**.3. USER PORTAL IMPLEMENTATION**

***5.3.1 Signup Page***

In order to construct the user dashboard, a signup form must be created so that people can create accounts. Let's dissect the code to see how it works:

Component Initialization (signup. component.ts): This component uses Angular's FormBuilder to initialise a form group and produce a reactive form.

First and last names, email addresses, phone numbers, passwords, and other user information are put up as form controls.

When the form is submitted, the onSubmit method is called. It validates the form, verifies the password's strength, and sends the form's data to the backend so that users can register.

The strength metre is updated by the onStrengthChange function in response to the password input.

export class SignupComponent implements OnInit {

  form!: FormGroup;

  passwordVisible = false;

  errorAlertVisible = false;

  passwordFieldType = 'password';

  successAlertVisible = false;

  isSubmitting = false;

  constructor(private formBuilder: FormBuilder, private authService: AuthenticationService,private router: Router) { }

ngOnInit(): void {

    this.form = this.formBuilder.group({

      firstName: ['', Validators.required],

      lastName: ['', Validators.required],

      email: ['', [Validators.required, Validators.email]],

      phoneNumber: ['', [Validators.required, Validators.maxLength(10)]],

      password: ['', Validators.required],

      role: ['USER'],

      mfaEnabled: [true]

    });

  }

  onSubmit(): void {

    if (this.form.valid) {

      this.isSubmitting = true;

      this.authService.signup(this.form.value).subscribe(response => {

        this.successAlertVisible = true;

        this.router.navigate(['/googleqr'], { state: { response: response } });

      }, error => {

        this.errorAlertVisible = true;

      }).add(() => {

        this.isSubmitting = false;

      });

    } else {

      this.form.markAllAsTouched();

    }

  }

  onStrengthChange(score: number) {

  }

}

Error Handling: The user is presented with the relevant error messages in the event that the form submission is unsuccessful due to unexpected errors or validation errors.

Incomplete form fields or inadequate password strength are examples of validation problems.

When an unexpected issue occurs, like a server error (HTTP status code 400), a generic error message is displayed.

Backend Communication: User registration and authentication are managed by the AuthService service.

Upon submitting the form, the user's information is transmitted to the backend so that the AuthService's signup method can be used to register the user.

The subscription handles both success and error cases by processing the response from the backend.

The error message that the backend returns if it answers with an HTTP response code 400 (Bad Request), which denotes validation or registration issues.

Spinner Component: A spinner component shows up while the form is being submitted, giving the user visible feedback.

A screen shot of a login form

Description automatically generated

***5.3.2QRCode Generating Page***

Backend Sending QR Code URI: Using a library like qrcode or something similar, the backend creates a QR code URI once the user has finished the signup process.

Usually, this URI is created using user-specific information, like the user's email address or unique ID.

The front end receives this QR code URI from the backend as part of the response.

Frontend Image Generation for QR Codes:

The responseData variable in the Angular frontend holds the URI of the QR code that was received.

With the src attribute set to the QR code URI, the frontend creates a <img> element dynamically using this URI.

The user is presented with the QR code image when this <img> element is rendered in the HTML template.

User Scanning QR Code: Using a smartphone app like Google Authenticator, the user scans the QR code that is shown on the front end.

Based on the shared secret included in the QR code, the mobile app decodes the code and creates a 6-digit validation code.

Verifying Validation Code: The user inputs the 6-digit validation code into the front-end form after scanning the QR code.

The front end requests an HTTP response from the backend to validate the validation code when the user submits it.

The validation code generated using the shared secret is compared with the one received by the backend.

The backend notifies the frontend that the verification was accomplished if the validation code is accurate.

If the validation code is incorrect or verification fails for any reason, the backend sends an error response, and the frontend displays an error message to the user.

The front end moves on to the login page after updating the user interface to let the user know that the verification was successful.

The front end notifies the user of the error and the backend sends an error response if the validation code is invalid or the verification process fails for any other reason.

