

Low Level Design

Introduction

About Power Awareness Tool

The Power Awareness Tool 1.0 was launched by Partos in February 2020, as a tool to help make power relations more visible. In 2022, the tool was evaluated, resulting in the improved Power Awareness 2.0 version which serves as the basis for this assignment. The tool gives the different stakeholders the chance to share and monitor their insights on their decision-making practice within their project. It is part of a shared objective of Partos and involved members to actively promote and work to shift power, understand the effects of power dynamics on the decision-making process and create transparency on why, how and by whom decisions are being made.

The Purpose of Power Awareness Tool

The Power Awareness Tool (PAT) is designed to help stakeholders make power relations within their projects more visible, facilitating transparency in decision-making processes. **PAT 2.0** aims to build on the lessons learned from the initial version, offering an improved user experience and more effective insights for stakeholders. By helping users better understand power dynamics and how these affect the decision-making process, PAT promotes more equitable partnerships and empowers stakeholders to take informed action. The digital platform we are developing will bring PAT 2.0 to life in a more accessible and user-friendly format, supporting its core mission.

Key Objectives

1. **Implementing a user-friendly platform:** The primary goal is to digitize PAT 2.0, focusing on usability, ease of navigation, and accessibility. This will enable facilitators and participants to interact with the tool in an intuitive way, making it easier to capture, manage, and analyze data related to decision-making processes in their projects.
2. **Core functionality first, with room for future expansion:** The platform will begin with essential features to support user needs, but it will also be designed to accommodate future expansion. As users provide feedback, new features and improvements can be added, ensuring the platform remains flexible and evolves with changing requirements.

Functional Overview

Here is a breakdown of the core functionalities that will be implemented in the first version of the digital PAT 2.0 platform:

1. Home Page

- **Multi-language support:** The home page will include options for users to select their preferred language, allowing for greater inclusive and wider use across different regions and organizations.
- **Brief tool description:** A concise and clear explanation of the purpose of the Power Awareness Tool will be available, helping new visitors understand its value and relevance.
- **Downloadable PDF:** A downloadable guide or overview of PAT will be available as a PDF file, providing users with more in-depth information and allowing them to learn about the tool offline.
- **Contact and help options:** Easily accessible contact information and help documentation will ensure that users can reach out for support when needed.
- **Account creation for session facilitation:** Facilitators can register and create accounts through the home page, granting them access to all relevant platform features, including session management, report generation, and participant invitations.

2. PAT Application

The core of the digital platform, where facilitators and participants interact with the tool's decision-making features:

- **Invite participants:** Facilitators will have the ability to invite stakeholders and project participants a unique invitation code or links that can be sent via email. This ensures that all relevant parties are included in the decision-making and power analysis process.
- **Create lists of partner organizations and key decisions:** Facilitators can set up lists of partner organizations involved in the project, and important decisions to be analyzed. This data will help participants reflect on who holds decision-making power and how it impacts the project's outcomes.
- **Navigate through PAT steps:** The tool will guide users through a step-by-step process that helps them document and analyze power dynamics. These steps could include identifying key stakeholders, mapping out power relationships, and evaluating decision-making practices.
- **Report generation and printing:** After completing the steps, facilitators will have the option to generate detailed reports summarizing the findings. These reports will be available for download or printing, making it easy to share insights with all stakeholders.

3. Access and Rights Management

Ensuring that data privacy and user access control are handled effectively is critical to maintaining trust and transparency:

- **Role-based access control (RBAC):** Different user roles (facilitator, participant, admin) will have different permissions within the tool. For example, facilitators will have full editing and management rights, while participants will only have view or limited input access, depending on their role in the project. Admin can act as either a facilitator or participant and has additional privileges to manage users and view statistics.
- **Data privacy:** The platform will follow best practices for securing user data. All data entered will be encrypted both at rest and in transit, ensuring compliance with GDPR and other data protection standards. Users will have control over their own data, and the platform will ensure that sensitive insights are only visible to authorized parties.
- **Editing rights for facilitators:** Only facilitators will have the ability to edit case information, participant lists, and decision-related data. This ensures that data integrity is maintained and that participants' contributions are not altered by other users.
- **Viewing rights for participants:** Participants will be able to view their own contributions and the results of the power analysis, but they will not be able to edit content added by the facilitator or other participants. This access model preserves the collaborative nature of the tool while ensuring data control.

