Password Vault

By Muhammad Al Ghali

1st Muhammad Al Ghali
Student at University of Michigan-Dearborn
Dearborn, United States of America
Mhdghali@umich.edu

***Abstract*—Due to the recent increase in web transactions and everything migrating to online interactions, a lot of accounts are being created every day, all with different passwords, credit card information, and different identities.**

Keywords— Encryption, Vaults, Password, cards, identities.

I. Introduction

A lot of people these days browse the web, shop online, and complete medical and financial applications. All these services require one of three things. Email and password, Credit Card information, or identity information. In this project, we will be creating a password vault protected by 1 master password. This vault can contain passwords, credit card info, and identities.

II. Software Engineering Method

A. Motivation for method use

Due to time limitations and complete knowledge of the problem we are facing, secure storing of credentials, the requirements are known and don't usually change.

B. The method selected

The method usually used in this case is the Waterfall method, this method usually proceeds in a strict order without overlapping and iterative steps.

Every software engineering method has its step, and for you to achieve the outcome you expect you should follow these steps.

The steps of our method, Waterfall method are six: requirements elicitation, design, implementation, testing, installation, and maintenance and deployment.

C. Requirements

- As a user, I would like to create an account and be able to change my username whenever I wish.
- As a user, I would like to insert website passwords and emails into my vault.
- As a user, I would like to insert credit card information and credentials into my vault.
- As a user, I would like to insert my identity credentials or a family member's identity into my vault.

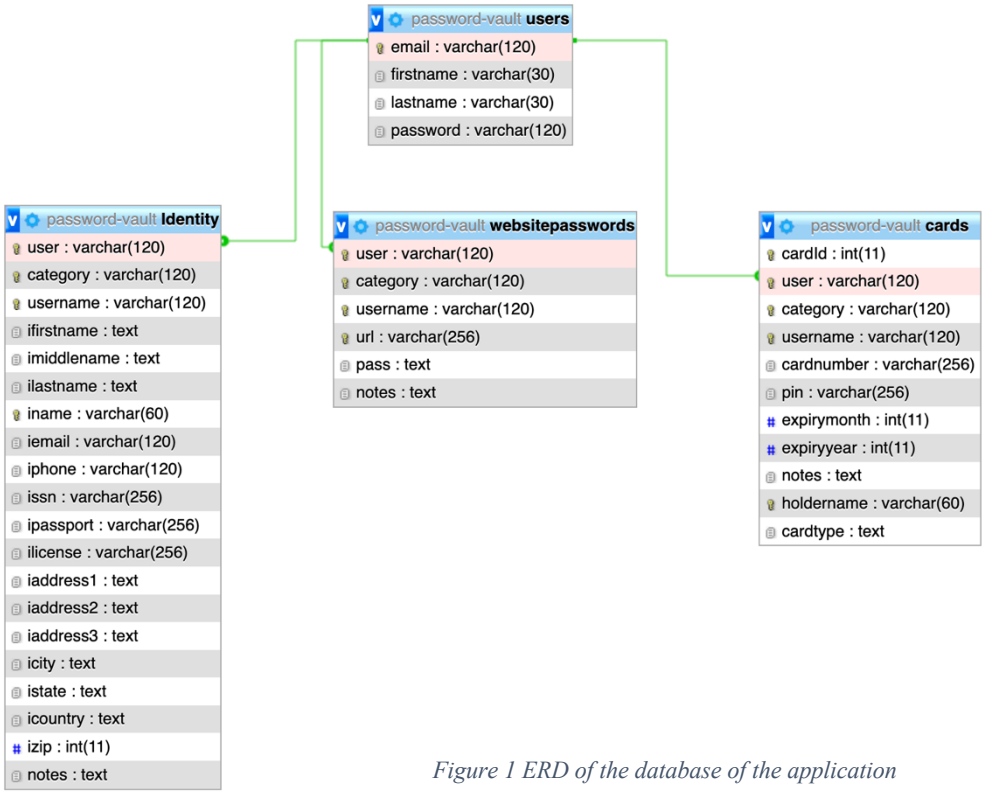


Figure 1 ERD of the database of the application

- As a user, I would like to be able to delete my account whenever I like.

