

## 1. Introduction

### 1.1 Overview

The News Aggregator is a full-stack web application designed to provide users with a streamlined and personalized news reading experience. By aggregating articles from various sources, the platform aims to offer a comprehensive view of the latest news while allowing users to tailor their content preferences.

### 1.2 Key Features:

#### 1. User Authentication and Registration:

Users can create accounts and log in to save user detail.

#### 2. Article Search and Filtering:

Robust search functionality with filtering options by date, category, and source.

#### 3. Personalized News Feed:

Customization news feeds based on user-selected sources, categories, and authors. User can change their preference from edit Profile. Give toggle button option on news-feed to on and off user preference data.

#### 4. Mobile-Responsive Design:

Optimized for viewing on a range of devices, ensuring a seamless user experience.

#### 5. Data Sources:

The application integrates data from reputable news APIs, including NewsAPI, The Guardian, New York Times. Scheduled scraping commands fetch and store data locally, ensuring efficient filtering without relying on live data sources.

## 1.3 Technologies Used:

### Backend:

Language: Laravel

Version: 10.10

### Frontend:

Language: React.js

Version: 18.2

**Containerization:** Docker

**APIs:** NewsAPI, The Guardian, New York Times

## 2. System Architecture

### 2.1 High-Level Architecture

The News Aggregator follows a client-server architecture, where the client (Frontend) interacts with the server (Backend) to retrieve and display news articles. Additionally, Docker is employed for containerization, ensuring consistency across various environments.

#### Client (Frontend):

- Developed using React.js for a dynamic and responsive user interface.
- Communicates with the Backend to fetch and display news articles.
- Implements user authentication and preferences.

#### Server (Backend):

- Built with Laravel, a robust PHP framework.
- Manages user authentication, registration, and personalized settings.
- Integrates with various news APIs for data retrieval.
- Implements scheduled commands for scraping and storing data locally.
- Exposes RESTful APIs consumed by the Frontend.

## **Database:**

- Stores user information, preferences, and locally scraped news data.
- Utilizes relational tables for efficient data retrieval and management.
- Make migration for create table

## **Docker Containers:**

- Dockerized environments for both Backend and Frontend.
- Enables easy deployment and scaling across different environments.

## **2.2 Backend (Laravel)**

The Backend is responsible for handling user-related functionalities, interacting with external APIs, and managing the local database.

### **User Management:**

- Registration and authentication of users.
- Storage of user preferences and settings.

### **API Integration:**

- Utilizes various news APIs for fetching articles.
- Implements mechanisms for search and filtering.
- Implements functionality for edit user data and user preference.
- Develop logout functionality.

### **Data Scraping:**

- Scheduled commands for scraping data from external APIs.
- Local storage of scraped data for improved performance.

### **API Endpoints:**

- RESTful API endpoints for communication with the Frontend.
- Handles requests for user data, articles, and preferences.

## **2.3 Frontend (React.js)**

The Frontend is responsible for providing an intuitive and interactive user interface for interacting with news articles and managing user preferences.

## **User Interface:**

- Dynamic and responsive UI using React.js components.
- User-friendly interfaces for article display, search, and filtering.

## **User Authentication:**

- Implements user registration and login features.

## **Personalization:**

- Allows users to customize their news feed based on preferences.

## **Communication with Backend:**

- Makes API calls to the Backend for fetching and updating data.

## **2.4 Interaction between Backend and Frontend**

The Backend and Frontend communicate through RESTful API calls. The Frontend sends requests for user data, articles, and preferences, and the Backend responds with the required information. This decoupled architecture enables scalability and flexibility.

## **2.5 Database Schema**

The database schema includes tables for users, user preferences, and locally stored news data. Relationships between tables ensure efficient data retrieval.

### **users Table:**

- Stores user credentials, including username and password.

### **user\_authors Table, user\_categories Table, user\_sources Table**

- Saves user preferences such as selected sources, categories, and authors.

### **articles Table:**

- Contains locally scraped news data with fields like title, category, source, and publication date.