4. Statistics and Analytics

The platform will offer built-in statistical tracking features to monitor the effectiveness and usage of the tool over time. This data can be valuable for both the internal development team and stakeholders using PAT:

- **Usage statistics:** Track key usage metrics such as the number of active accounts, the frequency of logins, and the number of ongoing sessions or projects.
- **Participant engagement:** Monitor how many participants are involved in each session and their level of engagement (e.g., the number of insights shared, steps completed, etc.).
- **Active accounts:** Provide an overview of how many users and organizations are actively using the platform.
- **Reporting outcomes:** Analyze the types of reports generated, the common themes emerging from the data, and how insights are being utilized in decision-making. These analytics can help stakeholders understand the impact of PAT in real-world projects and inform future improvements.

User Workflow

This section outlines the user journey and interactions within the Power Awareness Tool 2.0 (PAT 2.0). From account creation to managing and participating in sessions, each step of the workflow is designed to be intuitive, ensuring users can easily navigate and use the platform's core features.

1. Create Account

The first step for any user is to create an account, which provides access to the platform and its functionalities. The account creation form includes the following fields:

- **Full Name:** The user's full name, which will be used to identify them across sessions.
- **Select Country of Origin:** Users choose their country from a per-defined list, providing geographical context for their participation.
- **Email Address:** Users provide a valid email address for account verification and ongoing communication.
- **Password:** Users create a secure password, which must be validated through platform security standards *except symbols*.
- **Confirm Password:** To ensure accuracy, users must re-enter their password.
- **Checkbox (Agree to PAT Terms):** Users must agree to the PAT 2.0 terms and conditions before proceeding.

Once all fields are filled out and validated, the platform will send a verification link to the user's email. After verifying the email, the account is fully activated, and the user is ready to begin using the platform.

2. PAT Dashboard

Once logged in, users will be directed to the **PAT Dashboard**. From here, they have two main options: **Create Session** (for facilitators) or **Join Session** (for participants).

2a. Create Session (For Facilitators)

Facilitators can create a new PAT session by filling in the following fields:

- **Name of the Partnership (String):** The title of the PAT session, such as the name of the project or collaboration being evaluated.
- **Countries:** Facilitators select the countries relevant to the session from a per-defined list.
- **Purpose of Creation:** Select 'Purpose of Creation': Users specify the purpose of their session by choosing from the following options:
 - I am just checking out the Power Awareness Tool (PAT) out of curiosity.
 - I want to facilitate a real PAT session in an existing partnership to evaluate the way decisions are made, and to determine how decisions should be made.
 - I want to facilitate a real PAT session in an emerging partnership to determine the way decisions will be made.
 - I want to facilitate a real PAT session in our organization or team to evaluate the way decisions are made, and to determine how decisions should be made.
 - I want to use the PAT for an individual thought experiment just to figure out how decision making could work in a partnership.

- I want to use the PAT to explore if this is a useful facilitation tool for consultancy assignments.

- **Date:** The date when the session will be held or started.
- **Add Organisations:** Facilitators can add up to 8 organizations. For each organization, the facilitator enters: **Organisation Name** and **Acronym**. These organizations will be included in a drop-down list for participants to select from when they join the session.
- **Context (Textbox):** A description or contextual background of the partnership, providing relevant information for the session participants.

Once the session is created, the platform will generate a unique **join code**. Facilitators can share this code with participants via email or by copying the code using a **Copy Code** button on the interface.

2b. Join Session (For Participants)

Participants can join a session by following these steps:

- **Enter Join Code:** Participants enter the code provided by the facilitator to access the session.
- **Select Organisation:** Participants select their organization from a drop-down list populated by the facilitator during session setup.
- **Enter Role:** Specify your organization's role.

Once this information is submitted, participants can join the session and begin navigating through the different sections of the PAT session.

3. PAT Session Structure

Each PAT session is organized into several sections that guide users through a process of evaluating decision-making power and participation within their project or partnership. The main navigation items for a PAT session include:

1. **Decisions (List of Important Decisions):** Participants and facilitators write the list of key decisions made within the project.
2. **Level of Participation (Determine Actual Level of Participation):** Users assess and record the actual level of participation by each stakeholder or organization involved.
3. **Actual Participation (Reflect on Actual Level of Participation):** Participants reflect on whether the recorded levels of participation were appropriate and fair.
4. **Desired Level of Participation:** Users document what they believe should be the ideal or desired level of participation for each stakeholder.
5. **Action for Change:** Participants and facilitators suggest and discuss actions that could be taken to shift power dynamics and improve participation equity.

6. **Close PAT Session:** Facilitators close the session once all steps have been completed, and final reports are generated.

Each section in the navigation bar includes a status indicator: **In Progress** – if some data has been entered but the section is incomplete. **Not Started** – if no data has been entered yet.