D. Design

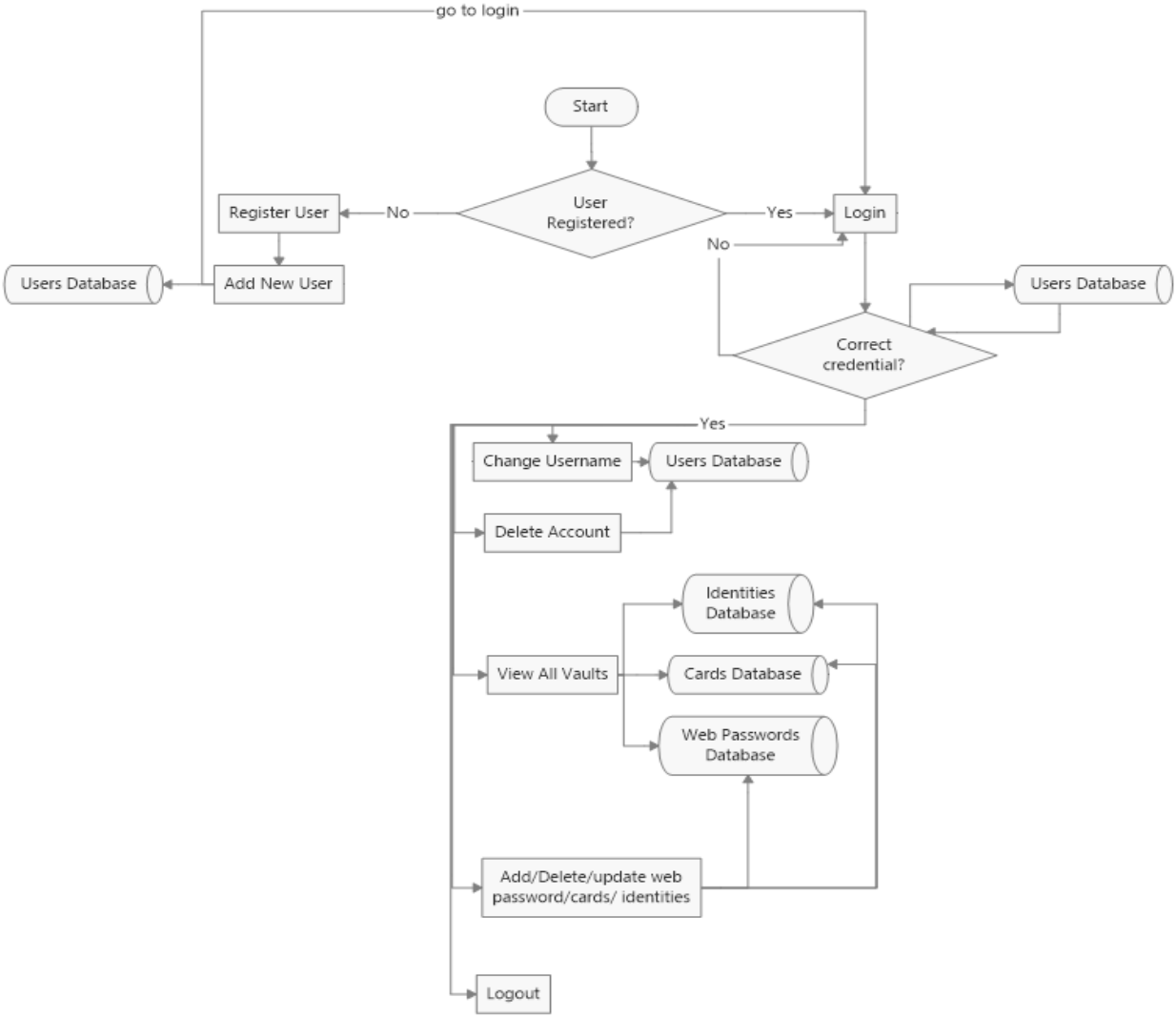
Figure 1 presents the ERD demonstrating the database structure. First, we have the main table the Users table, which has the attributes provided for each user. Secondly, we have the identity table, which has the category of identity, username of choice , first name, middle name, and last name on the identity, Iname which is the full name present on the identity, iemail is the email the user uses as his primary, iphone is the phone number, issn is the social security number, ipassport is the passport number of the person, ilicense is the license number of the person, and the addresses, city, state, country and zip are known, the user can also add specific notes for his identity. Then, we have the website passwords table which contains the category of websitepass, a username which is usually the email used on the password, URL of the website, password, and a specific note for each password. Finally, we have the cards table, which contains the category of card, a card Id unique for each card, username is the name of who's card this is, cardnumber are the usual 16 digits of the card, pin is the password of the card, expiry month and year are the dates of when the card expires, and a user can add a specific note for each card, there is also the holdername which is written on the card, and of course the type of a card be it VISA, AMEX, and so. Notice that each table has an attribute that has the type of VARCHAR (256) these attributes are all encrypted in the database and no one can access them, even the database administrator.

