Project command 1-12 (Matthew Zavalick, Dmitry Osipov) – Rohignya Charity Organisation WebSite (Web Application)

Description

Title: Rohingya Charity Organization Website (Web Application)

Description:
The Rohingya Charity Organization (RCO) website enables users to learn about the organization, view reports and news, explore specific goals (such as medical operations and educational projects), donate, contact the organization, or even become members. Admins have control over content management, such as posting news, updating goals, and managing reports.

User stories (briefly)

User Features:

- Users can create an account to receive updates, donate, and track their donation history.
- Users can change their password.
- Users can update their email address.
- Users can view all reports and goals.
- Users can donate.

Admin Features:

- Admins can post new information, including news.
- Admins can update or modify specific goals (e.g., fundraisers or operations).
- Admins can post reports and manage existing ones.

Volunteer Features:

- Volunteers can assign themselves to specific goals or campaigns.
- Volunteers can view special goals and contribute in non-monetary ways.

User stories (full)

Create account

Story

As a user, **I want to** create an account to receive updates, donate, and track my donation history **so I can** stay informed about the organization's work and manage my contributions easily.

API

Method: Post

endpoint: /signup

Request body:
user object
{ "email": "user@example.com", "password": "StrongPassword123!" }

Response:

success: user object
{ "email": "user@example.com", "password": "StrongPassword123!" }

error:
{ "error": "Password must contain at least one uppercase letter, one number, and one special character" }
{ "error": "Email must be in a valid format" }

Acceptance Criteria:

1. **Account Creation**:
    a. Users can create an account by entering an email and password.
    b. Password must meet strength requirements:
        1. At least 8 characters.
        2. One uppercase letter, one number, and one special character.
    c. Email must be in a valid format (e.g., user@example.com).
2. **Error Handling**:
    a. If the password or email does not meet requirements, appropriate error messages should display.
3. **Email Verification**:

a. Upon successful sign-up, a verification email will be sent to the user.

b. Users cannot log in or donate until they verify their email.

Update password

Story

As a user, **I want to** change my password **so I can** ensure my account stays secure.

API

Method: PUT

Endpoint: /users/{user_email}/password

RequestBody:
Old and new password
{ "current_password": "OldPassword123!", "new_password": "NewStrongPassword456!" }

Response:

Success: { "message": "Password successfully updated" }

Error: { "error": "Current password is incorrect" }

Acceptance Criteria:

1. **Password Update**:
    i. Users must provide their current password and a new password.
    ii. The new password must meet strength requirements (similar to sign-up).
2. **Error Handling**:
    i. If the current password is incorrect, an error message should appear.
    ii. If the new password does not meet the strength requirements, an appropriate error message should be displayed.

Update email address

Story

As a user, **I want to** update my email address **so I can** receive notifications and account updates at my preferred email.

API

Method: PUT

Endpoint: /users/{user_email}/email

RequestBody:
{ "current_password": "Password123!", "new_email": "newemail@example.com" }

Response:

Success: { "message": "Email address successfully updated" }

Error: { "error": "Please provide a valid email address" }

Acceptance Criteria:

1. **Email Update**:
    a. Users must enter their current password and the new email address.
    b. The new email must be in a valid format (e.g., user@example.com).
2. **Error Handling**:
    a. If the current password is incorrect, an error message should display.
    b. If the email format is invalid, an error message should appear.

See all reports

Story

As a user, **I want to** view all reports **so I can** stay informed about the organization's activities and progress and also verify their work.

API

Method: GET

Endpoint: /our_work

RequestBody: none

Response: list of report or error

Acceptance Criteria:

1. **Report and News Viewing**:
    a. Users can access a page or endpoint displaying all reports.
    b. Reports and news should be listed in chronological order with the latest first.
2. **Error Handling**:
    a. If reports or news fail to load, an error message should display.

See all goals

Story

As a user, **I want to** view all specific goals **so I can** choose the causes or projects I want to support based on my preferences.

API

Method: GET

Endpoint: /

RequestBody: none

Response: List of goals

Acceptance Criteria:

1. **Goal Viewing**:

    a. Users can access a list of all available goals, with details like title, description, target amount, current amount, and deadlines.

    b. Goals should be displayed in a user-friendly manner, sorted by urgency or deadline.

2. **Error Handling**:

    a. If the goals fail to load, an error message should appear.

Donate

Story

As a user, **I want to** donate either generally or to a specific goal **so I can** support the organization or a cause that matters most to me.

API

Method: POST

Endpoint: /donate

RequestBody:

{ "user_email": "true.hero@example.com", "amount": 100, currency: "euro", "donation_type": "general" }

Response:

Success: New donation Object

{ "donation": { "donation_id": "abc123", "user_id": "12345", "amount": 100, "donation_type": "goal", "goal_id": "67890", "timestamp": "2024-10-13T14:30:00Z", "status": "successful" } }

Error: { "error": "Failed to process donation" }

Acceptance Criteria:

1. **Donation Options**:
    a. Users can choose to make a general donation or donate to a specific goal.
    b. If the donation is tied to a goal, the goal_id must be included.
2. **Successful Donations**:
    a. The system should return a detailed **donation object** after successful donations, including information like donation_id, amount, user_id, and timestamp.
3. **Error Handling**:
    a. If a donation fails, the system should return an error message with relevant information.
4. **Security**:
    a. Ensure secure processing of payment information and donation records.

Post new report

Story

As an admin, **I want to** post new reports and **so I can** provide users a possibility to verify our work.