3a. Session Interactions

The platform offers different roles and interactions for **Facilitators** and **Participants** during a PAT session.

a. As a Facilitator

Facilitators have full control over the session. They can:

- **Navigate through all sections:** Facilitators can access every step of the session, such as documenting key decisions, determining participation levels, reflecting on power dynamics, and suggesting actions for change.
- **Fill out all fields:** They are responsible for completing the necessary data for each step, ensuring that all relevant information is captured from participants and about the partnership.
- **Monitor participant contributions:** Facilitators can track input from participants, helping guide discussions or ensure everyone's voice is heard.
- **Generate reports:** At any point during the session, facilitators can generate preliminary reports based on the data entered so far.
- **Publish and close the session:** When all steps are completed and the session is ready for finalization, the facilitator can click the **Publish** button.

Once the facilitator clicks the **Publish** button, the session is officially marked as complete. Here's what happens after:

1. **Session Becomes Read-Only:** After publishing, the session is **locked** and can no longer be edited by the facilitator or participants. All fields and data become read-only, ensuring that the information recorded during the session remains unchanged and can serve as a permanent record.
2. **Session Moves to the Dashboard:** The closed session will appear under a dedicated section on the dashboard labeled **Closed Sessions**. These sessions are no longer active and serve as an archive of completed analyses and reports. Users can revisit these closed sessions to review the data and insights captured, but they will not be able to modify any of the content.
3. **Access to Reports:** Facilitators and participants will still be able to download or view the session's reports, which summarize the outcomes and key findings. This ensures that the insights gathered from the session can be easily shared with other stakeholders or used in future discussions or evaluations.

The finalization process, marked by clicking the **Publish** button, transitions the session from an editable, active state to a closed, read-only state. This ensures data integrity and creates a permanent record of the insights gathered during the session. By moving completed sessions to the **Closed Sessions** list on the dashboard, facilitators and participants can always access past analyses for reference or reporting without risking accidental changes or loss of data.

b. As a Participant

Participants have limited control and can only contribute to certain sections, depending on their role in the project. They can:

- **Navigate the session:** Participants can view all sections of the PAT session and contribute where applicable.
- **Post a comment:** In the last step (Step 6) of the active session, participants can leave any final remarks they might have on the session before the facilitator closes it.
- **View the finalized session:** Once the session is published, participants can still access the data they contributed and review the entire session in read-only mode via the **Closed Sessions** section on their dashboard.

4. Users Page

This page is accessible only to admin users and displays user data in a tabular format. Admins can perform the following actions:

- **Search by Name:** Filter users based on their names.
- **Update User Role:** Change a user's role between regular user and super admin.
- **Delete User:** Remove a user from the system.
- **Download Users Data:** Downloads user data in CSV format, generated from a manually executed server command.

5. Statistics Page

This page is accessible only to admin users and displays various statistics related to the system. The following information is provided:

- **Number of accounts:** Displays the total count of user accounts.
- **New Accounts in the Last 30 Days:** Shows the number of accounts created in the past month.
- **Number of sessions completed:** Indicates the total number of sessions that have been completed.
- **Number of sessions completed in last 30 days:** Displays the number of sessions completed in the past month.

- **Number of cases within each purpose category:** Provides the number of cases categorized by their purpose.
- **Number of cases created per month grouped by year:** Shows the number of cases created each month, grouped by year..

6. FAQs Page

The FAQs page offers a collection of frequently asked questions along with their corresponding answers to assist users in understanding everything they need to know about the Power Awareness Tool.

Architecture

The architecture of the Power Awareness Tool 2.0 (PAT 2.0) is designed to ensure scalability, security, and flexibility while delivering a seamless user experience. The system will follow a **modular architecture** that separates the front-end, back-end, and data layers, ensuring clear separation of concerns, maintainability, and ease of future enhancements.

The core components of the architecture include the **Next.js** frontend, the **Django** backend, and a **PostgreSQL** database, all orchestrated via **Kubernetes** for scalability and reliability. Continuous integration and deployment will be managed using **GitHub Workflows**.

High-Level Architecture Overview

1. Front-end (Next.js)

This is the user-facing layer where participants and facilitators interact with the platform. Next.js (built on React) will be used for rendering user interfaces, routing, and server-side rendering (SSR) to improve performance and SEO (for Home Page).

Components:

- **User Authentication and Authorization:** Login, account creation, and role-based access control (RBAC).
- **Dashboard:** Displays session options (Create Session, Join Session, Closed Sessions).
- **Session Management:** Forms and interactive elements for managing sessions, navigating steps, and data input.
- **Multi-language Support:** User interface will offer multiple language options to accommodate international users.
- **Communication:** The front-end will communicate with the back-end via **RESTful APIs**, and data will be fetched or submitted in JSON format.

2. Back-end (Django)

This is the core of the platform, responsible for handling business logic, validating user input, managing sessions, and interacting with the database. Django will be used to develop the API layer and handle back-end tasks like session management, user roles, and permissions.

