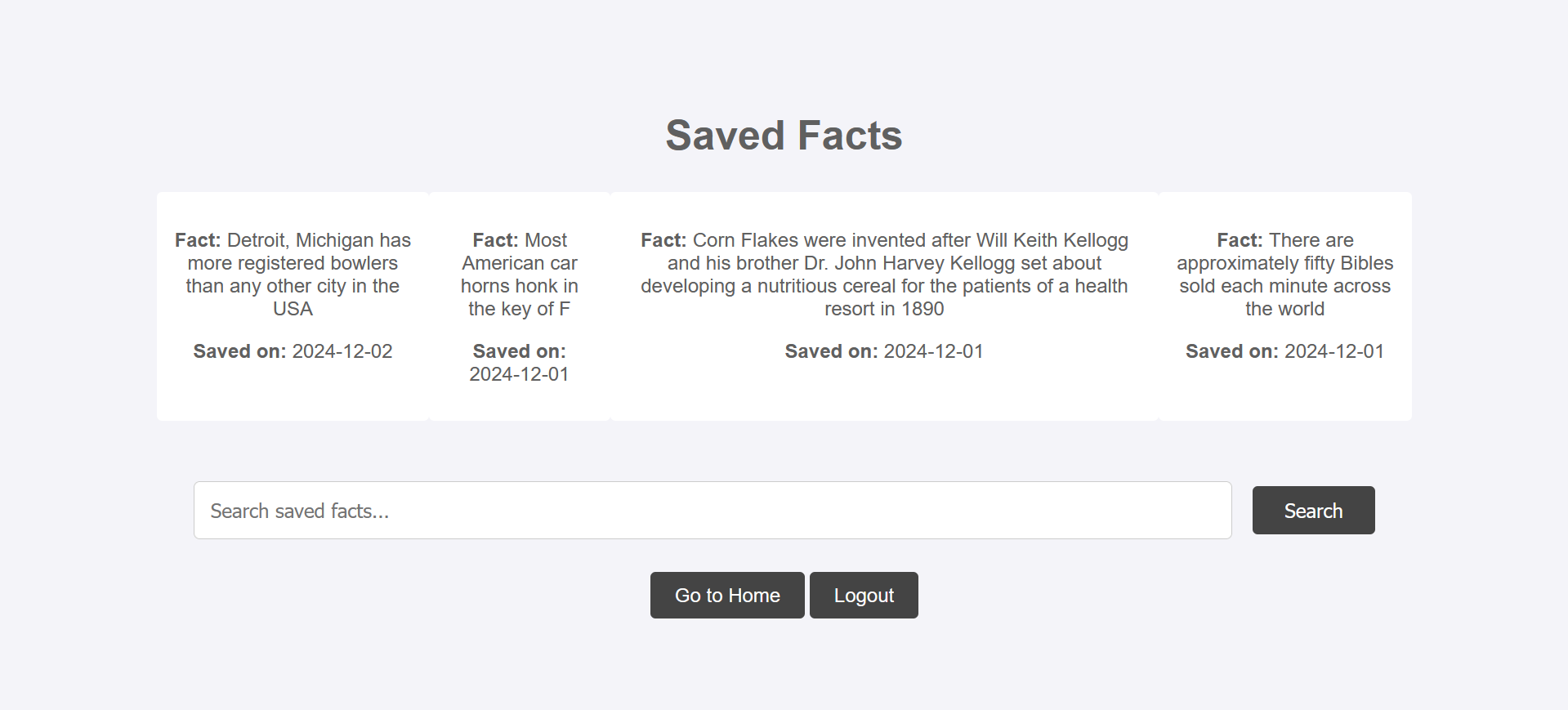
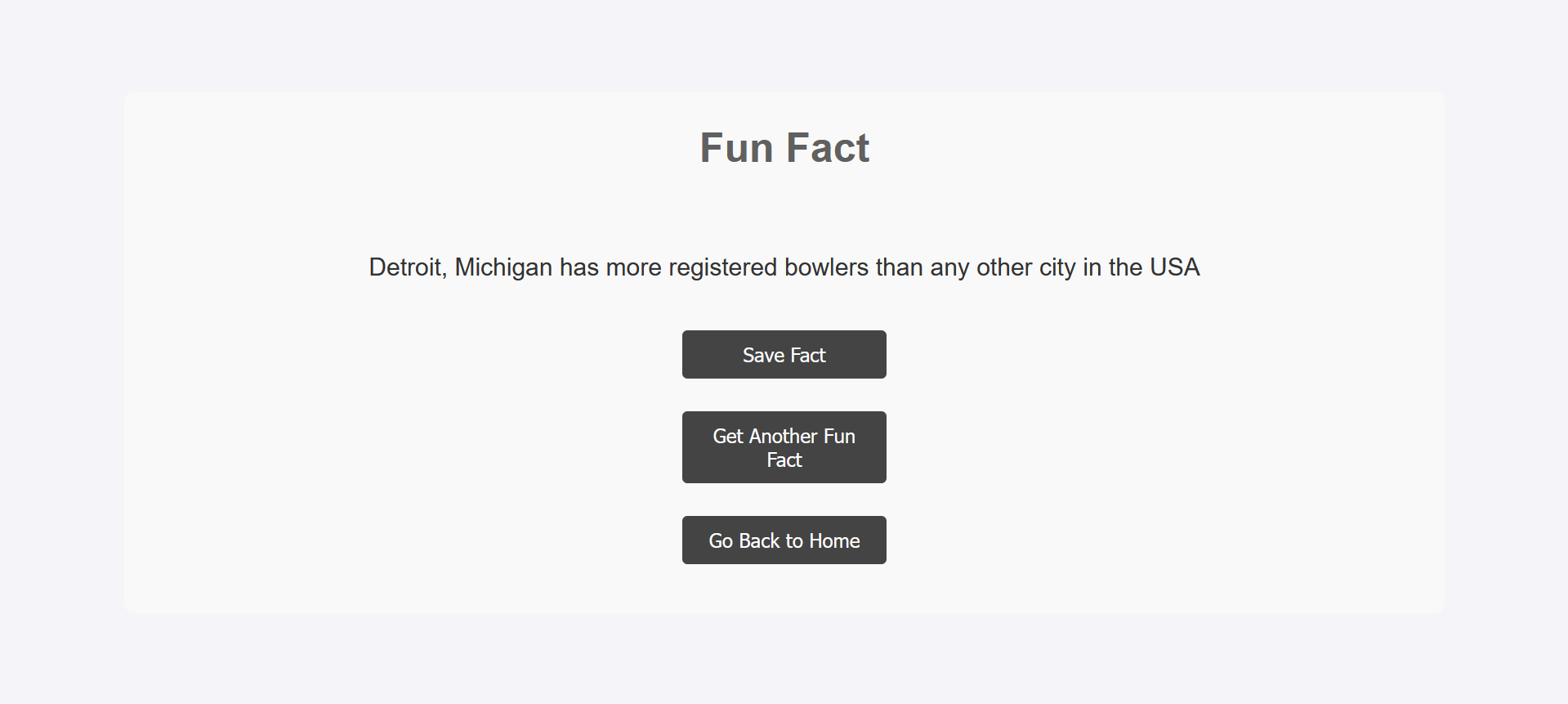
**Data Model**