## 3. Installation

### 3.1 Prerequisites

Before setting up the News Aggregator, ensure that the following prerequisites are met:

- **Node.js:** Install Node.js from <https://nodejs.org/>.
- **npm:** Node Package Manager (npm) is typically installed with Node.js.
- **Composer:** Install Composer, the PHP dependency manager, from <https://getcomposer.org/>.
- **Docker:** Install Docker from <https://www.docker.com/get-started>.
- **Git:** Install Git

### 3.2 Getting the Source Code

Clone the News Aggregator repository from the GitHub repository:

`git clone https://github.com/bansifaldu/news-app.git`

Navigate to the project directory:

#### **Backend (Laravel):**

Configure the database settings in the .env file:

Create passport key

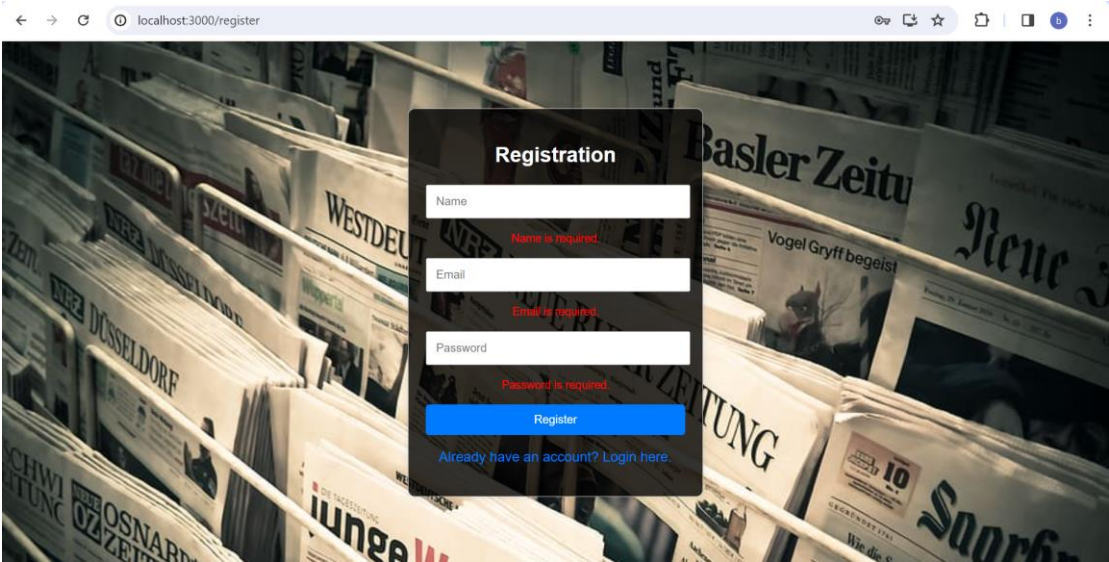
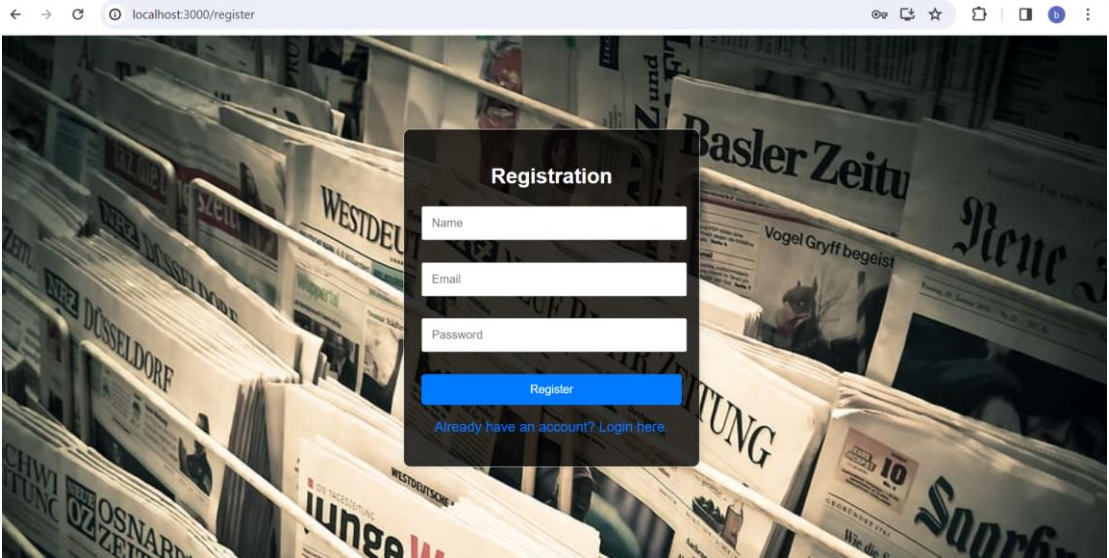
#### **Running the Application**