Components:

- **API Layer:** RESTful APIs for handling all interactions between the front-end and back-end.
- **Session Management:** Creation, storage, and access control of PAT sessions.
- **Authentication and Authorization:** Secure login and user roles (facilitators and participants) management.
- **Email Service:** Handles sending of verification emails, invitation links, and notifications.
- **Communication:** The back-end will interact with the database using Django's ORM, ensuring efficient data retrieval and manipulation.

3. Database (PostgreSQL)

To store and manage all data related to users, sessions, organizations, and reports. PostgreSQL, a robust relational database, will be used for data storage.

Data Model:

- **User Data:** Stores user profiles, roles, and account-related information.
- **Session Data:** Stores all session-related data, such as organizations involved, decisions made, participation levels, etc.
- **Statistics and Reports:** Stores platform usage statistics and generated reports for analysis and review.
- **Interaction:** The Django ORM will be used to define the models and handle all CRUD (Create, Read, Update, Delete) operations on the database.

4. Container Orchestration (Kubernetes)

Kubernetes will be used to deploy and manage the containerized services (Next.js and Django applications). It ensures the platform remains scalable, fault-tolerant, and highly available.

Key Components:

- **Pods:** Each service (frontend, backend) will run inside its own containerized environment.
- **Load Balancing:** Kubernetes will automatically manage load balancing between instances of the frontend and backend to ensure smooth performance under high traffic.
- **Auto-scaling:** The system can automatically scale resources up or down based on traffic or load to optimize performance.
- **Secrets Management:** Sensitive information like API keys, passwords, and database credentials will be securely managed using Kubernetes secrets.

5. CI/CD Pipeline (GitHub Workflows)

Purpose: Continuous integration and deployment will be automated using GitHub Workflows to ensure smooth development and deployment processes.

Workflow:

- **Automated Testing:** Each code push or pull request will trigger tests to ensure code quality.
- **Build and Deploy:** Successful tests will automatically trigger build processes and deploy to the appropriate environment (staging, production).
- **Environment Management:** Separate environments for test, and production will be maintained to ensure stability in releases.

Security and Stability Considerations

Security is a paramount concern in the architecture of PAT 2.0. Here are the key security measures implemented:

- **Authentication & Authorization:** Ensures that only authorized users can access specific functionalities based on their roles (facilitator, participant, admin).
- **Data Encryption:** All sensitive data is encrypted both in transit (using HTTPS) and at rest (using database encryption).
- **Secure APIs:** Implements token-based authentication (e.g., JWT) for API access, protecting against unauthorized access.
- **Regular Audits:** Conducts regular security audits and vulnerability assessments to identify and mitigate potential threats.
- **Compliance:** Adheres to data protection regulations such as GDPR, ensuring user data is handled responsibly and transparently.

The architecture is built to handle growth and ensure a smooth user experience:

- **Horizontal Scaling:** Kubernetes allows for adding more instances of front-end and back-end services as the user base grows.
- **Load Balancing:** Efficiently distributes traffic to prevent any single component from becoming overwhelmed.
- **Caching:** Implements caching strategies at various levels (e.g. assets caching for frequent API requests) to reduce latency and improve response times.
- **Optimized Queries:** Ensures database queries are optimized for performance, reducing load times and improving overall efficiency.

Database Design

The database for the Power Awareness Tool 2.0 (PAT 2.0) will be designed using **PostgreSQL** as the relational database management system. The schema is designed to efficiently store user data, session details, organizations involved, and various statistics and reports generated by the platform. The following is a breakdown of the key tables that will be used in the database, along with a brief explanation of their fields.

1. Users Table

This table will store information about the users of the platform, including facilitators and participants.

| Field Name | Data Type | Description |
|-------------------|------------------|--|
| id | BIGINT (Primary) | Unique identifier for each user. |
| full_name | VARCHAR(255) | Full name of the user. |
| country | VARCHAR(100) | Country of the user’s origin (predefined list). |
| email | VARCHAR(255) | Email address (must be unique). |
| password_hash | TEXT | Hashed password for security purposes. |
| verification_code | UUID | For verifying user's email |
| is_verified | BOOLEAN | Indicates whether the email is verified (True / False). |
| is_superuser | BOOLEAN | Indicates whether the user has administrative privileges. The default value is False . |
| created_at | TIMESTAMP | Timestamp when the account was created. |
| updated_at | TIMESTAMP | Timestamp when the account was last updated. |

2. Sessions Table

This table stores information about each session, which is created by a facilitator for partnership evaluation or power mapping.

