Requirement 4: User Management

Business Document

1. Summary

This is a user management for a currency exchange simulation system. It is implemented in Node.js with MySQL, supporting registration, authentication, profile management.

2. Specific Requirements

2.1. User registration with username, email and password

- Action: User submits registration form with username, email and password.
- Outcome: New user account is created and stored in the database.
- Definition of Done: User can register, data is validated, password is securely hashed, and user record appears in the users' table.

2.2. User authentication(login/logout)

- Action: User provides credentials to log in or requests to log out.
- Outcome: User receives authentication token on successful login; session is ended on logout.
- Definition of Done: Login returns JWT token, logout invalidates session/token, and only valid users can log in.

2.3. User profile management(view/update profile)

- Action: User views or updates their profile information.
- Outcome: Profile data is displayed or updated in the database.
- Definition of Done: User can view and edit their profile, changes are saved, and data is

validated.

2.4. Role-based access(admin, regular user)

- Action: System checks user role before allowing access to certain features.

- Outcome: Only users with appropriate roles can access restricted endpoints.

- Definition of Done: Role checks are enforced in middleware, and unauthorized access is blocked.

(Admin: manage all the users, manage all the currency exchanges.

Regular users: check exchange rates etc.)

2.5. Store user data in a database(MySQL)

- Action: User data is saved and managed in a MySQL database.

- Outcome: All user-related actions persist data in the users table.

- Definition of Done: User data is stored, retrieved, and updated in MySQL as per schema.

3. Related Table Design (users):

user_id (primary key, int)

email (unique, varchar)

password_hash (varchar)

username (unique, varchar)

role (boolean: 1-'admin', 0-'regular user')

created_at (timestamp)

updated_at (timestamp)

Solution Design

Solution (Node.js, MySQL):

Tips: Use JWT for authentication and mysql2 or sequelize for database access.

Implement endpoints for registration, login, profile update.

Store user data in a users table as specified.

1. Registration

- Endpoint: /register

- Input: username, email, password

- Output: status

- Info: validate input, store user in DB

2. Authentication

- Endpoint: /login

- Input: username, password

- Output: JWT token

- Info: validate credentials, return JWT token

3. Profile Management

- View profile

- Endpoints: /searchProfile (GET)

- Input: JWT token

- Output: user profile information by JSON

- Info: allow users to view their profile

- Update profile

- Endpoints: /updateProfile (PUT)

- Input: username, password, email

- Output: status
- Info: allow users to update their profile

4. Role-Based Access

- Admin account can update, delete users
- Regular users can only view the currencies

Middleware checks role before allowing access to admin endpoints.

Email is used for alert in other requirement.