Once the Docker containers are up and running, access the News Aggregator application at <http://localhost:3000> for the frontend and <http://localhost:8000> for the backend.

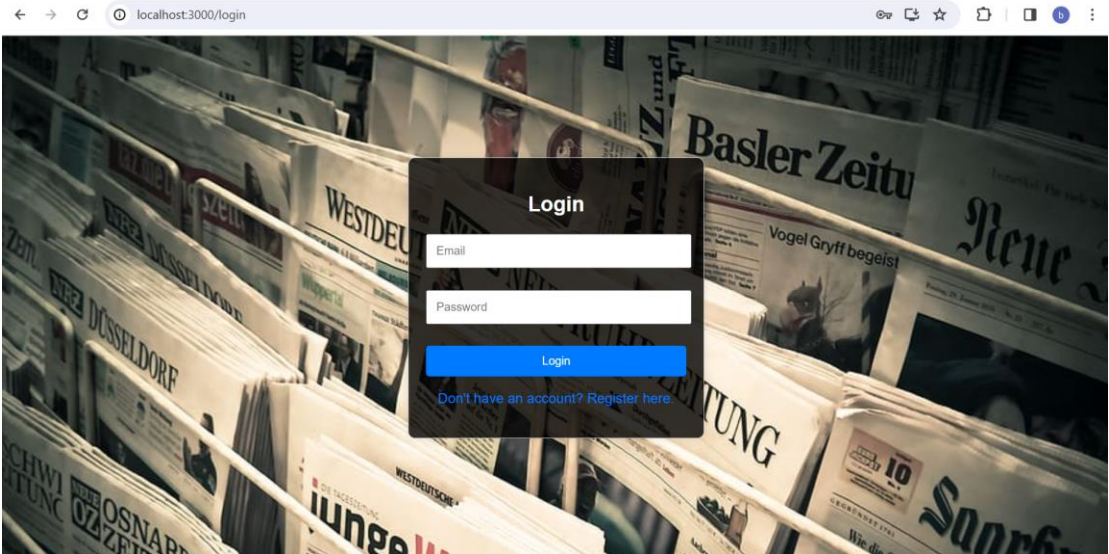
## 4. User Authentication and Registration