API

Method: POST

Endpoint: /our_work

RequestBody: Report

{ "title": "Annual Report 2024", "content": File with all info, "date": "2024-10-13" }

Response:

Success: {"message": "Report successfully posted"}

Error: {"message": "Report posting failed"}

Acceptance Criteria:

1. **Posting New Reports**:
    a. Admins must provide a title, content, and date to post a new report.
    b. The system should return a message
2. **Error Handling**:
    a. If posting a report fails, the system should return an error message.
3. **Security**:
    a. Only authenticated admins should have permission to post reports.

Update existing report

Story

As an admin, **I want to** update existing reports **so I can** ensure that the information provided to users is accurate and up-to-date.

API

Method: PUT

Endpoint: /our_work/{report_id}

RequestBody:

{ "title": "Updated Annual Report 2024", "content": Updated details about the organization’s activities in a file, "date": "2024-10-14" }

Response:

Success: { "message": "Report successfully updated" }

Error: { "message": "Failed to update report" }

Acceptance Criteria:

1. **Updating Reports**:

a. Admins can update an existing report by providing the report ID, along with the new title, content, and date.

b. The system should confirm successful updates with a message and return the message.

2. **Error Handling**:

a. If the update fails (e.g., due to an invalid report ID), the system should return an appropriate error message.

3. **Security**:

a. Only authenticated admins should have permission to update reports.

Assign yourself to specific goal

Story

As a volunteer, **I want to** assign myself to specific goals or campaigns **so I can** contribute my time and skills to causes I am passionate about.

API

Method: POST

Endpoint: /volunteer

RequestBody: { "volunter_email": "volunteer@example.com", "goal_id": "67890" }

Response:

Success:{ "message": "Successfully assigned to the goal" }

Error: { "error": "Failed to assign to the goal" }

Acceptance Criteria:

1. **Goal Assignment**:

    a. Volunteers can submit a request to assign themselves to a specific goal or campaign using the goal ID.

    b. The system should confirm the assignment and return a message

2. **Error Handling**:

    a. If the assignment fails (e.g., the goal ID is invalid or the volunteer is already assigned), an appropriate error message should be returned.

2. **Security**:

    a. Only authenticated volunteers should be able to assign themselves to goals or campaigns.

Domain Model:

User:
Represents registered users who can donate, receive updates, and track their donation history.

Admin:
Users with elevated privileges who can manage site content such as news, reports, and goals.

Goal:
Specific objectives the organization is working toward (e.g., medical operations, educational projects), which users can donate to or volunteers can join.

News:
Updates posted by the organization to keep users informed about current events and activities.

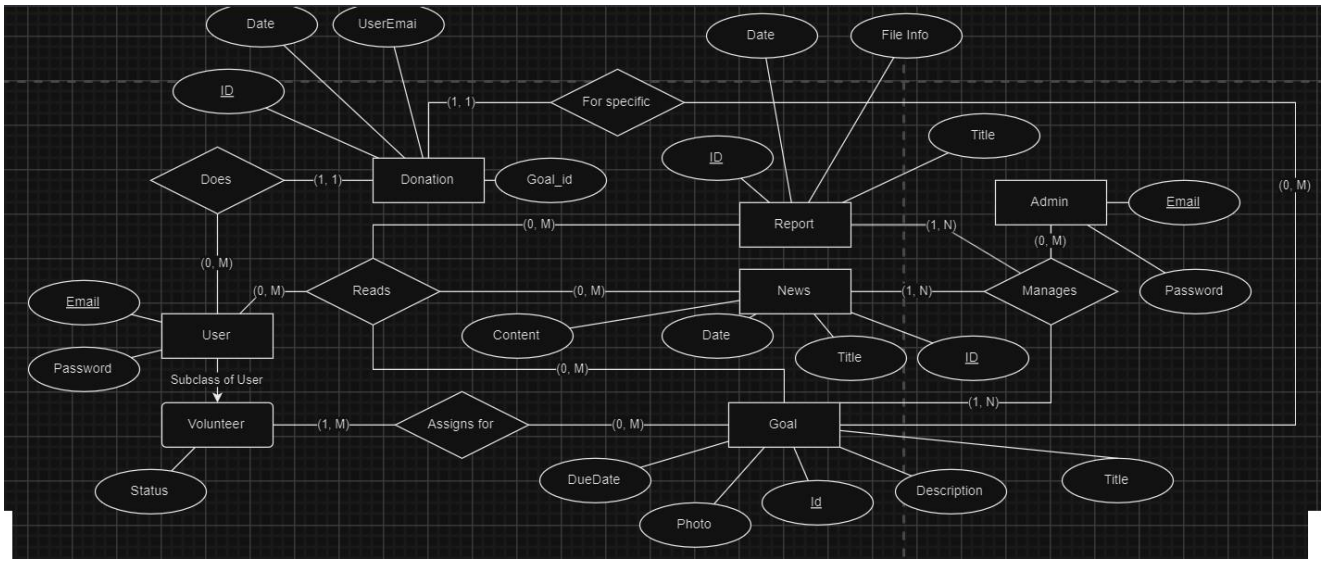
Donation (Transaction):

Records of user contributions, tracking amounts, dates, and associated goals.

Volunteer (Member):
Registered users (subclass of user) who can actively participate in special goals. Volunteers can view, join, and contribute to specific campaigns.

Report:
Document that provide updates about the organization's activities, goals, and achievements. Reports can be created, updated, and managed by admins to keep users informed and provide them information to be sure about our charity organisation really works

Conceptual Model



Logical Model