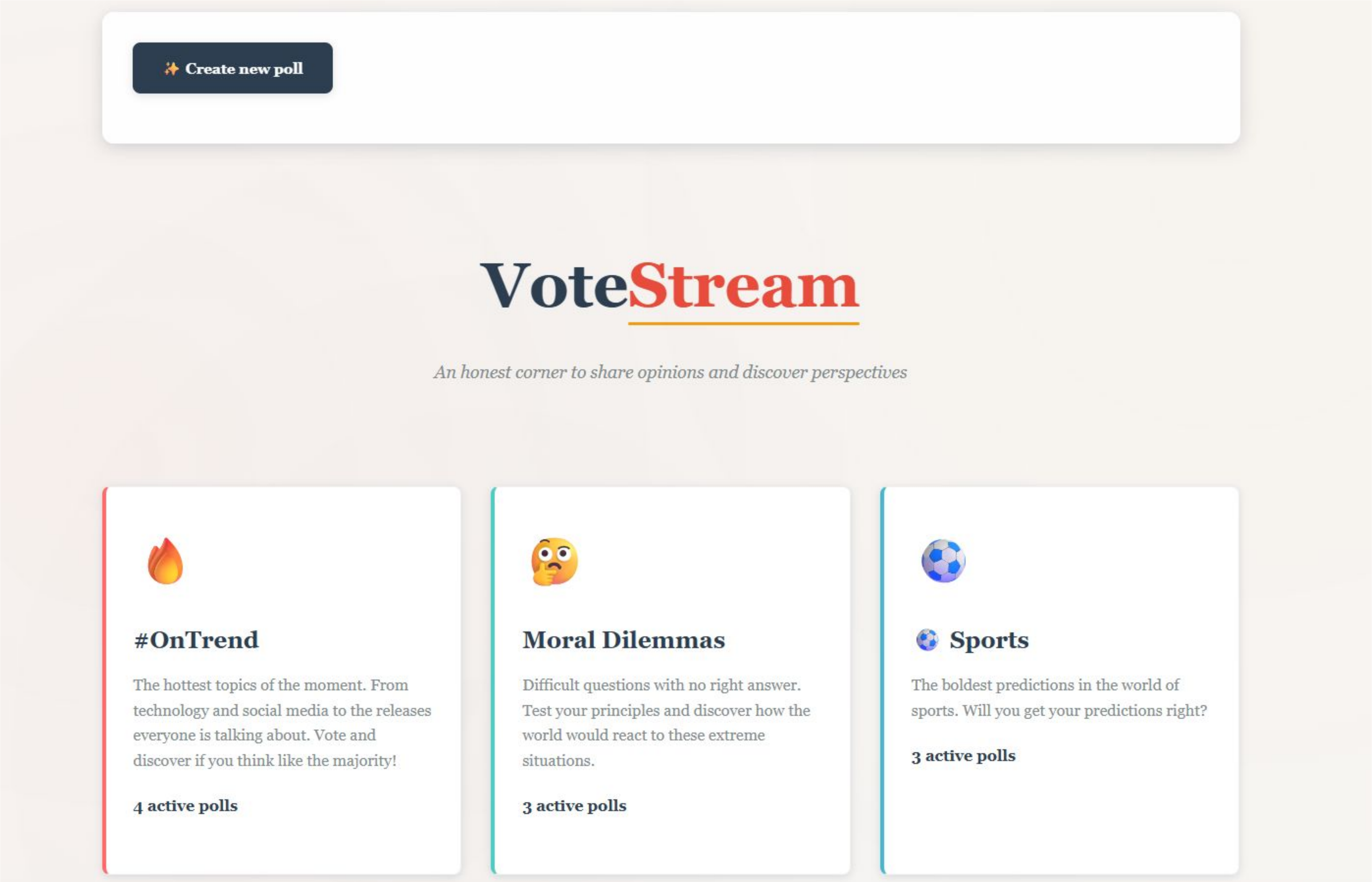
VoteStream: A Scalable Prototyping Project



VoteStream

An honest corner to share opinions and discover perspectives

🔥

#OnTrend

The hottest topics of the moment. From technology and social media to the releases everyone is talking about. Vote and discover if you think like the majority!

4 active polls

🤔

Moral Dilemmas

Difficult questions with no right answer. Test your principles and discover how the world would react to these extreme situations.

3 active polls

⚽

⚽ Sports

The boldest predictions in the world of sports. Will you get your predictions right?

3 active polls

Technische Universität Berlin

What is VoteStream?