### **Screenshots of the Registration Process**

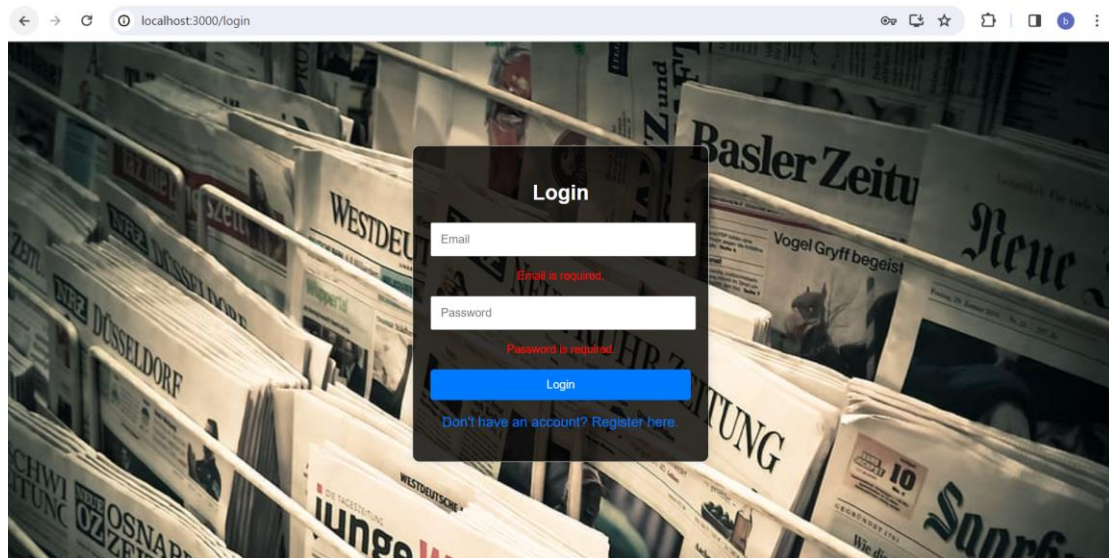
# News App Documentation



## Screenshots of the Login Process