| Field Name | Data Type | Description |
|--------------|----------------------|---|
| id | BIGINT (Primary) | Unique identifier for each session. |
| user_id | BIGINT (Foreign Key) | Refers to the id of the facilitator in the Users table. |
| session_name | VARCHAR(255) | Name or title of the session. |
| countries | TEXT (Serialized) | List of countries relevant to the session. |

| Field Name | Data Type | Description |
|--------------|--------------|---|
| purpose | ENUM | Purpose of the session |
| date | DATE | Date when the session is held or started. |
| context | TEXT | A description of the session's context. |
| notes | TEXT | A Notes before closing PAT Session (optional) |
| summary | TEXT | A Summary before closing PAT Session |
| join_code | VARCHAR(100) | Unique code generated for participants to join the session. |
| is_published | BOOLEAN | Indicates whether the session is published and closed. |
| created_at | TIMESTAMP | Timestamp when the session was created. |
| updated_at | TIMESTAMP | Timestamp when the session was last updated. |
| closed_at | TIMESTAMP | Timestamp when the session was published/closed. |

3. Organizations Table

This table stores information about the organizations involved in each session. Facilitators can add organizations during session creation, and participants must select their organization when joining.

| Field Name | Data Type | Description |
|-------------------|----------------------|--|
| id | BIGINT (Primary) | Unique identifier for each organization. |
| session_id | BIGINT (Foreign Key) | Refers to the id of the related session in the Sessions table. |
| organization_name | VARCHAR(255) | Full name of the organization. |
| acronym | VARCHAR(50) | Acronym or short name of the organization. |

4. Participants Table

This table links participants to sessions. It stores information about which users are participating in which sessions and their selected organization.

| Field Name | Data Type | Description |
|-----------------|----------------------|--|
| id | BIGINT (Primary) | Unique identifier for each participant entry. |
| user_id | BIGINT (Foreign Key) | Refers to the id of the user in the Users table. |
| session_id | BIGINT (Foreign Key) | Refers to the id of the session in the Sessions table. |
| organization_id | BIGINT (Foreign Key) | Refers to the id of the organization in the Organizations table. |
| role | VARCHAR(100) | Store the participant's role name when they join the session |
| joined_at | TIMESTAMP | Timestamp when the participant joined the session. |

| Field Name | Data Type | Description |
|--------------------|-----------|---|
| session_deleted_at | DATETIME | Records the date and time when a session was marked as deleted by the user. The session remains accessible to others even after deletion by the user. |

5. Decisions Table

This table will store information about important decisions made during a Partos session. Each decision is linked to a session and has details such as the decision name, agreement status, and optional notes provided by the facilitator or participants.

| Field Name | Data Type | Description |
|------------|----------------------|--|
| id | BIGINT (Primary) | Unique identifier for each decision. |
| session_id | BIGINT (Foreign Key) | Refers to the id of the related session in the Sessions table. |
| name | VARCHAR(255) | The name or description of the decision. |
| agree | BOOLEAN | Indicates whether the decision was agreed upon (True / False). |
| notes | TEXT | Additional notes or context regarding the decision. |
| created_at | TIMESTAMP | Timestamp when the decision was created. |
| updated_at | TIMESTAMP | Timestamp when the decision was last updated. |

6. Participant Decisions Table

This table will store individual participant evaluations or ratings of decisions made during a session. Each entry links an organization (via organization_id) to a specific decision (via decision_id) and records the organization's score or evaluation of that decision.

| Field Name | Data Type | Description |
|-----------------|----------------------|---|
| id | BIGINT (Primary) | Unique identifier for each participant's decision evaluation. |
| organization_id | BIGINT (Foreign Key) | Refers to the id of the participant (user) in the Users table. |
| decision_id | BIGINT (Foreign Key) | Refers to the id of the decision in the Decisions table. |
| score | INTEGER | The participant's evaluation or rating of the decision (e.g., on a scale of 1-5). |

| Field Name | Data Type | Description |
|------------|-----------|--|
| desired | BOOLEAN | Indicates whether the score pertains to disagreement discussions. The default value is <code>NULL</code> . |
| created_at | TIMESTAMP | Timestamp when the score was recorded. |
| updated_at | TIMESTAMP | Timestamp when the score was last updated (if applicable). |

7. Participant Comments Table

This table stores individual comments made by participants. Each entry links a user through `user_id` and a session through `session_id`.

| Field Name | Data Type | Description |
|------------|----------------------|--|
| id | BIGINT (Primary) | Unique identifier for each comment. |
| session_id | BIGINT (Foreign Key) | Refers to the <code>id</code> of the related session in the <code>Sessions</code> table. |
| user_id | BIGINT (Foreign Key) | Refers to the <code>id</code> of the user in the <code>Users</code> table. |
| comment | TEXT | Stores the comment made by a participant related to the session. |
| created_at | TIMESTAMP | Timestamp when the comment was recorded. |
| updated_at | TIMESTAMP | Timestamp when the comment was last updated (if applicable). |

API Design

The API for the Power Awareness Tool 2.0 platform will enable communication between the frontend (Next.js) and the backend (Django). It will follow RESTful principles, allowing for efficient data manipulation and retrieval for users, sessions, organizations, decisions, and participant decisions. Below, we'll outline the major API endpoints for managing **users**, **sessions**, **organizations**, **participants**, **decisions**, and **participant decisions**. Each endpoint will return or accept data in **JSON** format.

