Influencer Engagement & Sponsorship Coordination Platform Report

1. Introduction

The web application is designed to manage and facilitate interactions between users, influencers, sponsors, and administrators. It integrates functionalities for user authentication, role-based access, and data handling using various technologies and services. This report provides an overview of the application's features, architecture, implementation details, and future enhancements.

2. Features

- **User Authentication:**

  - Users can register, log in, and log out of the system. Passwords are securely hashed using bcrypt before being stored in the database.

  - JSON Web Tokens (JWT) are used for secure authentication and authorization.

- **Role-Based Access:**

  - The application supports different roles including Admins, Influencers, and Sponsors. Each role has distinct access rights and functionalities.

  - Redirection based on user roles is implemented to direct users to their respective home pages upon login.

- **Data Caching:**

  - Redis is used to cache frequently accessed data, improving performance and reducing load times.

  - Key functionalities include setting, getting, and checking the existence of cached data.

- **Email Notifications:**

  - SMTP is used to send email notifications for various purposes such as user registration and password recovery.

- **Campaign and Ad Management:**

  - Admins can manage users and oversee campaigns.

  - Sponsors can create campaigns, manage budgets, and view campaign statuses.

- Influencers can view and respond to ad requests.

3. Architecture

The application follows a client-server architecture with separate components for the frontend and backend:

- **Frontend:**
  - Built using Vue.js, the frontend provides a responsive and interactive user interface.
  - Utilizes `vue-router` for managing different views and navigation.
  - The application's layout and user interactions are managed through Vue components and templates.

- **Backend:**
  - Caching: Utilizes Redis for caching data.
  - Authentication: Implements password hashing and JWT for secure user authentication.
  - Email Service: Uses SMTP for sending emails.
  - Database Management: SQL-based relational database with tables for users, influencers, sponsors, campaigns, and ad requests.

- **Database:**
  - The application uses SQL tables to store and manage data related to users, influencers, sponsors, campaigns, and ad requests.

4. Implementation Details

- **User Model:**
  - Includes fields for ID, name, email, password (hashed), and role (Admin, Influencer, Sponsor).

- **Campaign Model:**
  - Stores details about campaigns such as name, description, budget, and associated sponsor.

- **Ad Requests Model:**

- Manages ad requests with fields for status, requirements, payment amount, and associated campaign and influencer.

- **Messages Model:**

- Keeps track of messages related to ad requests, including references to influencers, sponsors, and ad requests.

- **Redis Integration:**

- Caching functionalities for storing and retrieving data efficiently.

- Implementation includes setting expiration times for cached data.

- **Email Integration:**

- Sends emails using SMTP, with functionality for user notifications and other communication needs.

5. Conclusion

The web application provides a comprehensive platform for managing users, campaigns, and ad requests. By integrating caching, secure authentication, and email functionalities, the application ensures efficient data handling and user interaction. The architecture is designed to be scalable and extensible, supporting future enhancements and updates.

6. Future Enhancements

- **Improved User Interface:** Enhance the design with modern UI frameworks and ensure responsiveness across all devices.

- **Advanced Analytics:** Implement analytics for user activity, campaign performance, and ad request trends.

- **Social Features:** Introduce features such as user profiles, social interactions, and content sharing.

- **Integration with External APIs:** Expand functionality by integrating with third-party services for additional features.

- **Enhanced Security:** Regularly update security measures to protect user data and system integrity.

- **Automated Testing and CI/CD:** Implement continuous integration and continuous deployment (CI/CD) pipelines for automated testing and deployment.

7. References

- *[Redis Documentation]* (https://redis.io/documentation)

- *[JWT Documentation]* (https://jwt.io/)

- *[Passlib Documentation]* (https://passlib.readthedocs.io/en/latest/)

- *[SMTP Documentation]* (https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol)

- *[Vue.js Documentation]* (https://vuejs.org/)

- *[SMTP Server Configuration]* (https://support.google.com/mail/answer/7126229)

8. Video Tutorial

For a detailed walkthrough of the project, you can view the following video tutorial: [YouTube Video Link] (https://youtu.be/pfFgcf-FW_Y).