***5.3.3Login Page***

In my Angular application, the LoginComponent is where users log in. It features a simple form with fields for email and password. Once users enter their credentials and submit the form, I handle the login process. If the form is valid and no submission is already in progress, I prepare the login data and call the login () method from the AuthenticationService. Upon receiving a response, I check if the login was successful and navigate the user accordingly - admins to the admin dashboard and others to the verification page. If there's an error, such as invalid credentials or a server issue, I display an appropriate error message to the user, ensuring a smooth login experience.

***5.3.4Verify page***

After logging in as a regular user, the VerifypageComponent is where the user must authenticate themselves using Google Authenticator. Here, the user enters the 6-digit code generated by the authenticator app. Upon submission, the code is verified against the backend using the verifyCode () method from the AuthenticationService. If the code is correct, the user is redirected to the dashboard. However, if there's an error or the code is invalid, an appropriate message is displayed, ensuring secure access to the user's dashboard.

The addition of the verification step using Google Authenticator in the pension portal enhances security significantly by implementing two-factor authentication (2FA).

With traditional login methods, a user only needs to enter their username and password to access their account. However, passwords alone can be vulnerable to various threats such as brute force attacks, phishing, or password leaks. Even strong passwords can be compromised if they are stolen or guessed.

By introducing 2FA, the pension portal adds an extra layer of security. In addition to something the user knows (their password), it requires something they have (the mobile device with Google Authenticator). This means even if a malicious actor manages to obtain a user's password, they still cannot access the account without also having access to the user's mobile device.

Google Authenticator generates a unique 6-digit code that changes every 30 seconds, making it extremely difficult for attackers to predict or intercept. This time-based one-time password (TOTP) adds an additional level of security against unauthorized access.

Furthermore, the backend verification process ensures that the code entered by the user is valid and matches the code generated by Google Authenticator. If the code is incorrect or has expired, access is denied, providing further protection against unauthorized access attempts.

***5.3.5User Dashboard page***

A screenshot of a computer

Description automatically generated

The user dashboard presents a tailored interface based on the type of pension plan the user has. For users with a defined contribution pension plan, the dashboard features an "Investments" section where users can track various aspects of their investments. Additionally, the dashboard now includes a "Pension Pots" section, providing users with insights into the total amount of money they have in their pension pots.

Within the "Investments" section, users can monitor details such as the investment's name, symbol, purchase date, purchase price, current price, and investment result. The investment result is represented by emoji icons indicating whether the investment is in profit, loss, or neither. Users can also refresh the current price of their investments by clicking the "Refresh" button provided for each investment.

In the "Pension Pots" section, users can view the total amount of money accumulated across all their pension pots. This information offers users a consolidated view of their pension savings, facilitating better financial planning for retirement.

For users with a defined benefit pension plan, the dashboard continues to include a "Defined Benefit Pension Schemes" section. Here, users can monitor details related to their pension schemes, such as final salary, years of service, accrual rate, net pension salary, and retirement goal age.

A screenshot of a computer

Description automatically generated

The NocontributionComponent in the Angular application is designed to cater to users who have not yet initiated contributions to their pension plans. Upon loading the dashboard, users are greeted with a personalized welcome message displaying their first name. The dashboard is structured into sections, each offering valuable information and guidance tailored to users at this stage of their pension journey.

The primary focus of the dashboard is to inform users about the process of getting started with their pension contributions. It presents a series of informative cards, each representing a step or aspect of the pension initiation process. These cards succinctly outline key actions users need to take and provide insights into what to expect during each stage.

For example, the first card prompts users to create their job, emphasizing the critical role of employment status in determining the structure of their pension process. Subsequent cards guide users through steps such as waiting for approval from their employer, initiating contributions to their pension pot, and exploring the option to create multiple pension pots for diversified investments.

Additionally, the dashboard offers educational content on defined contributions and defined benefit pension plans through dedicated sections. These sections provide users with a comprehensive understanding of each plan type, including their features, benefits, and implications for retirement planning.