A. Classes & Utilities

A.1. Custom Pagination Class

Response Structure

```
{
  "current": <current page number>,
  "total": <total items count>,
  "total_page": <total number of pages>,
  "data": <paginated data>
}
```

Class Structure

```
from rest_framework.response import Response
from rest_framework.pagination import PageNumberPagination

class Pagination(PageNumberPagination):
    page_size = 10
    page_size_query_param = "page_size"
    max_page_size = 100

    def get_paginated_response(self, data):
        return Response(
            {
                "current": self.page.number,
                "total": self.page.paginator.count,
                "total_page": self.page.paginator.num_pages,
                "data": data,
            }
        )

    def get_paginated_response_schema(self, schema):
        return {
            "type": "object",
            "properties": {
                "current": {
                    "type": "integer",
                    "example": 123,
                },
                "total": {
                    "type": "integer",
```

```
        "example": 123,
      },
      "total_page": {
        "type": "integer",
        "example": 123,
      },
      "data": schema,
    },
  }
}
```

1. User API

1.1. Create User (Sign Up)

Register a new user on the platform.

Endpoint: `POST /api/v1/register/`

Request Body:

```
{
  "full_name": "John Doe",
  "gender": 2,
  "country": "ID",
  "account_purpose": 1,
  "email": "john.doe@example.com",
  "password": "Test#123",
  "confirm_password": "Test#123",
}
```

Response: 201 Created on success. A verification email is sent.

```
{
  "id": 1,
  "full_name": "John Doe",
  "email": "john.doe@example.com",
  "gender": 2,
  "country": "ID",
  "account_purpose": 1,
  "is_superuser": false
}
```



```
}
```

1.2. Verify User

Verify the user's email using the verification link.

Endpoint: `GET /api/v1/verify?code={code}`

Response: **200 OK** if successful.

```
{
  "message": "Email verified successfully."
}
```

1.3. User Login

This API endpoint allows users to log in to the platform using their email and password credentials. Upon successful authentication, a session is established for the user.

Endpoint: `POST /api/v1/users/login/`

Request Body:

```
{
  "email": "john.doe@example.com",
  "password": "password123"
}
```

Response: **200 OK** on success, with **JWT token for session authentication**.

```
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "expiration_time": "2024-10-22T18:06:11Z",
  "user": {
    "id": 1,
    "full_name": "John Doe",
    "email": "john.doe@example.com",
    "gender": 1,
    "country": "ID",
    "account_purpose": 1,
    "is_superuser": false
  }
}
```

```
}
```

1.4. User Forgot password

This API endpoint allows users to request a password change. Users can initiate the process by providing their registered email address, and a password reset link or instructions will be sent to them.

Endpoint: `POST /api/v1/users/forgot-password/`

Request Body:

```
{
  "email": "john.doe@example.com"
}
```

Response: **200 OK** on success

1.5. Verify password code

This API endpoint verifies the token provided during the password reset process. It validates that the token belongs to the correct user who requested the password reset, ensuring the authenticity of the request before proceeding.

Endpoint: `GET /api/v1/users/verify-password-code?token=secret`

Response: **200 OK** on success

```
{
  "message": "OK"
}
```

1.6. User Reset Password

This API endpoint allows users to update their old password with a new one. A valid token is required to authorize the password reset, ensuring the request is tied to the correct user.

Endpoint: `POST /api/v1/users/reset-password?token=secret`

Request Body:

```
{
  "password": "NewPass123",
  "confirm_password": "NewPass123"
}
```

Response: 200 OK on success

```
{
  "message": "Password reset successfully"
}
```

2. Session API

2.1. Create Session

This API endpoint allows facilitators to create a new Power Awareness Tool (PAT) session.

Endpoint: `POST /api/v1/sessions/`

Request Body

```
{
  "session_name": "Partnership Evaluation - Health Sector",
  "countries": ["NL", "KE"],
  "purpose": 1,
  "date": "2024-09-15",
  "context": "Evaluating the partnership dynamics.",
  "organizations": [
    { "name": "Akvo", "acronym": "AKV" },
    { "name": "Partos", "acronym": "PTS" }
  ]
}
```

Response: 201 Created on success. Return join code

```
{
  "id": 101,
  "session_name": "Partnership Evaluation - Health Sector",
  "join_code": "ABCDE12345",
}
```

```
"is_published": false
}
```

2.2. Get / Put Session by Session ID

This API endpoint allows users to retrieve or update the details of a specific session using its `session_id`. Access is restricted to the session owner or participants. If the request comes from a non-owner or non-participant, a "Forbidden" response will be returned.