Figure 2 below presents the flowchart of the whole project. The flowchart demonstrates a step-by-step route for every user. There are two routes:

1- First Route: After the application start, if a user is not registered, the user goes to the sign-up page and signs up. After he finishes, he is redirected to the login screen.

2- Second Route: If the user is registered he tries to log in, if the wrong credentials the app asks for the password again, else the user accesses his/her dashboard and there they can change their username and delete their account, they can view all of the items in their vaults and so, and the person can add and delete and update a password of the three categories whenever he/she likes, and of course the logout whenever done.

Figure 3 below shows the class UML of the backend which shows all classes that use the user class, since no user can do any operations on the website unless the session attribute is



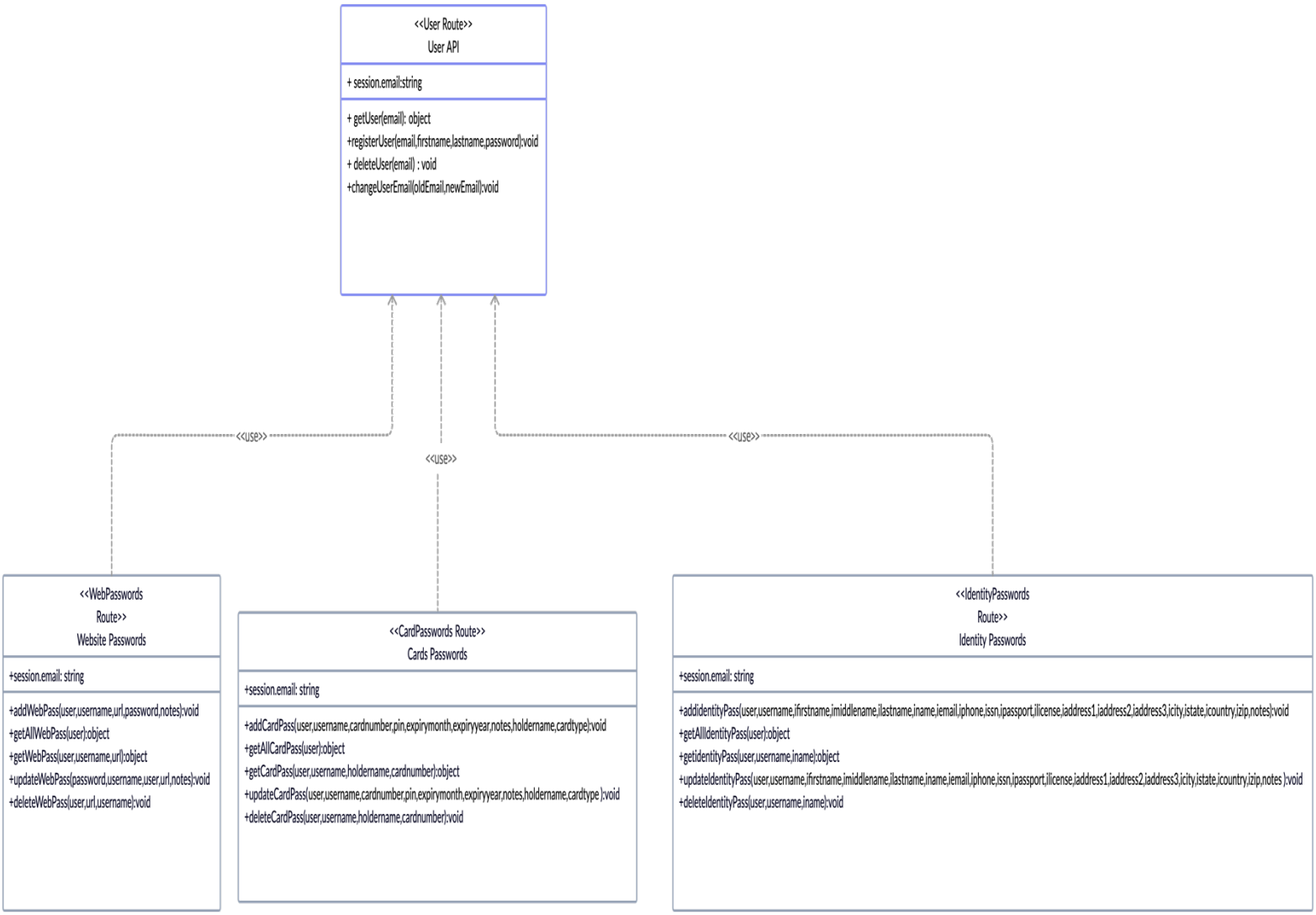


Figure 3 Class UML of the Backend

passed to the child classes since the session contains the information of the user and what is his user email. So, to access any class below the User API all should have a session provided by the user API.

E. Implementation

- The first step in implementation is building the database using MySQL [1].
- The second step was building up the queries needed for the CRUD API operations. Examples: adding a new user, creating a new password, creating an identity, and so.
- The third step after building the queries needed for the data retrieval and sending is building the server and making the APIs using Nodejs [2], and Express.js [3].
- The fourth step was building a user interface with a good user experience. This was built using ReactJs [4] a library made by Facebook, and a CSS library for ReactJs called Rsuitejs [5].
- The last step was implementing the API calls from the frontend. All the calls were made using Axios [6] which is a promise-based HTTP client.
- In the frontend to display a dashboard, after we receive the user information on login, we check the credentials and accordingly the frontend displays the targeted interface.
- In the backend when a user logs in the frontend send a request to get the passwords, cards, and identities, and according to the credentials the user receives his/her three vaults in an organized manner.

F. Testing

- *In the testing of this software, a combination of black-box and white-box testing has been applied in addition to regression testing.*
- *After building each query and API routes, every independent query and route was tested at least once, even logical decisions like checking if the user has privileges to do a certain request or not.*