***5.3.6User-Details Page***

The PersonaldetailsPage in the Angular application serves as a user interface for users to update their personal details and address information. It consists of a form divided into two sections: one for personal details and another for address details.

In the personal details section, users can input their full name, date of birth, gender, nationality, education, and NI number. Upon submission of the form, the data is validated, and if valid, it is sent to the backend server for processing. Success or error messages are displayed to the user accordingly.

Similarly, the address details section allows users to input their address information, including line 1, line 2, city, county, postal code, and country. Upon submission, the address details are also validated and sent to the server for processing, with success or error messages displayed as appropriate.

This page offers a seamless user experience, guiding users through the process of updating their personal and address information while providing real-time feedback on the status of their submissions. It ensures data accuracy and integrity, contributing to an efficient and reliable user management system.

***5.3.7Job Page***

The JobPage in the Angular application serves as a platform for users to add or update their job details. It presents a form where users can input information such as their employee ID, salary, date of job commencement, job title, and the company they work for. The form dynamically populates the company dropdown menu with options fetched from the backend server, allowing users to select their employer from a list of available companies.

Upon submission of the form, the data is validated, and if valid, it is sent to the server for processing. If the user already has job details associated with their account, the page pre-fills the form with their existing job information, enabling them to make necessary updates. Otherwise, the user can fill out the form with their job details and submit them for storage in the system.

The JobPage ensures a smooth user experience, guiding users through the process of adding or updating their job information while providing real-time feedback on the status of their submissions. It plays a crucial role in maintaining accurate user records and facilitating effective communication between users and their respective employers

***5.3.8Contribution Page***

Upon successfully adding job details, a request approval email is automatically dispatched to the employer for review. Upon receiving the request, the employer assesses the provided information and decides whether to approve or reject it. If the request is approved, the user gains permission to contribute to their pension plan. However, if the request is rejected, the user remains unable to make contributions until the approval is granted.

Once the employer approves the request, the ContributionPage becomes accessible to the user. Here, users can input their contribution details, such as the percentage of their salary to be contributed, the start date of contributions, and optionally, an end date. The form dynamically adjusts based on the user's input, ensuring accuracy and completeness of the contribution information.

Upon submission of the contribution form, the data is validated, and if valid, it is sent to the backend server for processing. A successful submission triggers the creation of the contribution record associated with the user's account. However, if any errors occur during the submission process, appropriate error handling mechanisms are in place to provide feedback to the user and ensure data integrity

A screenshot of a computer

Description automatically generated

***5.3.9Pension Pots Page***

Get Pension Pots:

Upon initiating a pension contribution, a main pension pot is automatically created for the user.

This main pot serves as the primary repository for pension contributions, where funds accumulate over time to support the user's retirement savings.

Typically, in a conventional scenario, users contribute a portion of their salary to their pension pot on a regular basis, such as monthly or bi-monthly. However, in our system, we have a unique approach.

Instead of relying solely on user-initiated contributions, the backend periodically injects funds into the main pension pot at regular intervals, such as every 10 minutes.

Following, the component fetches all pension pots associated with the user's account from the backend using the getPensionPotsByUserId method provided by the PensionpotService.

The retrieved pension pots are then displayed in the UI using Angular's data binding mechanisms, dynamically populating each card with essential details such as the pot's name, total amount, and creation date.

Users are also provided with an option to create a new pension pot by entering a unique name in the input field and clicking the "Create Pension Pot" button. This action triggers the createPensionPot method, which sends a request to the backend to create a new pension pot with the specified name and initializes it with a total amount of 0.

Open Modal:

When a user clicks the "Transfer Money" button on a pension pot card, the open method is invoked, which opens a modal window using NgbModal.

The selected pension pot's details are passed to the modal for reference. Users can then select a destination pension pot from a dropdown menu and specify the amount to transfer.

Once the user confirms the transfer, the transferFunds method is triggered to handle the fund transfer operation.

Delete Pension Pot:

Users have the option to delete a pension pot by clicking the "Delete" button on the respective card.

