Documentation for Blockchain Application using Flask

Overview

This Python application implements a simple blockchain for task management. The blockchain ensures transparency and security in group-based task management, including features for creating groups, enrolling participants, creating tasks, approving tasks, and rewarding participants. It uses Flask to expose RESTful APIs for interacting with the blockchain.

Main Components

1. Block Class

Represents an individual block in the blockchain.

Attributes:

- `index`: The position of the block in the chain.
- `timestamp`: Time of block creation.
- `data`: Contains the task or other relevant information.
- `previous_hash`: Hash of the previous block in the chain.
- `nonce`: A number used for mining.
- `hash`: The unique hash of the block.

Methods:

- `calculate_hash`: Computes the hash of the block based on its contents.
- `mine_block`: Adjusts the `nonce` until the block's hash satisfies the required difficulty level.

---

2. Blockchain Class

Manages the entire blockchain and its associated functionality.

Attributes:

- `chain`: List of all blocks in the blockchain.
- `difficulty`: Mining difficulty (number of leading zeros required in the hash).
- `pending_tasks`: Tasks waiting to be added to the blockchain.
- `groups`: A dictionary to manage groups, participants, and tasks.
- `reputation_threshold`: Minimum reputation required to approve tasks.
- `approval_threshold`: Percentage of approvals required to mine a task.
- `reputation`: Tracks reputation points for participants.
- `rewards`: Tracks rewards (e.g., coins or points) for participants.

Methods:

- `create_genesis_block`: Creates the first block (Genesis Block).
- `get_latest_block`: Retrieves the last block in the chain.
- `create_group`: Adds a new group to the system.
- `enroll_in_group`: Enrolls a participant in a specific group.
- `create_task`: Adds a task to a group and assigns reputation points to the creator.
- `approve_task`: Allows participants to approve tasks if they meet the reputation threshold.
- `mine_pending_task`: Mines a task and adds it to the blockchain.
- `reward_users`: Rewards participants for creating and approving tasks.
- `is_chain_valid`: Validates the integrity of the blockchain.

---

3. Flask API Endpoints

Exposes RESTful APIs to interact with the blockchain.

Endpoints:

1. **GET /chain**
   - Returns the entire blockchain.
   - Response: JSON containing the chain and its length.
2. **GET /groups**
   - Retrieves all groups and their details.
   - Response: JSON containing group details.
3. **GET /reputation**
   - Returns the reputation points of participants.

   - Response: JSON with reputation data.
4. **GET /rewards**

   - Retrieves the rewards for participants.
   - Response: JSON with reward details.
5. **POST /create_group**

   - Creates a new group.
   - Request Body: `{ "group_name": "GroupName" }`
   - Response: Success or failure message.
6. **POST /enroll_in_group**

   - Enrolls a participant in a group.
   - Request Body: `{ "group_name": "GroupName", "participant": "UserName" }`
   - Response: Success or failure message.
7. **POST /create_task**

   - Creates a task in a group.
   - Request Body: `{ "group_name": "GroupName", "id": "TaskID", "description": "TaskDescription", "creator": "CreatorName" }`
   - Response: Success or failure message.
8. **POST /approve_task**

   - Approves a task by a participant.
   - Request Body: `{ "group_name": "GroupName", "task_id": "TaskID", "participant": "UserName" }`
   - Response: Success or failure message.
9. **GET /validate_chain**

   - Validates the entire blockchain.
   - Response: JSON indicating if the chain is valid.

Key Features

1. **Task Management**

   - Create and manage tasks within groups.
   - Tasks can be approved by participants based on reputation.
2. **Reputation System**

    - Participants earn reputation points for creating and approving tasks.
    - Reputation affects the ability to approve tasks.
3. **Rewards System**

    - Participants are rewarded with coins or points for task-related actions.
4. **Blockchain Security**

    - Tasks are added to the blockchain only after meeting approval thresholds.
    - Each block is mined to ensure integrity.
5. **Group Management**

    - Flexible group creation and participant enrollment.

Running the Application

Install the required packages:

```
pip install flask
```

1.

Run the application:

```
python app.py
```

2.
3. Access the APIs via `http://127.0.0.1:5000`.

Example Usage

Create a Group:

```
curl -X POST -H "Content-Type: application/json" -d '{"group_name": "DevTeam"}'
http://127.0.0.1:5000/create_group
```

1.

Enroll in a Group:

```
curl -X POST -H "Content-Type: application/json" -d '{"group_name": "DevTeam", "participant": "Alice"}' http://127.0.0.1:5000/enroll_in_group
```

2.

Create a Task:

```
curl -X POST -H "Content-Type: application/json" -d '{"group_name": "DevTeam", "id": "1", "description": "Fix bug", "creator": "Alice"}' http://127.0.0.1:5000/create_task
```

3.

Approve a Task:

```
curl -X POST -H "Content-Type: application/json" -d '{"group_name": "DevTeam", "task_id": "1", "participant": "Bob"}' http://127.0.0.1:5000/approve_task
```

4.

Validate the Blockchain:

```
curl http://127.0.0.1:5000/validate_chain
```

5.

Conclusion

This application combines blockchain technology with task management, ensuring secure, transparent, and fair collaboration in group environments. It provides a strong foundation for decentralized task management systems with a robust reputation and rewards mechanism.