Endpoint: GET, PUT `/api/v1/sessions?id={session_id}`

JWT Access: Owner & Participant

Response: 200 OK on success.

```
{
  "id": 101,
  "session_name": "Partnership Evaluation - Health Sector",
  "facilitator": {
    "id": 1,
    "full_name": "John Doe"
  },
  "organizations": [{
    "id": 1,
    "name": "Akvo"
    "acronym": "AKV"
  }, {
    "id": 2,
    "name": "Partos"
    "acronym": "PTS"
  }],
  "countries": ["Netherlands", "Kenya"],
  "purpose": 2,
  "context": "Evaluating the partnership dynamics.",
  "date": "2024-10-22",
  "join_code": "ABCDE12345",
  "is_published": false,
  "notes": "string",
  "summary": "string",
  "created_at": "2024-09-10T12:30:45Z",
  "updated_at": null,
}
```

```
"closed_at": null
}
```

2.4 List Session

Filter params

| Parameter | Type | Description |
|-----------|---------|--|
| published | BOOLEAN | Filters sessions based on their status: - False : Retrieve all active sessions (default). - True : Retrieve all closed sessions. |
| role | ENUM | Filters the session list based on the user's role: 1 = Facilitated 2 = Participated |
| search | STRING | Filter session by session_name or context. |

Retrieve a list of sessions.

Endpoint: `GET /api/v1/sessions`

JWT Access: Owner & Participant

```
{
  current: 1,
  total: 1,
  total_page: 1,
  data: [{
    "id": 101,
    "session_name": "Partnership Evaluation - Health Sector",
    "facilitator": {
      "id": 1,
      "full_name": "John Doe"
    },
    "date": "2024-10-22",
    "context": "Evaluating the partnership dynamics.",
    "created_at": "2024-10-22T03:48:43.612Z",
    "updated_at": "2024-10-22T03:48:43.612Z",
    "closed_at": "2024-10-22T03:48:43.612Z",
    "is_owner": true
  }]
```

```
}]
]
```

2.5. Publish Session (Finalise)

Publish and close the session, making it read-only.

Endpoint: `PUT /api/v1/sessions?id={session_id}`

JWT Access: Owner

Response: 200 OK on success.

```
{
  "id": 101,
  "session_name": "Partnership Evaluation - Health Sector",
  "is_published": true,
  "closed_at": "2024-09-20T14:45:30Z"
}
```

3. Participant API

3.1. Get Organisation and Country List by Session Code

Retrieve the organization list on a session.

Endpoint: `GET /api/v1/sessions?code={join_code}`

Response: 200 OK on Success

```
{
  "id": 101,
  "session_name": "Partnership Evaluation - Health Sector",
  "organizations": [{
    "id": 1,
    "name": "Akvo"
    "acronym": "AKV"
  },{
    "id": 2,
```

```
"name": "Partos"
"acronym": "PTS"
}],
"join_code": "FG-HIJKL-12345"
}
```

3.2. Join Session

A participant joins a session by entering a join code.

Endpoint: `POST /api/v1/participants/join/`

Request Body:

```
{
  "role": "MnE Manager",
  "session_id": 1,
  "organization_id": 1
}
```

Response: 201 Created on success.

3.3. List participants

Retrieve the participants list on a session.

Endpoint: `GET /api/v1/session/{session_id}/participants`

Response: 200 OK on Success

```
[
  {
    "id": 41,
    "full_name": "Jane Doe",
    "email": "string",
    "role": "IT Support",
    "organization_name": "Akvo Foundation",
    "organization_acronym": "AKVO",
    "organization_id": 111
  },
  {
    "id": 42,
```

```
"full_name": "John Doe",
"email": "john.doe@example.com",
"role": "MnE Manager",
"organization_name": "Partos Global",
"organization_acronym": "PARTOS",
"organization_id": 112
},
]
```

5. Decision API

5.1. Init Multiple Decisions

Add multiple decisions to a session.