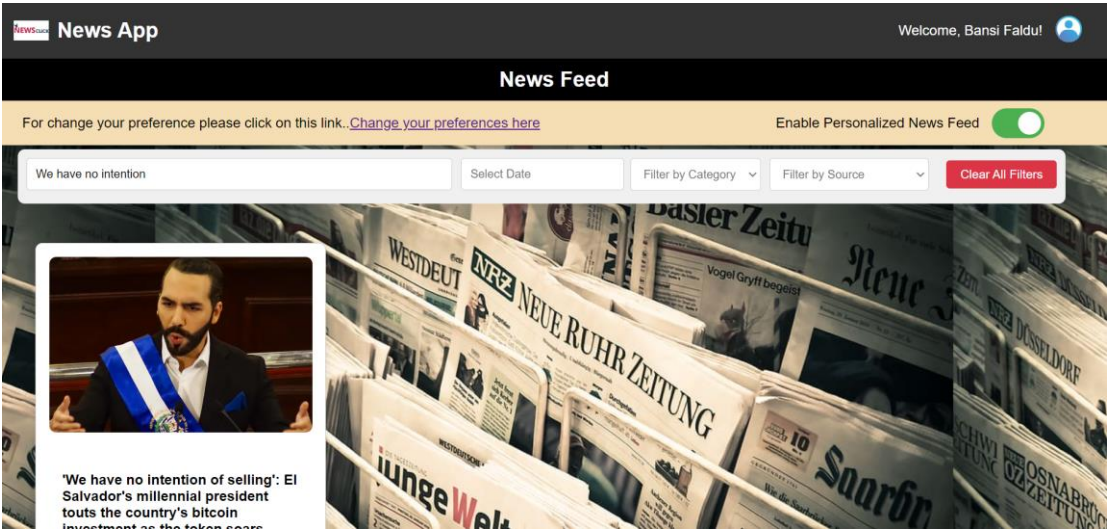
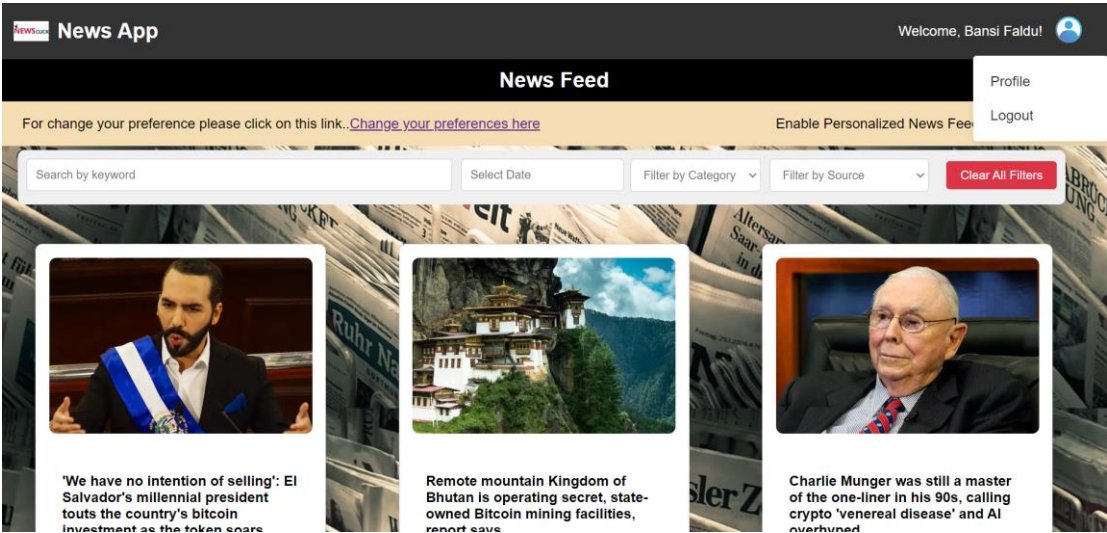
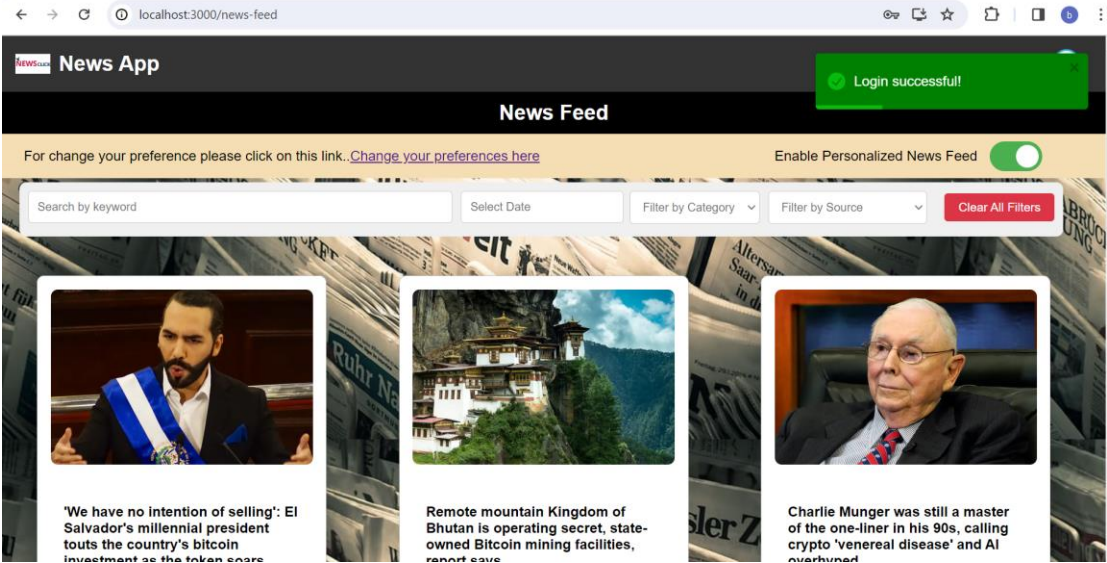
## 5. Personalized News Feed With Search And Filter