The database has two tables, Users and Saved\_Facts. The Users table stores user credentials such as the username, email and the hashed password, and the user id. This allows users to log in and create accounts so that they can save facts. The password is hashed so that it is harder for a hacker to steal data even if they have access to the database. The Saved\_Facts table stores the user id, the fact that was saved and the date that the fact was saved at. It also formats the date to a readable format.

**User Functionality**





On the home page, users are greeted with controls to manage their account and view new facts or their saved facts. The home page is the main navigation hub for the user, as it has all the pages linked to it via buttons. Users have the option to create an account, log in, or log out, all from the home page. The about page has a brief overview of the site so users know what the site entails. It also has a button to go back to the home page, in an accessible position. The fun fact page allows users to view facts, reroll facts, save facts and go back to the home page. The saved facts page shows all the facts a user has saved, and allows users to search with key terms for their facts. Searching by specific terms shows all saved facts that contain that word. Logging out also displays a nice screen where users can go back to the home page or log in again. The log in page allows users to log in or click a ‘register here’ button to go the register page. After successful login, the user is taken to the dashboard page. The dashboard page allows users to view their saved facts, go back to the home page, or log out completely. The register page allows users to create an account by taking first and last names, email, password and username and has various sanitisation techniques to improve data handling. It will display a short confirmation message upon registering to ensure they are aware of it. The confirmation message also has a link to the home page so that users can go back and log in with their new credentials, to allow them to start saving facts they like. The whole site uses a intuitive design with colours that contrast in order to enhance readability. This ensures most users can use the web app easily. It is also responsive and caters to all screen sizes. The search function uses partial terms to search for and displays all the saved facts that contain that word or part of that word. The app is customisable as users can create accounts as many times as they like, and save as many facts as they would like. Most pages also use the user’s username to make it feel more familiar and warm to the users. Passwords are hashed using bcryptjs to ensure data security so that users’ can keep their data secure. There are visual effects when hovering over buttons to highlight that they are buttons and can be pressed. I have kept a consistent colour scheme and theme throughout to make the website more aesthetic and easy on the eyes.

**Security**

Security is a major issues in most applications and so I have taken steps to ensure a high level of protection. Passwords are never stored in plain text, so even if the database is compromised they can not be seen. The application uses bcryptjs to hash the passwords before storing them in the database. The hashed passwords are also salted by bcryptjs so the passwords are even more secure making rainbow tables less effective. I have added input sanitisation and validation to ensure user inputs are “clean” before they are sent to the database. All of my sql is used in the backend to ensure harmful inputs are treated as data instead of executable code. The application also uses secure session handling tied to unique session ids stored on the server to manage user authentication. The application also has Cross-Site Scripting (XSS) so saved facts are properly escaped so if there are any malicious scripts embedded in user inputs they are properly escaped before being rendered. This neutralises any threats preventing attacks. There is access controls in place too. Sensitive pages and functionalities can only be accessed by certain authorised users to help mitigate data breaches. For example, the saved facts and dashboard pages are only available to logged in users and users that aren’t logged in will be redirected to the log in page if they try to access it. The application also has error handling in place. It shows user-friendly messages that do not share sensitive data when things go wrong, this is done so that malicious users cannot get any insight into the back end of the system. By using password management, input sanitisation and validation, secure session handling, my application covers the key security issues that most applications face, thus ensuring a protected and safe user experience.

**API Usage**

In my application I use the Fun Fact API provided by API Ninjas to deliver random facts to users. The application uses a GET request to retrieve a random fact by sending it to the API endpoint. The app includes an API key in the code for authentication, as shown below:  


It receives the facts in a JSON format which is not readable for the users. We beautify this by taking by extracting the fact value from the facts to the backend and sending it to the facts.ejs, where it is displayed. There is HTML and CSS that assist in the decoration of the responses so that users can have an easy way of reading the data, and like looking at it too.