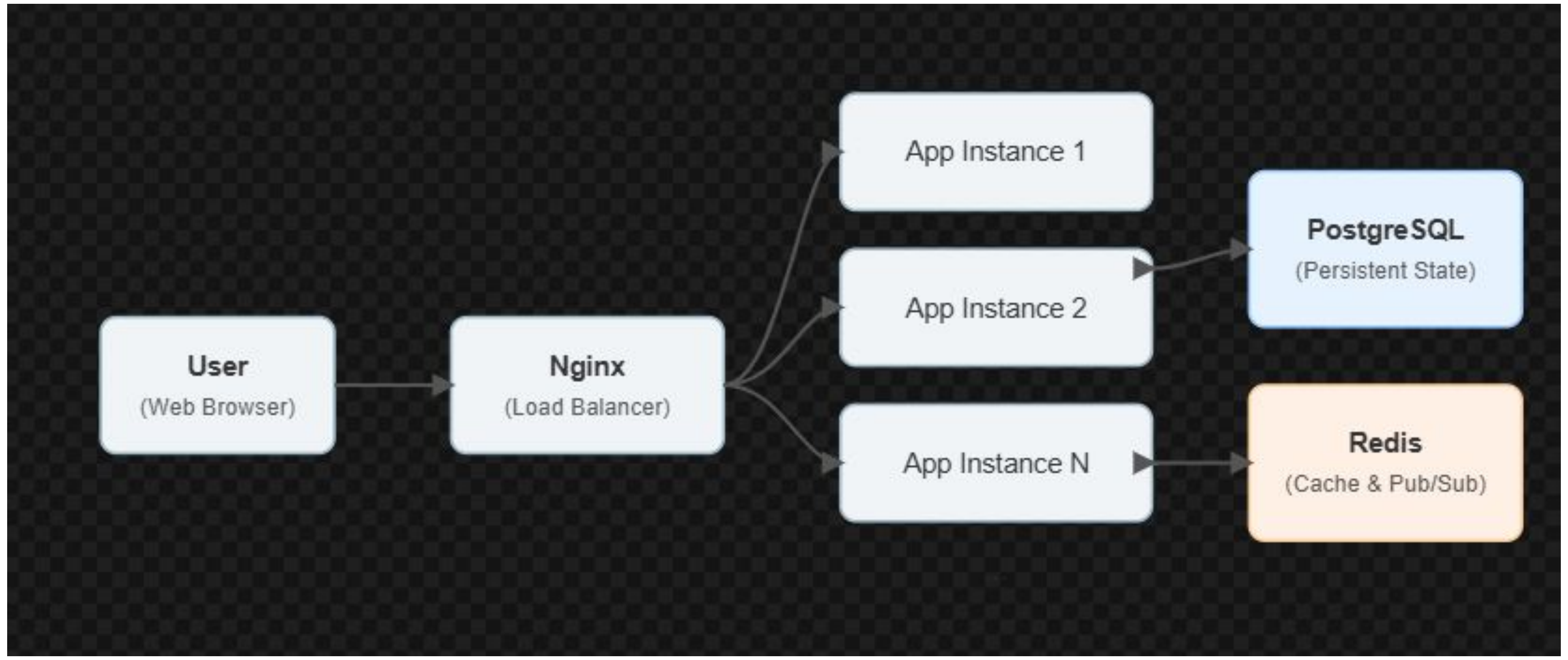
- VoteStream is a real-time, scalable polling application.

- Our motivation was to create a platform where people can genuinely express their opinions and see what the community really thinks about trending topics, moral dilemmas, and more.

- The platform provides instant feedback, with results updating live for everyone as votes are cast.

Technische Universität Berlin

System Architecture



Our system uses a decoupled, stateless architecture for high scalability. The main components are:

·**FastAPI:** The core web framework for the application.

·**PostgreSQL:** Handles the persistent storage of all polls and votes.

·**Redis:** Serves a dual role for caching to speed up responses and for real-time messaging via Pub/Sub.

·**Nginx:** Acts as the reverse proxy and load balancer, distributing traffic to the application instances.

Technische
Universität
Berlin

Requirement 1: State Management

Requirement: "Your application must manage some kind of state."

Our Solution:

- All application state (polls, options, votes) is durably persisted in a PostgreSQL database.