**In news-feed page:**

1. Implement Lazy loading
2. Implement Toster notification
3. Implement User preference toggle
4. Implement Profile dropdown in Header with Logit and Edit profile option
5. Filter for Date, Category, Sources and Clear Filter
6. Search by keyword
7. Implement Loader

# News App Documentation

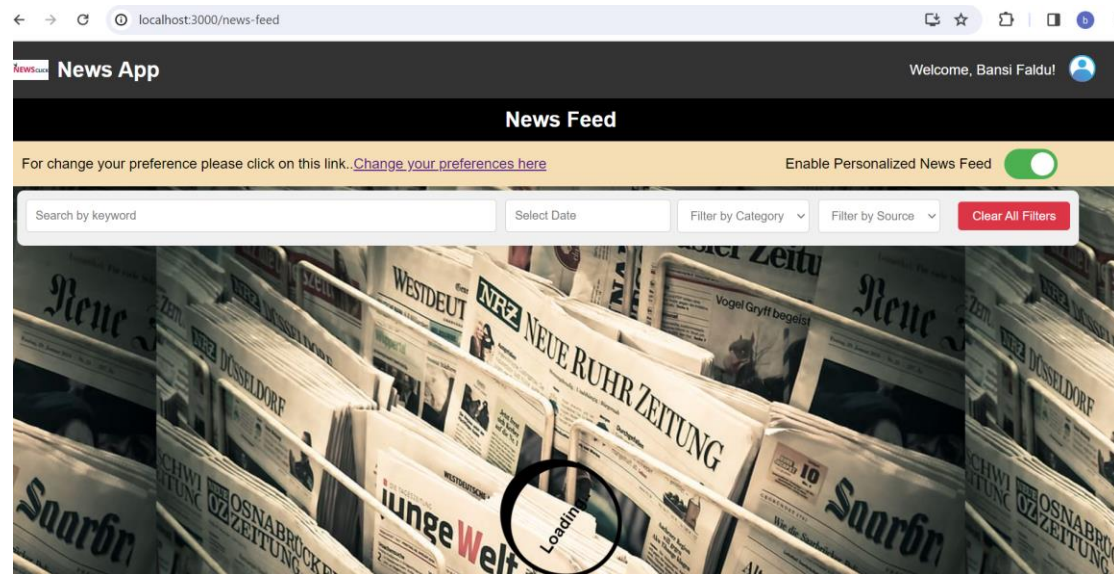
8



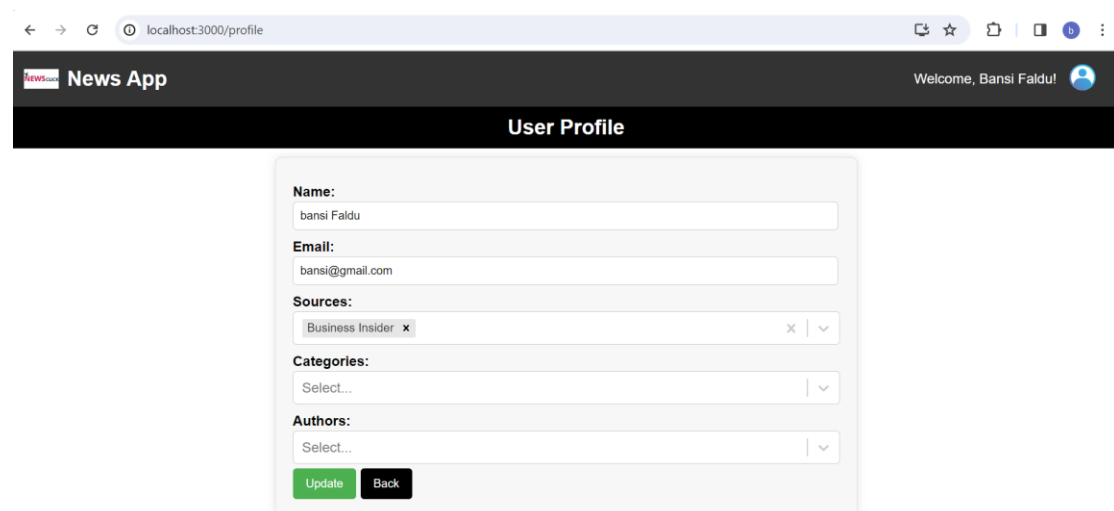


# News App Documentation

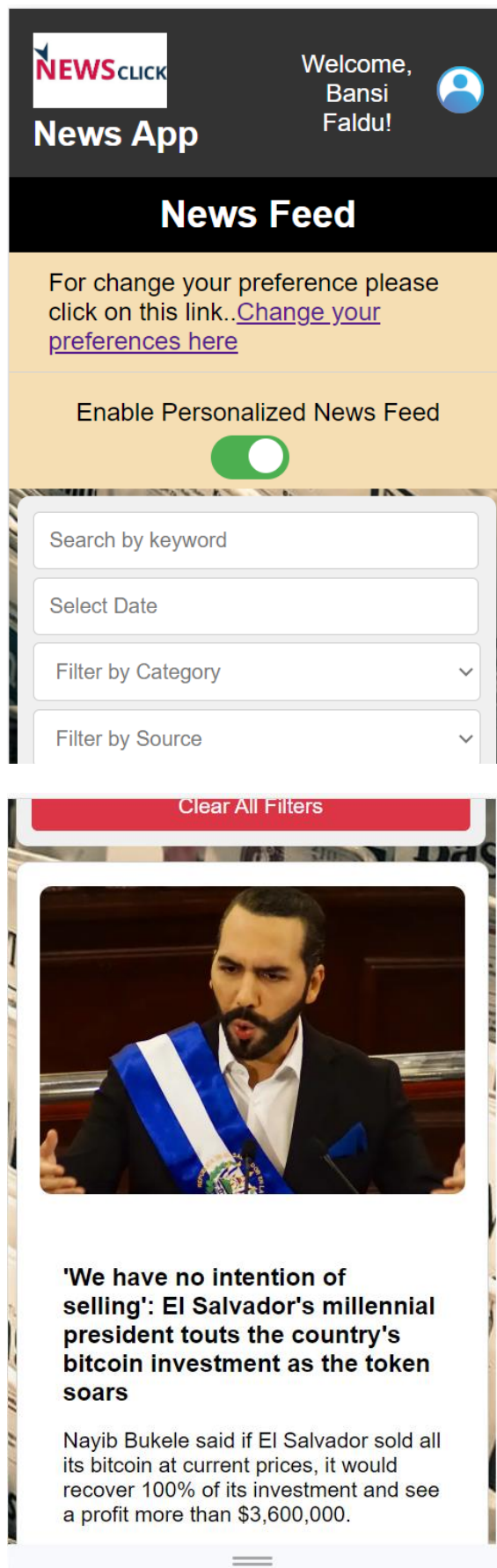
9

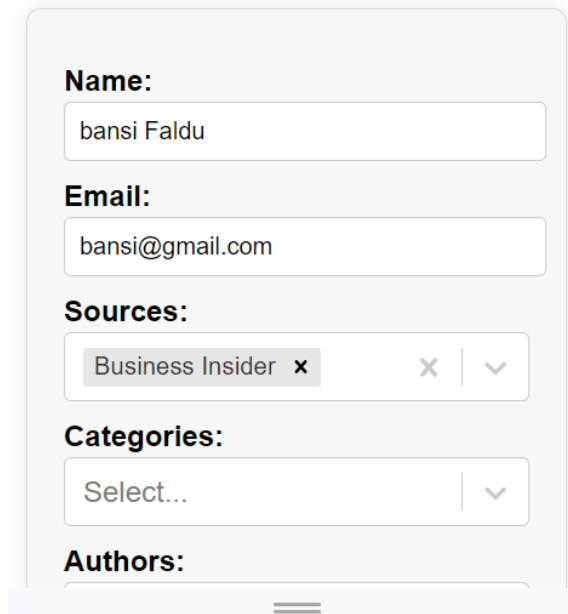
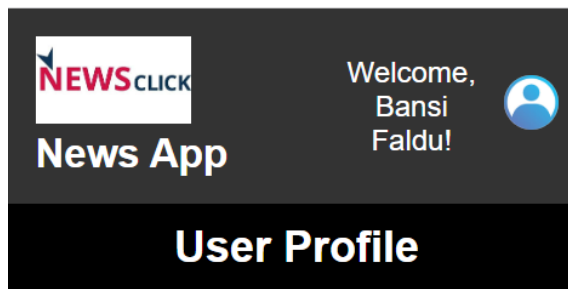


