Data Plan

Group Name/Number: 96

Feature/Interaction	Purpose/Description	Request Data Source	Request sent to server	Processing Needed/Response Data Source	Response sent to client	Request Type & Path

Automatic Emails	By sending users timely relevant emails in response to pre-defined triggers or information, the volunteer website's "automated email" function aims to keep the user free properly integrated to automate communication activities A product that can send updates, notifications, follow-up messages and reminders to improve user experience, increase participation, and force strengthen relationships within the volunteer community by using automation to deliver consistent, personalized and targeted communication.
	
	Organizations, employees, and incentive programs can use the "automated email" functionality to send emails to users at predetermined intervals or in response to special events or actions Emails can be sent a automated delivery of user-designed and volunteer website links , for example; event reminders, confirmation emails, thank-you emails, notification emails and follow up emails.
		The Automatic Email feature relies on various data sources to generate and deliver automated email communications to users based on specific events. Email templates are populated with dynamic content sourced from the system including the email subject and HTML content. Additional contextual data related to the triggering event, such as event details or relevant metadata, may be included to enhance the usefulness of the email content.	{
			"query": "[id_number]",
			"eventType": "[newEvent]",
			"emailAddress": "example@email.com",
			"emailSubject": "New event...",
			"emailContent": "Event details...",
			}
				Retrieve user information including the userId, username, and email address from the database based on the provided userID. Use the appropriate email template based on the event type, such as an event email template for an event email. Populate the email template with dynamic content including the user's name and other details. Send the generated email to the user's email address using an SMTP server. Return a success response indicating the email was successfully generated and sent.	{
					"status": "success",
					"message": "Email sent successfully to user.",
					"emailDetails": {
					"userId": "[id_number]",
					"emailAddress": "example@email.com",
					"emailSubject": "Event details",
					}
					}
						Request type: POST
						Path: /sendAutomaticEmail

Making Posts	The purpose of the 'making posts' feature is to facilitate communication and engagement among users, admins, organisations and volunteers. It provides the ability for admins to share information, update announcements, and post stories related to volunteering. By allowing users to do this it fosters a sense of community, collaboration and communication, ultimately improving user engagement and participation.
	
	The description of this feature is the ability for users to write text-based posts, upload images or even attach relevent documents. They can also catagorize their posts to fit a certain topic to make it easier for it to reach the target audience. Additionally, other users are able to view, comment and like these posts to facilitate engagement between users.
		Each organisation will have a 'Make Post' button which is accessible only to managers and can not be seen by general members of the organisation. When a manager clicks the button, the system captures the 'userID' of the loged-in user along with the content of the post. The manager is able to add any additional attachments such as flyers, documents, or images to the post.	{
			"postId": "[post_id]",
			"userId": "[user_id]",
			"postTitle": "Title of the post",
			"postContent": "Content of the post",
			"postCategory": "Category of the post",
			"postTags": ["tag1", "tag2", "tag3"],
			"postAttachments": ["attachment1.jpg", "attachment2.pdf"],
			"postVisibility": "Public",
			"postStatus": "Draft"
			}
				Receive the POST request with the 'postContent' and 'userID' from the client. Create a new post in the database with details about the userID, postContent, and timestamp. Assign this entry with a unique postID for referencing purposes. Log the details of the post creation to audit and track the post. Indicate a successful post creation with a status of "success" along with a confirmation message stating the postID.	{
					"status": "success",
					"message": "Post created successfully.",
					"postDetails": {
					"postId": "[post_id]",
					"userId": "[user_id]",
					"postTitle": "Title of the post",
					"postCategory": "Category of the post",
					"postVisibility": "Public"
					}
					}
						Request type: POST
						Path: /createPost

RSVP for events	The purpose of 'rsvp for events' feature on the website is to be able to communicate, streamline event management, and allow for easier attendence tracking by the organisers. It enables users to express their intetion to attend or stay out for specific volunteering events, allowing organisers to plan accordingly. By providing this convinient way, this feature enhcances event coordinisation, increase attendent rates and foster a better community.
	
