Webtech Coursework 2

Hari Kharel – 40312824

Coursework 2 – Extending CW1 to support encoded messaging platform

Contents

Introduction

Overview

Following the requirement of the assignment, this coursework was designed to have a well thought out user interface and robust set of features. This coursework extends on coursework 1 in terms of the design and flow. Much of the previous coursework was written with modularity in mind. To save time on the design phase, much of the HTML, CSS and JavaScript was adapted from coursework 1. The front end is also designed to fit many screen sizes as it automatically resizes elements to fit the screen. In terms of the API, it returns data in JSON format, so any front-end solution can consume the data.

Choice of Features and Functionality

Both the API and the front-end have a host of advanced features and functionalities. The user can send messages to any user registered on the platform, add users to their contact list for easier accessibility. They can also view the messages that they have received as well as the messages they have sent. An email notification is also sent when a user receives a new email. User profile section is available to them to change the data the platform holds about them. Admin users also have an Admin Panel where they can email users, delete accounts, view current platform stats as well as making and removing users as admin.

Here are some of the additional useful functionalities that were not part of the core requirement.

- The addition of the **User Profile Page** enhances user experience as it gives the user control over their information and data the platform holds.
- The platform also features an **Admin Panel**, integrated with the rest of the website and only shown to users with admin Access. This allows proper administration of the platform.
- There are **various security features** in place as well. The messages are only accessible to the first party users meaning Admins and other users are not able to read the messages.
- Sensitive user information, such as passwords, are **hashed and stored**.
- The platform also supports **email notification** for password reset, new messages as well as account creation.

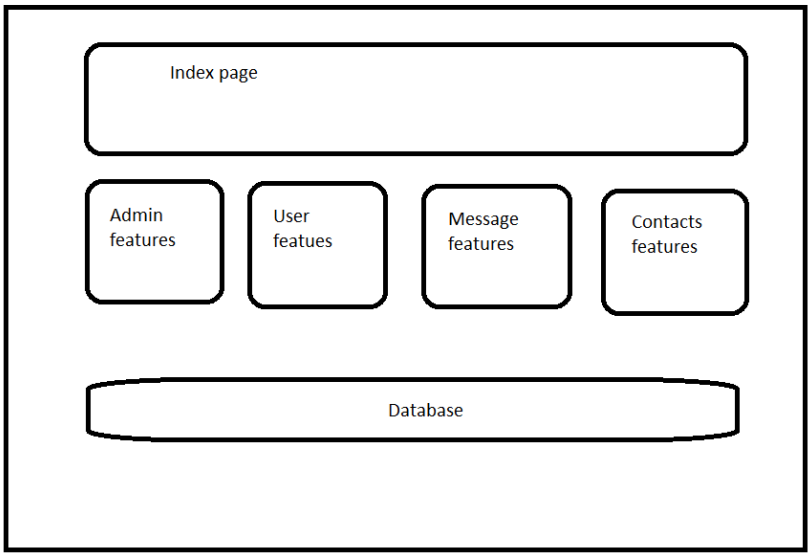
Software Design

API Design

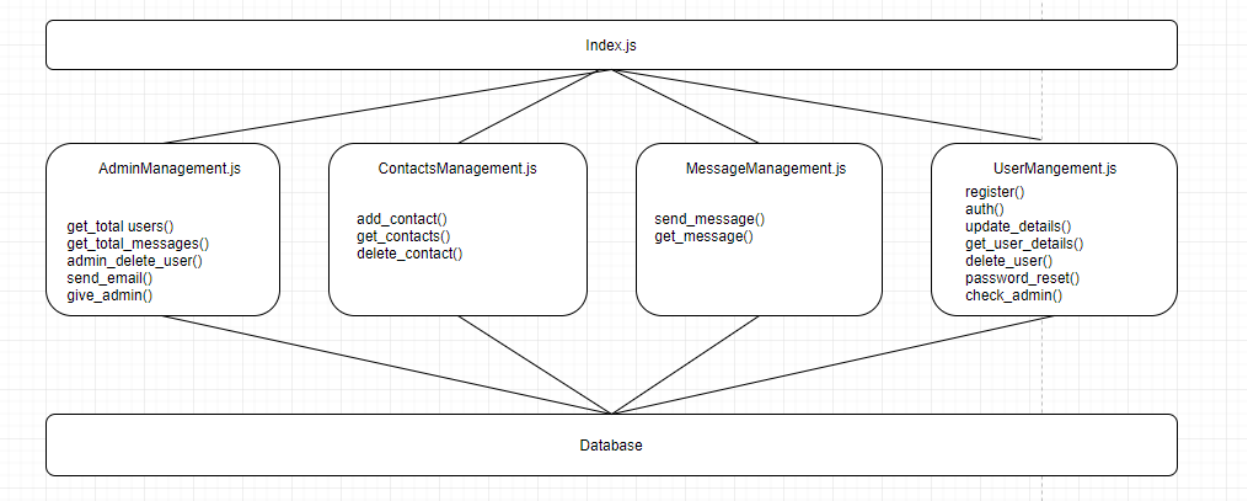
The API was planned out to be a 3 layered-containerized design. This means that there's a top-level, Layer 1 is an access page which handles all the API calls and routes. This layer does not contain very much logic as it passes all the processing to middleware functions in their appropriate layer.