- *After finishing the full project, every functionality was tested to make sure the software as whole works or not.*
- *As an extra layer of assurance regression testing was used, after adding any feature the software was tested to check If the feature breaks the app or works correctly.*
- *After all these tests many bugs were found and fixed, but many bugs could still be undiscovered.*

G. Installation

1. To install MySQL, you could visit: https://www.apachefriends.org/
2. To run the frontend and backend NodeJS should be installed on your laptop from: https://nodejs.org/en/
3. After downloading the previous two, you could go to the frontend and backend and just type in the command line npm install then npm start, this will install all the packages required in this project and will run the full project.
4. In the backend, some changes should be applied for the express server to connect to the MySQL server.

H. Maintenance and deployment

a) This phase is currently not applicable since the project is not going to be deployed

b) Steps to deploy the project on your device:

1- After installing Nodejs and the MySQL server, launch the xamp manager and turn on the Apache web server and MySQL database.

2- After running the MySQL server get the port number which is usually 3306 and input it in the backend in the databaseManager.js file

3- Then from the code files import the SQL code to the phpMyAdmin which you can access by typing in the browser localhost/phpMyAdmin

4- After making sure that the database is setup run the server by getting into the backend folder and typing in the console npm install then npm run

5- After that get into the frontend file and access the api.js file and put the link that your backend is running on and add to it /api and then in the console type npm install and then type npm run

6- Finally, you're all set up for the project. For any assistance contact: mhdghali@umich.edu

Results

This project is a small step in the security and password management field, but this application could be extended to be a mobile app and could even be supported by multiple platforms due to its simplicity and highly decoupled design.

Conclusion

In conclusion, security is an important aspect and due to the high web access and transactions, we need to have reliable software to store our data safely.

References

[1] Xamp in order to get MySQL: https://www.apachefriends.org/
[2] NodeJS: https://nodejs.org/en/
[3] ExpressJs: https://expressjs.com/
[4] Reactjs : https://reactjs.org/
[5] Rsuitejs : https://rsuitejs.com/
[6] Axios for API calls in frontend: https://axios-http.com/docs/intro