## 6. Profile Edit with Preference

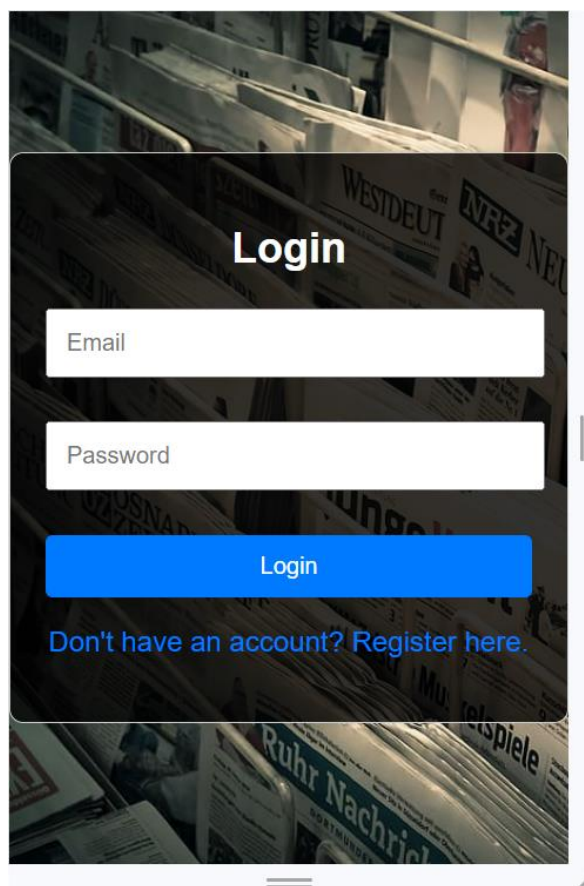


## 7. Mobile-Responsive Design

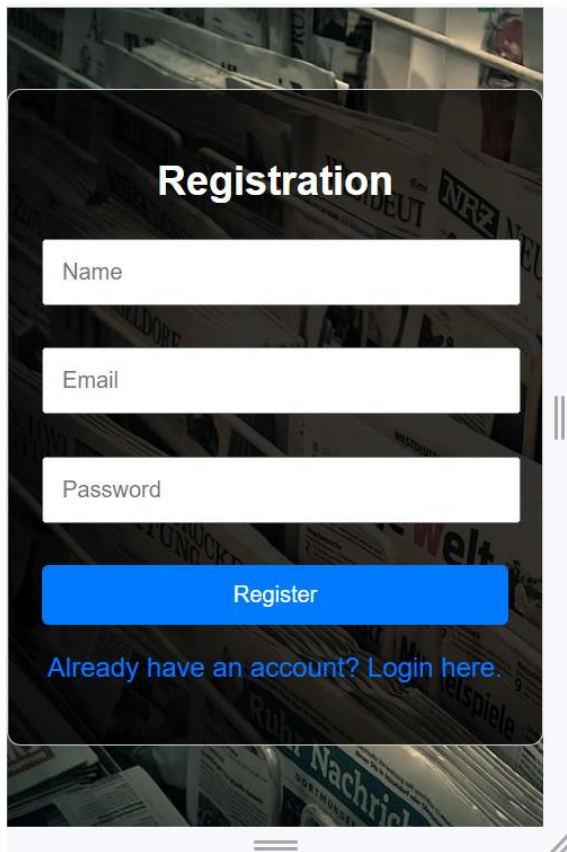




The main content area of the User Profile screen, containing several input fields and a list of sources. The fields are labeled 'Name:', 'Email:', 'Sources:', 'Categories:', and 'Authors:'. The 'Name' field contains 'bansi Faldu', the 'Email' field contains 'bansi@gmail.com', and the 'Sources' field contains 'Business Insider'. The 'Categories' field is a dropdown menu with 'Select...' as the placeholder. The 'Authors' field is empty. A hamburger menu icon is visible at the bottom left.



The Login screen of the News App. It features a background image of newspaper stacks. The screen has a dark overlay with the word 'Login' in white. Below the title are two input fields for 'Email' and 'Password'. A blue 'Login' button is positioned below the password field. At the bottom, there is a link that says 'Don't have an account? Register here.' A hamburger menu icon is visible at the bottom left.



## 8. Data Sources Integration

Run command in backend:

```
php artisan migrate  
php artisan app:srape-news
```

```
E:\xampp\htdocs\newsapp-docker\backend>php artisan app:srape-news  
Fetching news from NewsAPI...  
News scraping completed!
```

## conclusion

In conclusion, the News Aggregator is a powerful web application that seamlessly integrates news articles from diverse sources, offering users a personalized and responsive reading experience. Built with Laravel and React.js, the application follows best practices, employs Docker for containerization, and utilizes scheduled commands for efficient data



scraping. The comprehensive documentation guides users through the installation process, system architecture, and data source integration, ensuring a smooth setup and understanding of the project's inner workings. This News Aggregator serves as a testament to the effective implementation of full-stack development principles, providing a scalable and user-friendly solution for staying informed in the digital age.