User Dashboard Application

Objective: Create a web application with user authentication that offers a rich set of features for user interaction and administration.

Technologies:
- Frontend: ReactJS
- Backend: NodeJS (Express)
- Database: Your choice (e.g., MongoDB, PostgreSQL, MySQL)

Task Details:

1. **Setup & Configuration**:
   - Initialize a new React project.
   - Set up a new Express server in a separate directory.
   - Connect your backend server with your preferred database.

- **Frontend**:
  Login Page: Authenticate against the backend and display error messages for incorrect credentials.
  Registration Page: Register new users, ensuring password confirmation.
  Dashboard Page: Display a welcome message, last login time, an activity feed, and a list of friends.
  Profile Page: View/update profile details and upload a profile picture.
  Notifications: Show notifications for specific events.
  Theme System: Allow users to choose themes and implement a dark mode toggle.

2. **Backend:**
   - Implement routes for user registration, login, profile management, and more.
   - Implement middleware for authentication.
   - Store user information, friend lists, chat messages, activity logs, etc., in the database.
   - Implement two-factor authentication.
   - Implement API rate limiting.

4. **Database**:
   - Store user details, encrypted passwords, last login time, profile images, activity logs, friend lists, and chat messages.

5. **Hosting & Deployment**:
   - Host both frontend and backend on a platform of your choice.

Extended Features:
   - **Activity Feed**: Display latest user activities with a filtering option.
   - **Friend System**: Send/receive friend requests and search for other users.
   - **User Roles & Admin Panel**: Different roles (e.g., user, admin) with an admin panel for site management.
   - **Analytics**: Display user engagement metrics on the dashboard.

Bonus:
- Use Redux or Context API for frontend state management.
- Implement password reset functionality.
- Deploy the backend with Docker.
- Backend unit tests using Jest.
- Mobile-responsive design.
- Third-party login integration (e.g., Google, Facebook).
- GraphQL instead of a RESTful API.
- Implement caching mechanisms.
- Frontend internationalization (i18n) for multiple languages.

Evaluation Criteria:
- Code Quality: Organization, modularization, and naming conventions.
- Error Handling: Robust feedback and handling of potential issues.
- Security: Password encryption, protection against SQL injections, secure data transmission.
- UI/UX: Intuitive design and user-friendly interface.

Important Pointers :

- Upload your zipped code folder to GDrive or Dropbox and provide us with the link. Ensure that there is a good readme file.
- Bear in mind that this task is to test your innovation capabilities. If you think there is a cool feature you can pull it off in the time span, feel free.
- The task should be submitted within 24hrs of receiving this task.