Layer 2 handles all the processing of data. Every method is under its appropriate function. For example, Login, password reset, register, are some of the functions which directly relate to the user, so these are under the UserManagement.js script. Similarly, sending a message, retrieving messages fall under the MessageManagement.js script. This allows for easier planning of the application where the functions are categorized in terms of what they handle.

Finally, the 3rd layer is where the Database is located. This is where all the middleware functions query their data from.



The diagram above illustrates the basic design of the API



The Diagram above illustrates the API design as well as scripts and function designs.

Routes Design

The routes are designed to have a similar structure. Let's take a route that sends an email to the user.

The structure would be " ROUTE/URL, AUTHORIZE , CHECK_ADMIN, SEND_EMAIL "

When the route is called, it will initially check if the passed credentials match a registered user, if it does, it passes control to the 2nd middleware function which then checks if the user issuing the request is an admin and finally, If the user is indeed an admin, it calls the final middleware function which sends the email.

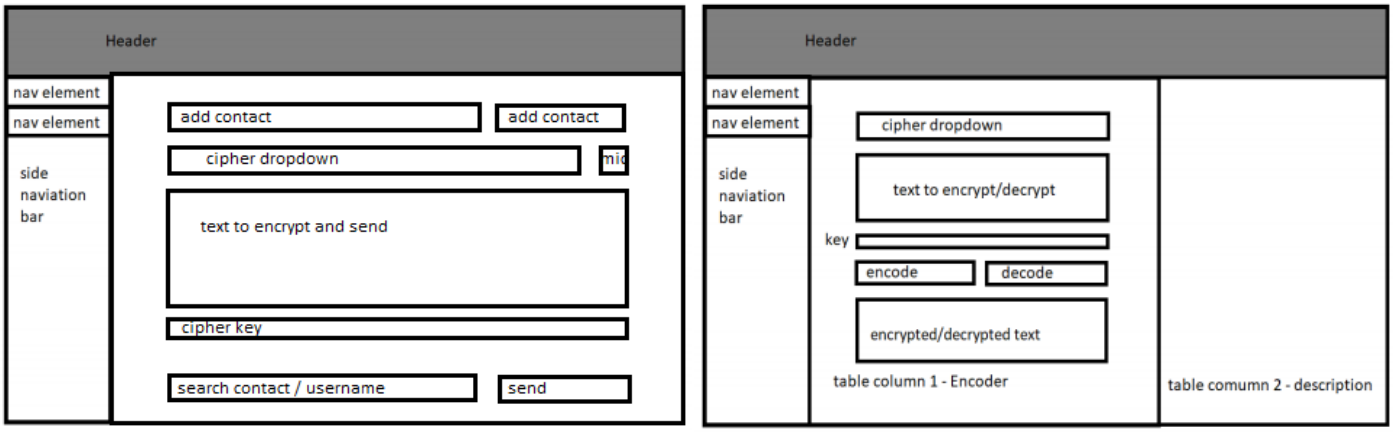
This design simplifies the code as no duplicate code is written to carry out the same task and increases the speed and efficiency of the API. This designed required a well thought out function that handles each request separately. All the initial functionalities were added as separate functions, the design philosophy, code should be open for extension, but closed for modification, was kept in mind while designing the functions.

Front End Design

Layout

Due to the requirement being out for both coursework at the same time, the first coursework was planned and designed with this extension in mind. It was initially developed as a platform that it would grow into. The first coursework implemented design that could be extend by the 2nd coursework. The stylesheet, the ciphers and well as the HTML could be fully reused to save development time as well as provide an appealing design to the user.

The panning ahead of the platform is evident when comparing the cipher page and the messenger page. Most of the components are reused in the messenger page, including the dropdown to select a cipher, text to speech feature, text box and button styles, the JavaScript that powers the site and a lot of other features.



The Diagrams above illustrate the new messenger page (left) which was designed using the reused components of the originally designed cipher page (right)

Logic & Navigation

The API calls are all handled by FetchAPI. Due to the limitation of the API, all the requests were written with only GET and POST request. Other standard HTTP requests are not used in this coursework due to technology stack limitation.

The navigation of the site is handled by a Sidebar that links all the pages together.

Implementation

Due to extensive design of the platform, the implementation went very smoothly with a few design changes here and there. New features, such as hashed passwords and auto generated password using node modules were also implemented.