	The description of this feature is the ability for users to browse through a list of available events, view event details, and indicate whether they plan to attend this event or not.
		When the user clicks the "RSVP" button for an event, the system collects the unique identifier of the event, the userID, and the user's RSVP status (rsvpStatus), which indicates whether the user is "going", "interested", or "not going" to the event. This information is packaged into a JSON object and sent to the server in a POST request. The request allows the system to record the user's response to the event, ensuring that the user's intentions are accurately captured and stored in the database.	{
			"eventId": "[event_id]",
			"userId": "[user_id]",
			"attendanceStatus": "Attending",
			"additionalNotes": "Notes about special requests"
			}
				When a user submits an RSVP for an event, the system begins by processing the received POST request, which includes the eventId, userId, and rsvpStatus. The first step is to validate the user's authentication and authorization to ensure they are logged in and permitted to RSVP for the event. Next, the system verifies that the event exists in the database and that the provided eventId is valid. Following validation, the system updates the database to reflect the user's RSVP status for the specified event, linking the userId with the eventId and storing the rsvpStatus.Once the database is updated, the system generates a response to inform the client of the outcome. If the RSVP process is successful, the response includes a status of "success" and a message confirming the RSVP.	{
					"status": "success",
					"message": "RSVP recorded successfully.",
					"eventDetails": {
					"eventId": "[event_id]",
					"eventName": "Volunteer Cleanup Drive",
					"attendanceStatus": "Attending"
					}
					}
						Request type: POST
						Path: /rsvpRecorded

Joining an organisation	Users can sign up and login to the website to join volunteer organisations. Once registered as a user, they can search and select organisations that interest them. They can view important information about the organisations such as their mission, location, and upcoming events. After joining, they will receive notifications and updates about new events and opportunities.
	
	When a new user has joined, the manager will be able to see the user as a member of their organisation and can see their contacts information. When managers post events, user that have joined the organisation will be able to register for these events.
		Each organisation will have a "Join organisation" button. When users click this button, they are instantly accepted into the volunteer organisation. The system processes this request and updates the database the the user's information including their username and contact info.	{
			"action": "joinOrganization",
			"userId": "[id_number]",
			"organizationId": "[id_number]"
			"timestamp": "[date_and_time]"
			}
				When user initiates join request, the system will first verify their authentication status to check that they are logged in and check that the organisationId is registered in the database. The system will also check if the user has already joined this organisation to prevent duplicate members. Once validation is completed, the system will create new membership record and add the user to the organisation.	{
					"status": "success",
					"message": "You have successfully joined the organization.",
					"organization": {
					"id": "[id_number]",
					"name": "Volunteer organization name",
					"description": "Description of the organization",
					"location": "Organization location",
					}
					}
						Request type: POST
						Path: /joinOrganisation

Editing Profile	The purpose of this feature is to allow volunteers to manage their own profiles to add or update their personal information, preferences, and settings within their profile. This empowers users to have control over their own identity on the website and allows them to have accurate up-to-date information about themselves. By offering this feature for customization, it fosters trust and transparency, and ensures that users can present themselves as they want to organisers, peers, volunteers and potential collaborators.
	
	Users registered through the specific profile management interface on the volunteer website can access and edit their profile information and settings through the "Edit Profile" option. The user profile can be updated in several ways, such as; personal information, contact information, skills and interests and privacy settings
		Users modify their profiles by filling out a form with editable fields for various details like name, contact info, skills, interests, and etc. Clicking "Save Changes" prompts instant processing, updating the user's profile in the database. Successful updates display a confirmation, while errors trigger appropriate messages, ensuring accurate and up-to-date profiles.	{
			"userId": "[user_id]",
			"profileData": {
			"name": "John Doe",
			"email": "john.doe@example.com",
			"phone": "+1234567890",
			"bio": "",
			"skills": ["Volunteer Management", "Event Planning"],
			"interests": ["Education", "Environment", "Healthcare"],
			"socialLinks": {
			"facebook": "https://www.facebook.com/john.doe",
			"twitter": "https://twitter.com/johndoe"
			}
			}
			}
				The editing profile features reiles on many different ascpets.The user's unique identifier (userId) and updated profile information (profileData), which includes the user's name, email address, phone number, biographical information, skills, interests, and social media links, are updated by the system when the user starts a profile. The datasource object also stores metadata associated with profile update events, including the data source type, trigger event, and context information such as the date and time of the most recent update and provided by the user who made the change.	{
					"status": "success",
					"message": "Profile updated successfully.",
					"profileDetails": {
					"userId": "[user_id]",
					"name": "John Doe",
					"email": "john.doe@example.com",
					"phone": "+1234567890",
					"bio": "",
					"skills": ["Volunteer Management", "Event Planning",],
					"interests": ["Education", "Environment", "Healthcare"],
					"socialLinks": {
					"facebook": "https://www.facebook.com/john.doe",
					"twitter": "https://twitter.com/johndoe"
					}
					}
					}
						Request type: POST
						Path: /profileUpdated