Upon clicking, a confirmation modal is displayed, prompting the user to confirm their action.

If the user confirms the deletion, the deletePensionPot method is called, which sends a request to the backend to delete the specified pension pot.

Transfer Funds:

The transferFunds method facilitates the transfer of funds between pension pots.

Users specify the source pot, destination pot, and the amount to transfer in the modal window.

The method then sends a request to the backend to execute the transfer operation.

If the transfer is successful, the UI is updated to reflect the changes in both the source and destination pension pots.

If the user attempts to transfer an amount greater than the available balance in the source pot, an error message is displayed, preventing the transfer from proceeding.

Get Transfer History:

The getTransferHistory method retrieves the fund transfer history for the user from the backend.

The history includes details such as the source pot, destination pot, transferred amount, and the date of transfer.

The retrieved data is displayed in a table format in the UI, allowing users to track all past fund transfer operations.

The fund transfer history not only includes inter-pension pot transfers but also reflects transfers from other sources, such as profits from stocks or other investments, into pension pots.

This comprehensive history provides users with a complete overview of all financial transactions related to their pension pots, aiding in better financial management and decision-making.

***5.3.10Defined Contribution Page***

Within the realm of pension management, there are defined contribution plans, which typically offer employees more control over their retirement savings. Under this category, there are two types: MegaSaving and Dream Fund.

MegaSaving primarily focuses on investment opportunities related to stocks and cryptocurrencies.

The application retrieves historical performance statistics and real-time stock prices for equities by utilising APIs from providers such as API Ninjas and Alpha Vantage. From a list of companies, users can choose one, evaluate its current price, examine its historical performance going back up to two years, and then determine how many stocks to buy. Depending on how many stocks they want to purchase, a calculator assists consumers in figuring out the overall cost of their investment. But only pension pots worth more than £100 can be invested in; the application makes sure users can only contribute money that is accessible in their pension pots. The tool helps users make wise investing decisions by offering insights regarding notable shifts in stock prices, such as notable increases or decreases.

In a similar vein, MegaSaving interacts with Alpha Vantage to retrieve past performance statistics and real-time exchange rates for cryptocurrencies. From a list of available cryptocurrencies, users can choose one, check the cryptocurrency's current exchange rate, examine its historical performance, and choose how much cryptocurrency to buy. Users can calculate the total cost of their cryptocurrency investment with a comparable calculator feature. Similar to stocks, investments can only be made in pension pots that meet the minimum balance requirement. Users can only invest the money that is accessible in their pension pots thanks to the application. In order to help consumers make wise investment decisions in the cryptocurrency market, MegaSaving also offers insights regarding notable fluctuations in cryptocurrency prices, such as notable increases or decreases. Ultimately, the goal of MegaSaving is to enable customers to choose wisely where to spend their money.

Dream Fund provides thorough market data and research tools in an effort to enable consumers to make wise investment decisions in the FX and commodities markets.

Dream Fund may retrieve historical performance data and real-time exchange rates for forex trading by integrating with APIs such as Alpha Vantage. From a list, users can choose a currency pair, evaluate the current exchange rate, and examine the pair's historical performance. Dream Fund works similarly to MegaSaving in that it limits investment opportunities to pension pots that achieve a certain balance criterion (in this case, £100). After that, users can choose how much money to buy, and a calculator will help them figure out how much their investment would cost overall.

In order to help users make wise investment choices in the foreign exchange market, the application also provides insights regarding noteworthy fluctuations in currency prices, such as notable increases or decreases.