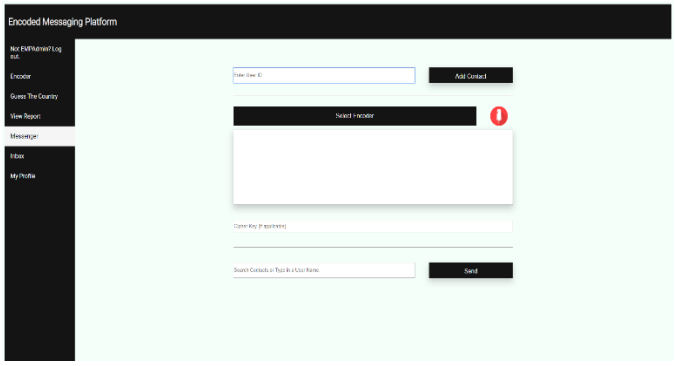
As for the justification of the node modules,

Nodemailer was used as it was one of the modules that could send email notification to the users

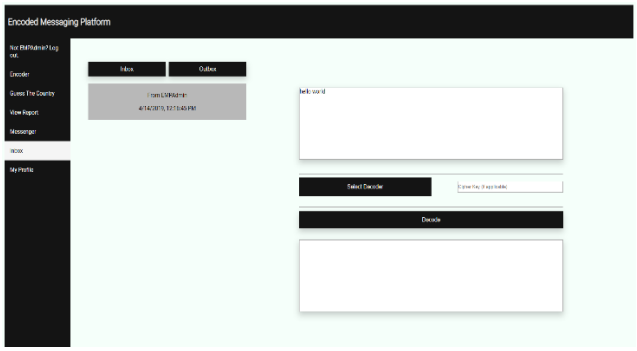
Bcrypt was used to hash passwords using 10 salt rounds to increase strength and security of the password.

Crypto was used to generate a random password when the user needed to reset it.

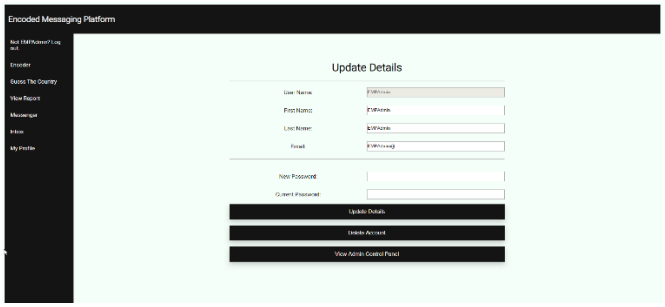
Below are some screenshots of the implemented platform.



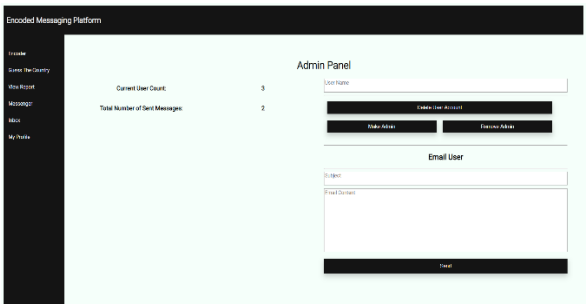
Messenger Page



Inbox/ outbox page



User Profile page



Admin Control Panel

Critical Evaluation

Overall, I believe the finished implementation remains very faithful to the requirements set out. The required features and the additional features use the same visual and coding design, so everything integrates properly. The additional features either enhance the requirements or the user experience and the platform feels ‘natural’ to the user to use and navigate.

Initially, the usage of additional node modules was not allowed and was later changed. By the time the packages were allowed, the platform had already been designed and mostly implemented. This caused the platform to miss out on certain crucial features such as JWT. An extension to this platform would be a mobile app as the API is already designed to work with any front-end solution by using the standard JSON format.

Personal Evaluation

Personally, I feel I have learned a lot by doing this coursework. The opened-ended requirements allowed me to experiment with different ideas and features. This taught me problem solving, designing as well as debugging applications. By designing both the API and the front-end. It gave me a lot of insight into how the web works. This coursework also helped me learn about user experience and what makes a good user interface and software.

References

Mosh, Programming with. “Node.js Tutorial for Beginners: Learn Node in 1 Hour | Mosh.” *YouTube*, YouTube, 21 Feb. 2018, www.youtube.com/watch?v=TlB_eWDSMt4.

Mosh, Programming with. “Express.js Tutorial: Build RESTful APIs with Node and Express | Mosh.” *YouTube*, YouTube, 8 Mar. 2018, www.youtube.com/watch?v=pKd0Rpw7O48.

Mosh, Programming with. “JavaScript Tutorial for Beginners: Learn JavaScript Basics in 1 Hour [2019].” *YouTube*, YouTube, 23 Apr. 2018, www.youtube.com/watch?v=W6NZfCO5SIk.

Express, Express. “Express Documentation.” *Express 4.x - API Reference*, expressjs.com/en/api.html.