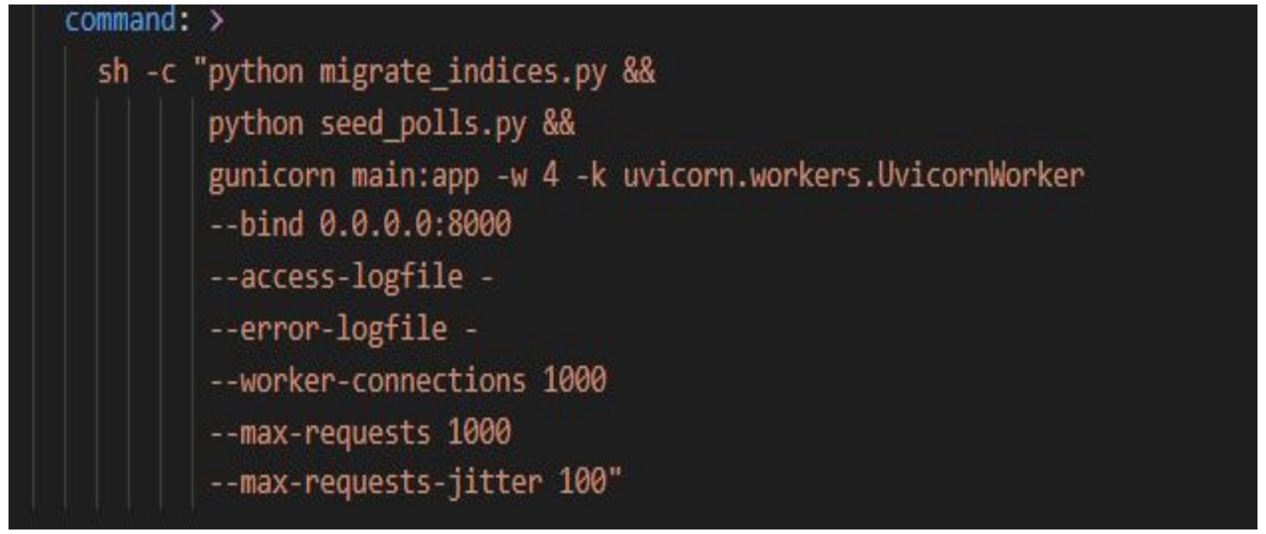
- We use SQLModel for clear and robust data mapping.

Requirement 2: Horizontal & Vertical Scaling

Requirement: "Your application needs to be able to scale vertically and horizontally."

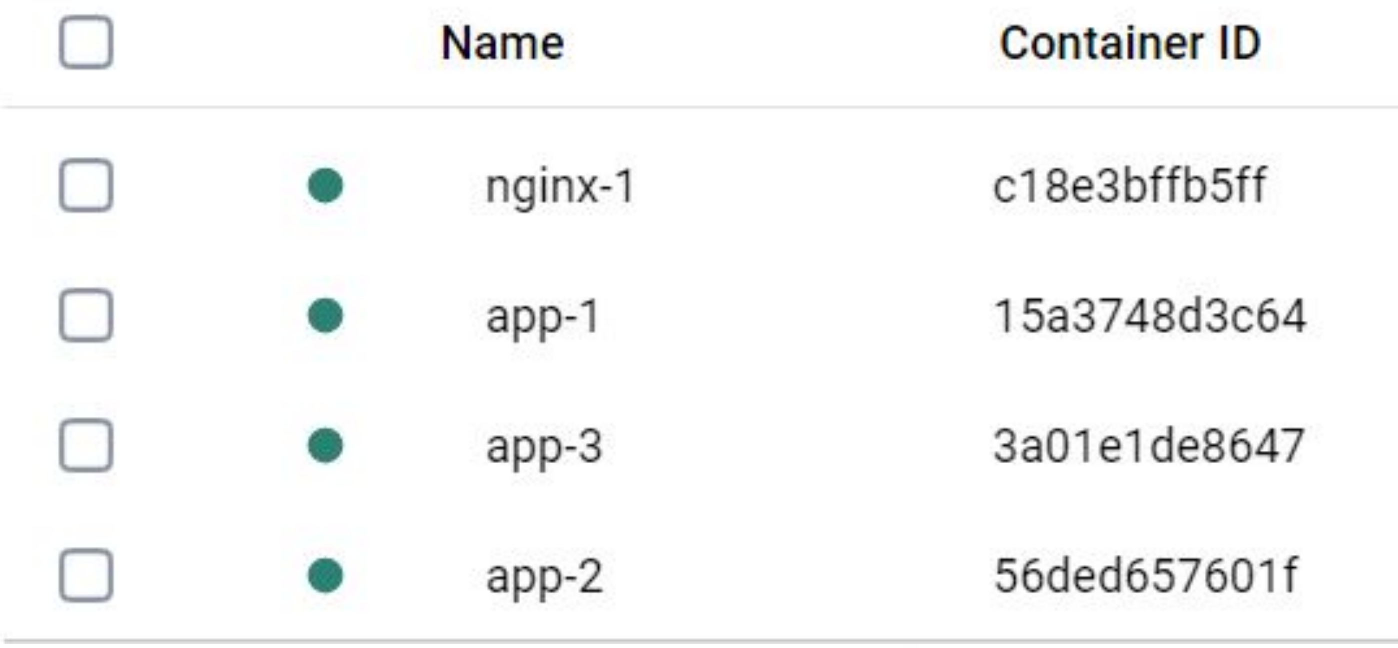
Our Solution:

·Vertical Scaling: Achieved by running the app with multiple Gunicorn workers, allowing it to use multiple CPU cores.



```
command: >
  sh -c "python migrate_indices.py &&
         python seed_polls.py &&
         gunicorn main:app -w 4 -k uvicorn.workers.UvicornWorker
         --bind 0.0.0.0:8000
         --access-logfile -
         --error-logfile -
         --worker-connections 1000
         --max-requests 1000
         --max-requests-jitter 100"
```

·Horizontal Scaling: Enabled by our stateless application design. As you can see, we can run multiple container instances behind the Nginx load balancer.



□		Name	Container ID
□	●	nginx-1	c18e3bffb5ff
□	●	app-1	15a3748d3c64
□	●	app-3	3a01e1de8647
□	●	app-2	56ded657601f

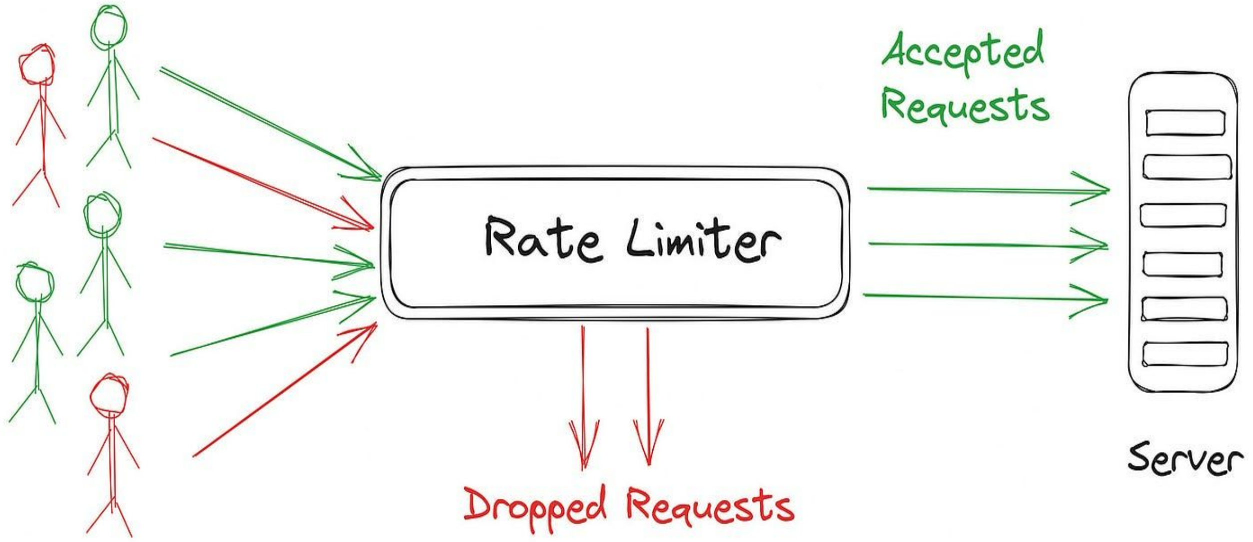
Requirement 3: Overload Mitigation

Requirement: "When your application is scaled up/out, it should not be possible to overload another component."

Our Solution:

·Rate Limiter: A custom middleware prevents abuse by limiting requests per client.

·Circuit Breaker: A custom-built pattern protects our database and cache from cascading failures during high stress or service outages.