Endpoint: `POST /api/v1/decisions/`

Request Body:

```
{
  "session_id": 103,
  "decisions": [
    "Decision on budget allocation",
    "Decision on resource distribution"
  ]
}
```

Response: 201 Created on success. Returns an array of the created decisions with their `id` and other relevant details.

```
[
  {
    "id": 501,
    "session_id": 103,
    "name": "Decision on budget allocation",
    "notes": null,
    "agree": false,
    "created_at": "2024-09-13T10:00:00Z"
  },
]
```



```
{
  "id": 502,
  "session_id": 103,
  "name": "Decision on resource distribution",
  "notes": null,
  "agree": false,
  "created_at": "2024-09-13T10:05:00Z"
}
```

5.2. Submit Decision Scores for Multiple Decisions

This API endpoint allows facilitators to submit scores for multiple decisions. Facilitators can provide a score for each `organization_id` along with a corresponding `decision_id`. Additionally, this endpoint includes a conditional `desired` field that flags scores related to disagreement decisions.

Endpoint: `POST /api/v1/participant-decisions/`

Request Body:

```
{
  "session_id": 101,
  "scores": [
    {
      "organization_id": 21,
      "decision_id": 1,
      "score": 1,
    },
    {
      "organization_id": 21,
      "decision_id": 1,
      "score": 3,
      "desired": true
    },
  ]
}
```

Response: 201 Created on success.

5.3. Update Decision Scores

This API endpoint allows facilitators to update the score for a specific decision. The id parameter is required to identify the decision being updated. Facilitators can modify the score and other relevant fields as necessary.

Endpoint: `PUT /api/v1/participant-decisions/`

Request Body:

```
{
  "session_id": 101,
  "scores": [
    {
      "id": 333,
      "agree": false,
      "notes": "We need to do this and that"
    },
    {
      "id": 334,
      "agree": true,
    },
  ]
}
```

Response: **200 OK** on success.

6. User management API

6.1. List users

This API endpoint retrieves a list of all users. Access is restricted to admin users and requires a valid JWT token for authentication.

Endpoint: `GET /api/v1/admin/users`

JWT Access: Admin

Response: **200 OK** on Success

```
{
  "current": 1,
  "total": 50,
  "total_page": 10,
```

```
"data": [  
  {  
    "id": 111,  
    "full_name": "John DOe",  
    "email": "user@example.com",  
    "gender": 1,  
    "country": "ID",  
    "account_purpose": 1,  
    "is_superuser": false  
  }  
]
```

6.2. Update user role

This API endpoint allows admin users to update a user's role by setting the `is_superuser` boolean value. The possible values are:

- `is_superuser` = True: Grants admin privileges to the user.
- `is_superuser` = False: Removes admin privileges from the user.

Access requires a valid JWT token for authentication.

Endpoint: `PUT /api/v1/admin/user/{id}/`

Request Body:

```
{  
  "is_superuser": true  
}
```

Response: **200 OK** on success.

6.3. Delete user

This API endpoint allows admin users to delete a specified user from the system. Access requires a valid JWT token for authentication.

Endpoint: `DELETE /api/v1/admin/user/{id}/`

JWT Access: Admin

Response: **204 No response body** on success.

7. Statistics API

7.1. Number of users

This API endpoint returns the total number of users in the system. The data reflects the count from the last 30 days. Access requires a valid JWT token for admin authentication.

Endpoint: GET /api/v1/admin/statistics/users

JWT Access: Admin

Response: 200 OK on Success

```
{
  "total_users": 34,
  "total_users_last_30_days": 26,
  "total_users_per_account_purpose": [
    {
      "account_purpose": "purposeOfAccount1",
      "total": 4
    },
    {
      "account_purpose": "purposeOfAccount2",
      "total": 8
    },
    {
      "account_purpose": "purposeOfAccount3",
      "total": 4
    }
  ]
}
```

7.2. Number of completed PAT Sessions

This API endpoint retrieves the total number of completed Power Awareness Tool (PAT) sessions within the last 30 days. Access requires a valid JWT token for admin authentication.

Endpoint: GET /api/v1/admin/sessions/completed

JWT Access: Admin

Response: 200 OK on Success

```
{
  "total_completed": 31,
  "total_completed_last_30_days": 7
}
```

7.3 Number of created PAT sessions in last 3 years

This API endpoint returns the total number of PAT sessions created in the last three years. Access requires a valid JWT token for admin authentication.

Endpoint: `GET /api/v1/admin/sessions/per-last-3-years`

JWT Access: Admin

Response: 200 OK on Success

```
[
  {
    "total_sessions": [2,2,4,2,2,5,2,1,3,3,1,3],
    "year": 2022
  },
  {
    "total_sessions": [1,1,4,3,2,0,2,4,2,2,7,2],
    "year": 2023
  },
  {
    "total_sessions": [1,5,3,3,2,3,4,5,10,8,3,2],
    "year": 2024
  }
]
```

Revision #27

Created 5 September 2024 03:00:26 by Deden Bangkit

Updated 8 January 2025 02:59:40 by Iwan Firmawan