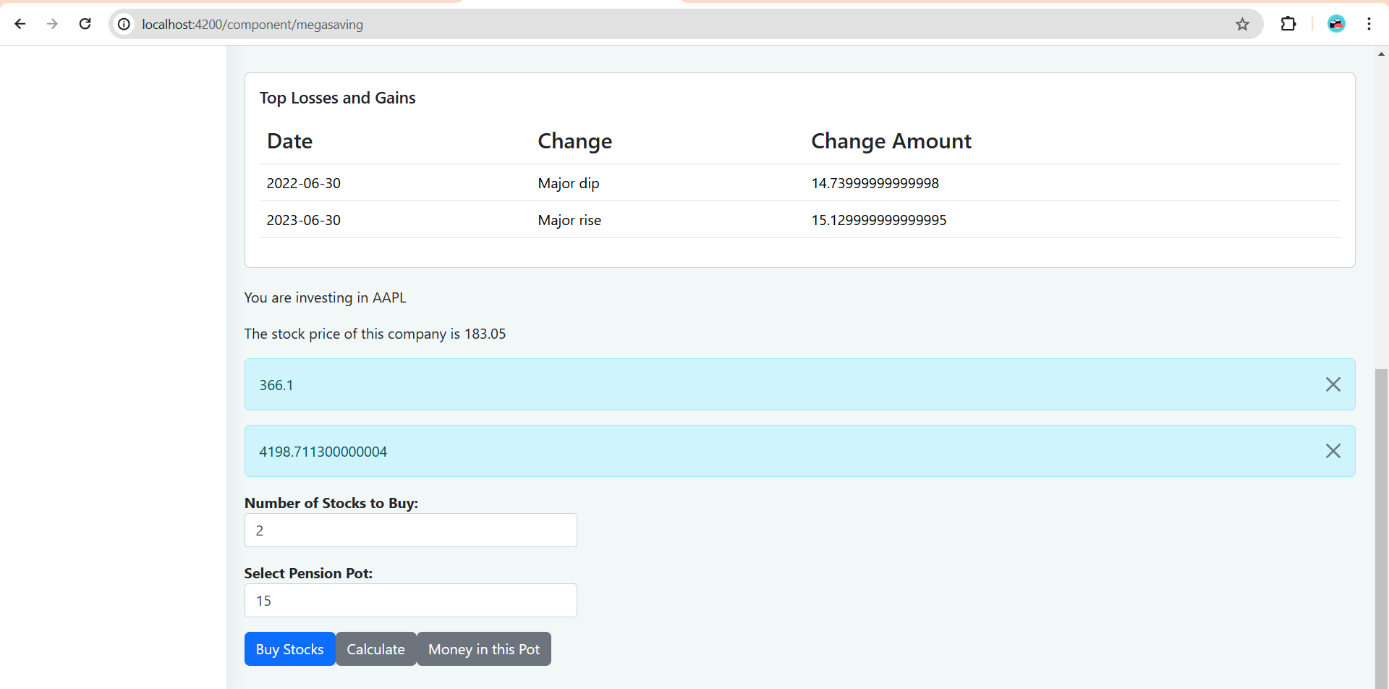
Dream Fund also interfaces with APIs for commodities trading in order to retrieve historical performance data and real-time commodity pricing. From a list, users can choose a commodity, view its current price per unit (such troy ounce), and examine its historical performance. The same as with forex trading, investments can only be made in pension pots that meet the minimum balance criteria. After that, users can choose how many commodities to buy, and a calculator will help them figure out how much their investment will cost overall. Additionally, the tool offers information regarding significant shifts in commodity pricing.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated



***5.3.11My Investments:***

The "My Investments" page in the Angular application serves as a comprehensive dashboard for users to monitor and manage their investment portfolio within a pension management system. Organized in an accordion-style layout, each investment card provides detailed information such as investment type, symbol, quantity, purchase price, purchase date, current price, and investment status. Users can expand individual cards to view these details and, if applicable, transfer profits to their pension pots with a single click. Real-time price updates for stocks, cryptocurrencies, forex, and commodities are seamlessly integrated, enhancing user experience and ensuring accurate investment tracking. Additionally, the presence of a loading indicator during data fetching and processing maintains transparency and responsiveness. Leveraging Angular's features like event binding and HTTP client, the "My Investments" page offers users a user-friendly interface to make informed decisions and effectively manage their pension investments.