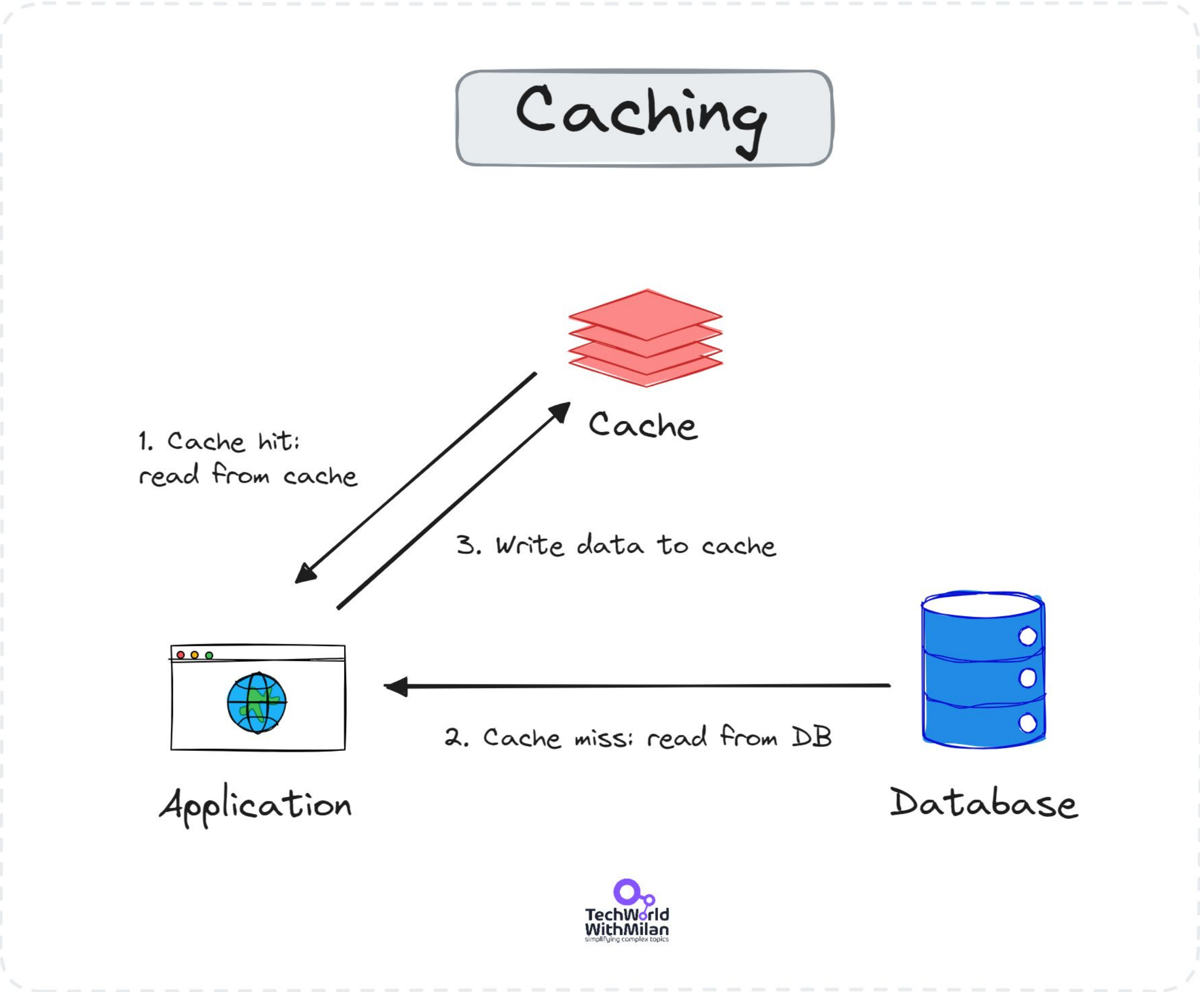
Requirement 4: Two Additional Strategies

