Secure Authentication System Development

Department of Data Science

Information Security (CS3002)

Group Members:

Muhammad Shamil Umar (21i-1786)

Hamiz Ahmed Siddiqui (21i-1678)

Afnaan Asif (21i-1387)

Contents

Chapter 1

Objective

Our primary goal for this project was to design and implement an effective and reliable login mechanism to provide a secure user authentication process. The system uses a username and password as the first layer of authentication, ensuring that only individuals with valid credentials can initiate access.

To improve the security of the application, we were to incorporated a two-factor authentication (2FA) system. The 2FA implementation involves generating a secure One-Time Password (OTP) for each authentication session. The OTP is a unique.

OTPs are also generated and needed for the initial sign up of the email. The system automatically sends the OTP to the user's registered email address. This ensures that only the rightful account owner, who has access to the associated email account, can complete the authentication process.

The integration of this email-based OTP verification as the secondary layer of authentication plays a pivotal role in preventing unauthorized access. It significantly reduces the risk of compromised credentials being exploited, as gaining access to both the login credentials and the associated email account would be required. Overall, this dual-layered approach ensures a high level of user protection, enhancing trust and reliability in the system.

Chapter 2

Methodology

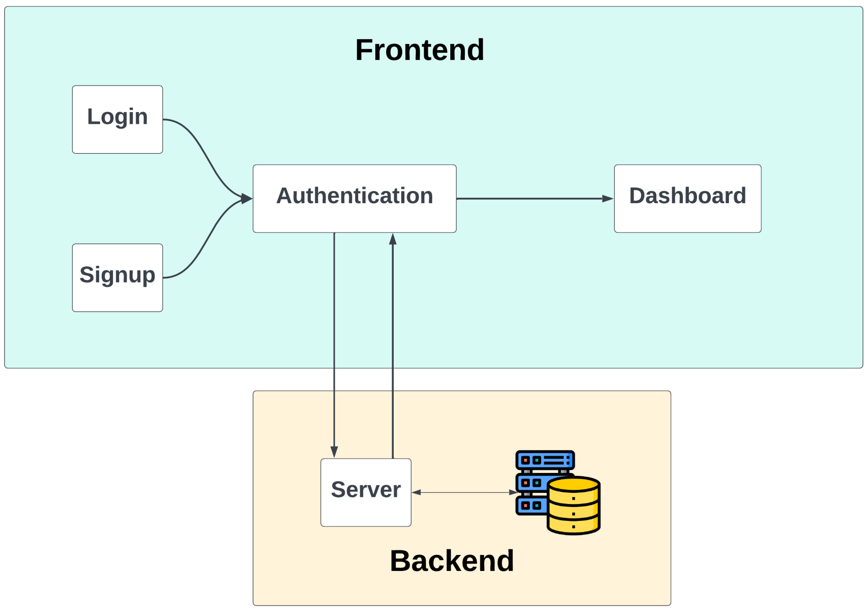

Figure 2.1: Basic Workflow

Fig 2.1 illustrates workflow of application. Our application follows a well structured client-server architecture that incorporates both front-end and back-end modules seamlessly. In the front-end, the user has the option of logging in or registering. Once submit button is pressed, our server send 6-digit OTP to users entered email and redirects user to authentication page where user will enter received OTP. Once successfully verified, the user is redirected to Dashboard page.

Chapter 3

Components

3.1 Front-end

Our front-end is built using REACT.js that is capable enough to provide response user experience. User first lands on login page where user is presented with option of either log in with existing credentials or register himself as new user. Once information is submitted, user is redirected to authentication page where he is required to enter OTP which he received via email. Once he enters valid password, he redirected to dashboard. Otherwise he will be redirected back to login page. Throughout the process, REACT ensures seamless transitions between each page while prioritizing performance and simplicity which leads to user satisfaction.

3.2 Back-end

A Flask Server application is used to handle the backend of the project. We are utilizing a mysql database for the application via mysql.connector. Bcrypt is used to ensure safe hashing of the user provided credentials and smtplib to handle the email OTP generation and transfer.

During registration, when a user submits their email and password, the system generates a 6-digit OTP and sends it to the user's email using SMTP (Gmail). The password is securely hashed using bcrypt before storage. The server temporarily stores the OTP and hashed password in memory (using dictionaries otp_storage and password_storage). When the user submits the OTP for verification, the system checks if it matches and then stores the user's information in the MySQL database. For login, the system verifies the provided email exists in the database and compares the submitted password with the stored hashed password. The server uses CORS (Cross-Origin Resource Sharing) to allow requests from different origins, making it suitable for use with a separate frontend application. Environment variables (stored in a .env file) are used to securely manage sensitive information like database passwords and email credentials.

3.3 OTP Generation & Verification

This functionality is often part of two-factor authentication (2FA) or account creation workflows. By sending an OTP, the application ensures that the email address provided by the user is valid and accessible. A user provides their email during account creation or login. The server generates a unique OTP and invokes this function to send the OTP to the user's email. The user retrieves the OTP from their inbox and inputs it into the application for verification.

Configuration and Setup:

The sender's email credentials (API_EMAIL and API_KEY) are retrieved from a secrets dictionary to keep sensitive information secure. smtp_server and smtp_port specify Gmail's SMTP server and the port for outgoing emails (587 for STARTTLS).

MIMEMultipart is used to structure the email, allowing for attachments or alternative content types. The body is added as plain text using MIMEText.

A connection to Gmail's SMTP server is established using smtplib.SMTP, and the connection is upgraded to secure using starttls(). The script authenticates the sender's email using the credentials (server.login). The email is sent to the recipient's address using server.sendmail, which requires the sender's email, recipient's email, and the formatted message.

3.4 Session Management

Session management is the functionality of allowing a logged in user to maintain that status throughout the time they are logged in and prevent logouts from occurring if the tab is closed or reloaded as does happen in cases where this is not implemented. To handle this, we simply used the localStorage in order to store the details we required to keep the user logged in. The username is propogated from the database upon login, through to the dashboard page where it is displayed. The user may close and reopen the tab and they will still see their name be displayed. When the user does click the logout button on the page, the session storage will simply be deleted and the flags updated. If a user tries to access the dashboard without logins they will simply be denied access and rerouted to the login/sign up page. All this is being handled in the React.js front-end and once the system logs out the user, the process will start again and causes no clashes with the back-end either.

3.5 Contributions

Components	**Major Work**	**Minor Contributions**
Front-end	Shamil	Hamiz
Back-end	Afnaan	Hamiz
OTP Generation & Verification	Afnaan	Shamil
Session Management	Hamiz	Afnaan

Table 3.1: Contributions of each member

In this project, everyone played major role for its development and took complete responsibility for their assigned components as mentioned in table 3.1. Shamil worked on designing and implementing front-end of the application with minor changes made by Hamiz which he did to match its working with his component. Afnaan managed Flask back-end API and took charge of implementing OTP generation and authentication with minor contribution from Shamil and Hamiz. Hamiz focused on implementing session management of this application. This collaborative effort helped us to build this application with ease and with minimal challenges.

Link to repository: https://github.com/HAPPYLAMMA2001/2FA