***5.3.12Compliance:***

The compliance page in the investment system serves as a critical component for monitoring regulatory adherence and risk management. Predefined compliance triggers are integrated, with varying degrees of severity, like "spend 10000 in 1 day," "5000 loss in 5 days," and "1000 loss in 1 day." These triggers are intended to identify investment behaviours that may be dangerous or noncompliant, allowing for early intervention to reduce related risks. The solution provides stakeholders with timely insights into investor actions and compliance status by generating alerts and notifications upon trigger. The compliance page also includes a tabular display of compliance data, which includes timestamps, actions that were triggered, reasons, notification dates, email status, and responses.

In addition to responding to warnings and managing compliance events, users can access tutorials for more regulatory compliance information. All things considered, the compliance page is an all-inclusive instrument for guaranteeing regulatory compliance, controlling investment risks, and encouraging investor protection in the investment ecosystem. The compliance trigger "1000 loss in 1 day" indicates that a compliance alert will be triggered if an investor loses £1000 or more in a single day. This trigger indicates a moderate severity level, meaning that although it needs to be addressed, there may not be an imminent high-risk situation. Real-time monitoring of investor transactions and portfolio activity by the system allows it to detect situations in which losses reach or above the predetermined level. The system triggers and then starts predetermined actions, such alerting stakeholders.

A screenshot of a computer

Description automatically generated

***5.3.13Defined Benefit Page:***

**User Input - Retirement Goal Age:**

The user enters their desired retirement goal age through the form field provided on the page.

**Fetching User Salary:**

Upon page initialization, the component fetches the user's salary using the getSalary () method. This information is crucial for calculating taxes and determining pension benefits.

The getSalary () method sends a request to retrieve the user's salary data from the backend server.

Based on the retrieved salary, the component calculates the applicable tax rate using the calculateTax () method. Tax rates are determined based on predefined thresholds of the user's salary.

**Calculating GDP Growth Rate:**

The component also fetches the GDP growth rate, which impacts the defined benefit accrual rate. This is done through the fetchGDPGrowthRate () method.

The method sends an HTTP GET request to an external API endpoint to retrieve the GDP growth rate for the United Kingdom.

Upon receiving the GDP growth rate data, it calculates the adjusted accrual rate using the calculateAdjustedAccrualRate () method. This adjusted accrual rate takes into account the GDP growth rate and adjusts the base accrual rate accordingly.

**Fetching User Age:**

Another important factor in pension calculations is the user's age. The getpersonaldetails () method retrieves the user's age from their personal details stored in the backend database.

It calculates the user's age based on their date of birth and the current date.

**Submitting Form Data:**

When the user submits the form with their retirement goal age, the onSubmit () method is triggered.

The method first checks if the user's personal details are available (personalid), and if the form data is valid.

It then validates whether the entered retirement goal age is reasonable based on the user's current age (age) with a buffer of 10 years.

If the conditions are met, it constructs a data object (formData) containing the retirement goal age and the adjusted accrual rate.

Depending on whether the user already has a defined benefit pension scheme (definedbenifit.id), it either updates or creates a new pension scheme using the

**DefinedBenefitService.**

Success or error messages are displayed to the user based on the outcome of the API requests.

The backend makes computations based on the user's submitted retirement goal age and other pertinent data, like the user's pay, years of service, and GDP growth rate. The backend provides the results, including the final salary, years of service, accrual rate, net pension salary, and retirement goal age, after the computations are finished.

After that, a structured table format containing this data is dynamically shown on the page to give visitors a clear overview of their defined benefit pension plans. The final salary, years of service, accrual rate, net pension salary, and retirement goal age are all listed in columns of the table.

***5.3.14How to Invest Page:***

This Angular component represents a tutorial page designed to help users invest effectively in various financial instruments such as stocks, cryptocurrencies, commodities, and forex. Here's a breakdown of its structure and functionality:

Heading Container: This section contains a heading with the title "Tutorial to Help You Invest Effectively".

Stock Investing Tutorial: This section provides a tutorial for stock investing. It includes a slider-container component powered by ng-image-slider Angular library. The stockimages array contains objects with image URLs, thumbnail URLs, alt text, and titles for each slide in the slider.