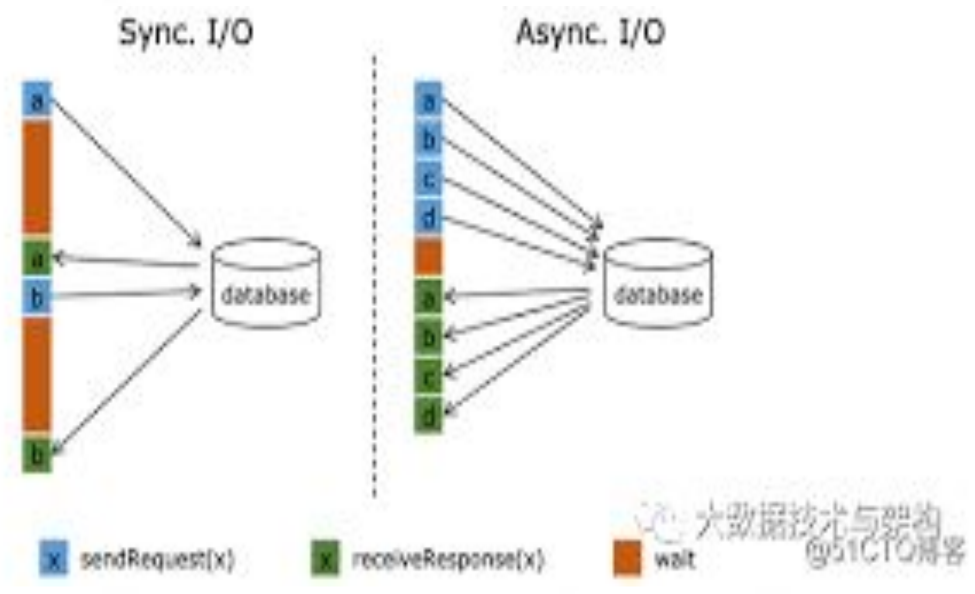
Requirement: "Implement two more strategies...covered either in the lecture content or during the presentations."

Our Solution:

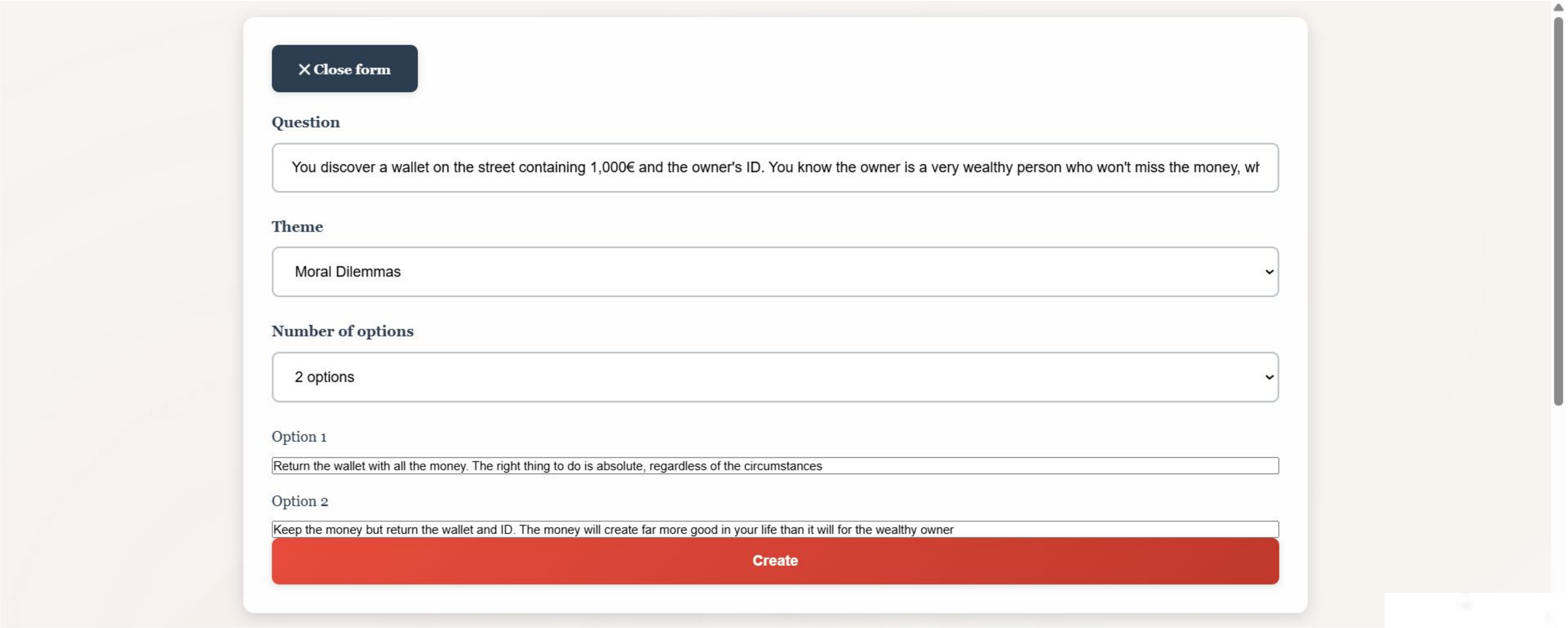
·Caching: We use Redis extensively as a cache-aside layer to reduce database load and serve frequent requests from memory.

·Asynchronous I/O: The entire backend is built with FastAPI, an async framework that handles thousands of concurrent connections (like WebSockets) efficiently without blocking.





Creating a Poll – State Management



Question

You discover a wallet on the street containing 1,000€ and the owner's ID. You know the owner is a very wealthy person who won't miss the money, wh

Theme

Moral Dilemmas

Number of options

2 options

Option 1

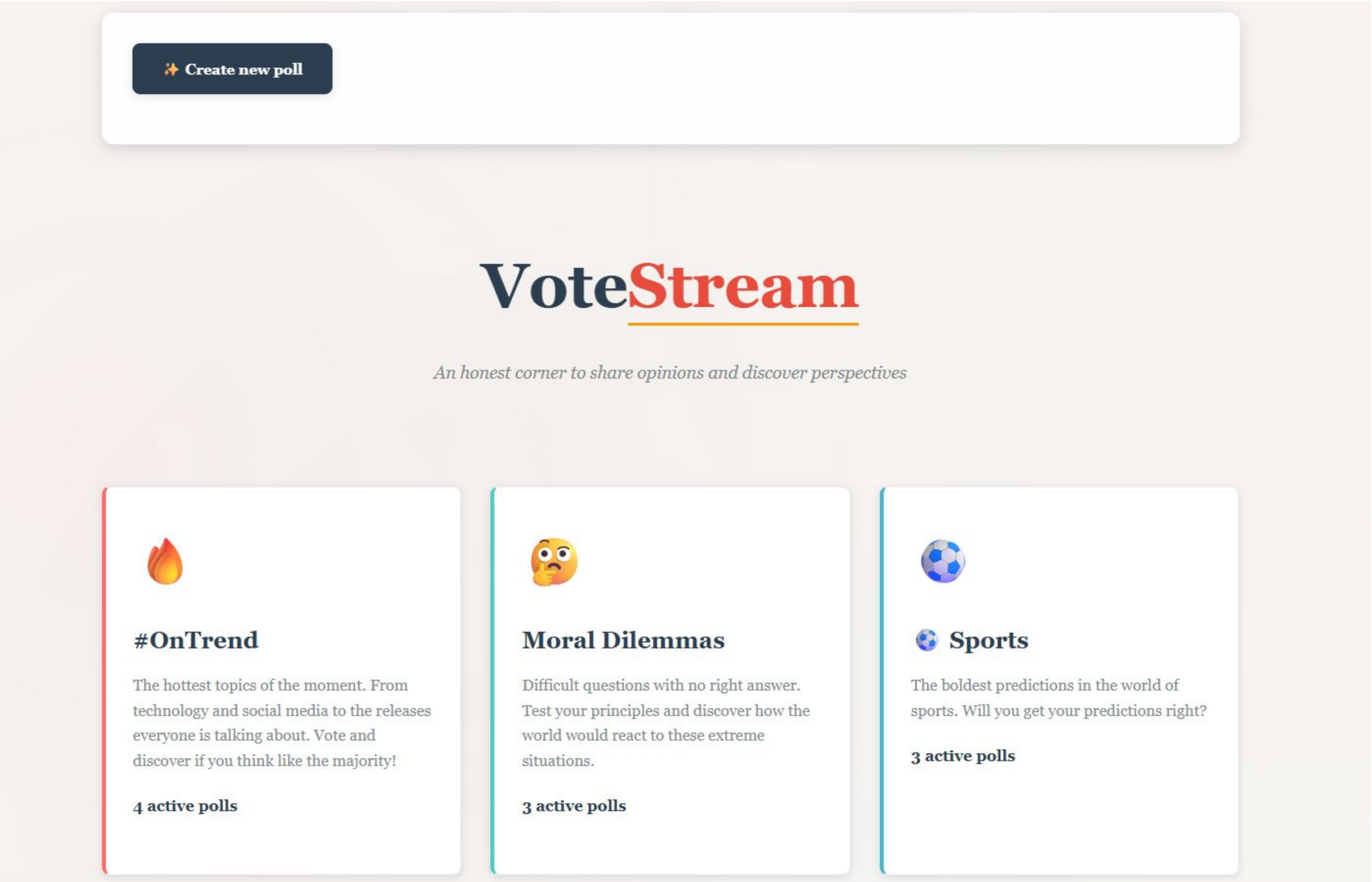
Return the wallet with all the money. The right thing to do is absolute, regardless of the circumstances

Option 2

Keep the money but return the wallet and ID. The money will create far more good in your life than it will for the wealthy owner

- The user can create a new poll from the frontend.
- This action is sent to the backend, which stores it in the PostgreSQL database using SQLModel.
- This demonstrates persistent state management.