Crypto Investing Tutorial: Similar to the stock investing section, this section provides a tutorial for crypto investing. It also includes a slider container with images specific to cryptocurrency investment.

Forex Investing Tutorial: This section offers a tutorial for forex (foreign exchange) investing. It includes a slider container with images relevant to forex trading.

Commodity Investing Tutorial: This section presents a tutorial for commodity investing. It includes a slider container with images specific to commodity trading.

Each tutorial section follows a similar structure but contains content tailored to the respective investment type. The images within each slider provide visual aids to enhance the learning experience for users interested in investing in these financial markets.

The component class HowtoinvestComponent defines arrays (stockimages, cryptoimages, foreximages, and comimages) that store image data for each tutorial section. These arrays contain objects with image URLs, thumbnail URLs, alt text, and titles. These objects are then utilized to populate the sliders in the HTML template.

A screenshot of a computer

Description automatically generated

***5.3.15Help Page:***

In this Help page, users and admins can engage in conversations with each other. Here's a breakdown of how it works:

Sending Messages:

Users can type their message in the text area provided.

Messages are limited to 120 words to keep them concise.

When the user clicks the "Send" button, the message is sent to the admin.

The message is then displayed in the "Previous Conversations" section.

Previous Conversations:

Users can view their past conversations with the admin.

Messages are automatically fetched from the database and displayed here.

Each message includes information about the sender, receiver, and message content.

A note informs users that messages are automatically deleted from the database after 15 minutes for database optimization.

Downloading Conversations:

Users have the option to download their previous conversations if they consider them important.

A button labeled "Download Previous Conversation" allows users to download the entire conversation history.

Implementation Details:

The page fetches previous conversations and user details from the database upon initialization.

Users can send messages, which are then added to the database using the sendMessage () method.

Previous conversations are displayed using \*ngFor loop to iterate through the messages array.

Users' messages are associated with their IDs, which are used to fetch their details (such as name) from the users' map.

**5.4 Admin Dashboard**

Overview of the Admin Dashboard

Functionality: The admin dashboard acts as a single location for all system administrative activities.

Administrators can effectively manage people and corporations with a variety of capabilities at their disposal.

Viewing User Details: Administrators have access to a list of all enrolled users in the user management area.

Every user entry includes the following necessary information: phone number, email address, first and last name.

Admins may easily retrieve user information with the help of this user list, which gives them an overview of the user base.

Adding a Company: System administrators can also include new businesses in the system.

The platform can grow by adding more organisations for job postings and user interaction thanks to this capability.

***5.4.1View User:***

Overview of View User Page

Purpose:

The View User page provides admins with detailed insights into individual user profiles.

User Information Display:

Admins can view various pieces of information related to the user, categorized into different sections based on the type of company the user belongs to.

Company Type Section:

This section displays the type of company associated with the user.

Admins can quickly identify whether the user is part of a defined benefit or defined contribution company.

Pension Pot Section:

For users associated with defined benefit companies, details about their pension schemes are provided.

This section offers admins a comprehensive overview of the user's pension scheme details.

Defined Benefit Pension Schemes Section:

For users associated with defined benefit pension schemes, this section provides detailed information about their pension plans.

Admins can gain insights into various aspects of the user's defined benefit pension scheme, including final salary, years of service, accrual rate, net pension salary, and retirement goal age.

Investments Section:

Users belonging to defined contribution companies have their investments showcased in this section.

Admins can see the investments made by the user, including details such as the name, symbol, purchase date, purchase price, current price, and investment result.

This section enables admins to understand the user's investment portfolio and track investment performance.

Request Approval Status:

The page also indicates the approval status of any requests made by the user.

Admins can quickly determine whether a request has been approved by the company or not.

Interaction Options:

Admins have the option to message the user directly from this page, facilitating communication between admin and user

***A screenshot of a computer

Description automatically generated***

***A screenshot of a computer

Description automatically generated***