Voting Live – Real-Time Updates



VoteStream

An honest corner to share opinions and discover perspectives

🔥

#OnTrend

The hottest topics of the moment. From technology and social media to the releases everyone is talking about. Vote and discover if you think like the majority!

4 active polls

🤔

Moral Dilemmas

Difficult questions with no right answer. Test your principles and discover how the world would react to these extreme situations.

3 active polls

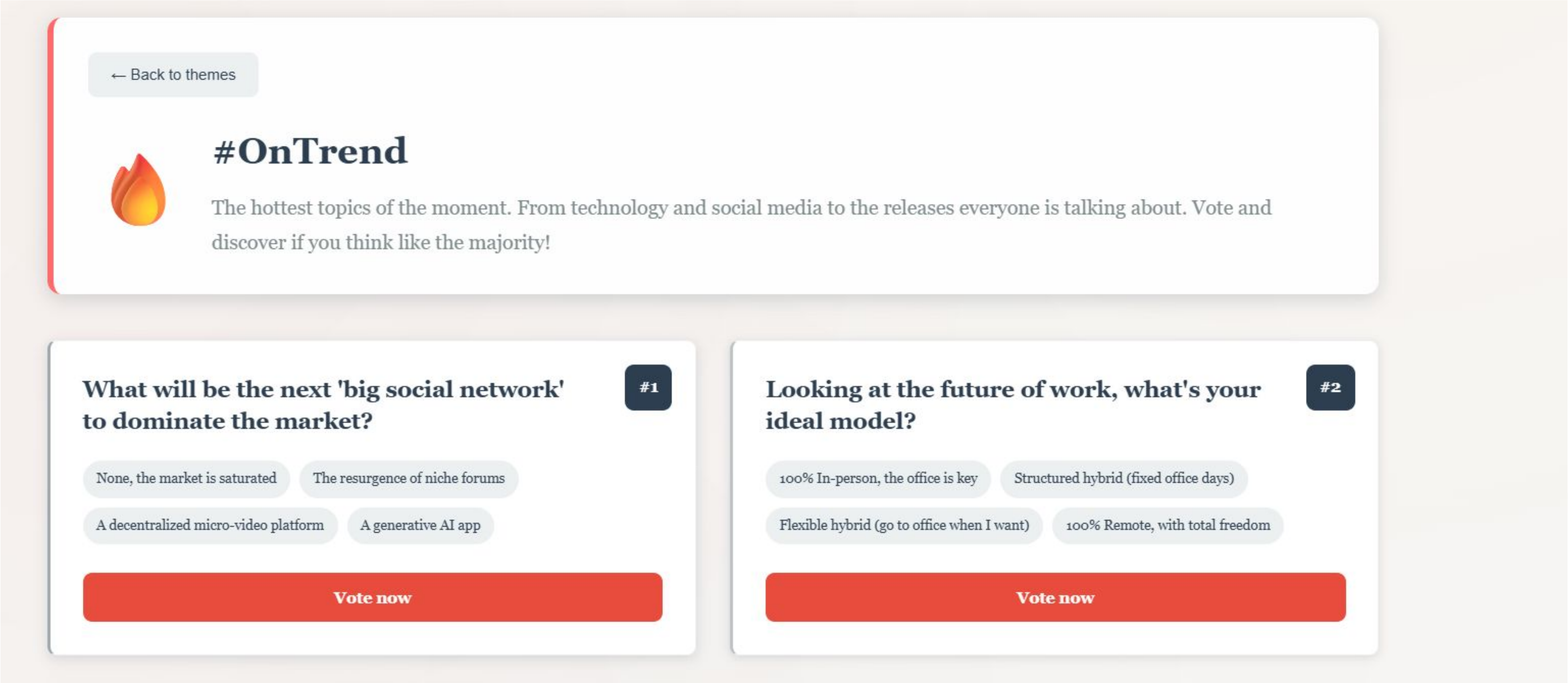
⚽

⚽ Sports

The boldest predictions in the world of sports. Will you get your predictions right?

3 active polls

Voting Live – Real-Time Updates



← Back to themes

🔥 **#OnTrend**

The hottest topics of the moment. From technology and social media to the releases everyone is talking about. Vote and discover if you think like the majority!

What will be the next 'big social network' to dominate the market? #1

- None, the market is saturated
- The resurgence of niche forums
- A decentralized micro-video platform
- A generative AI app

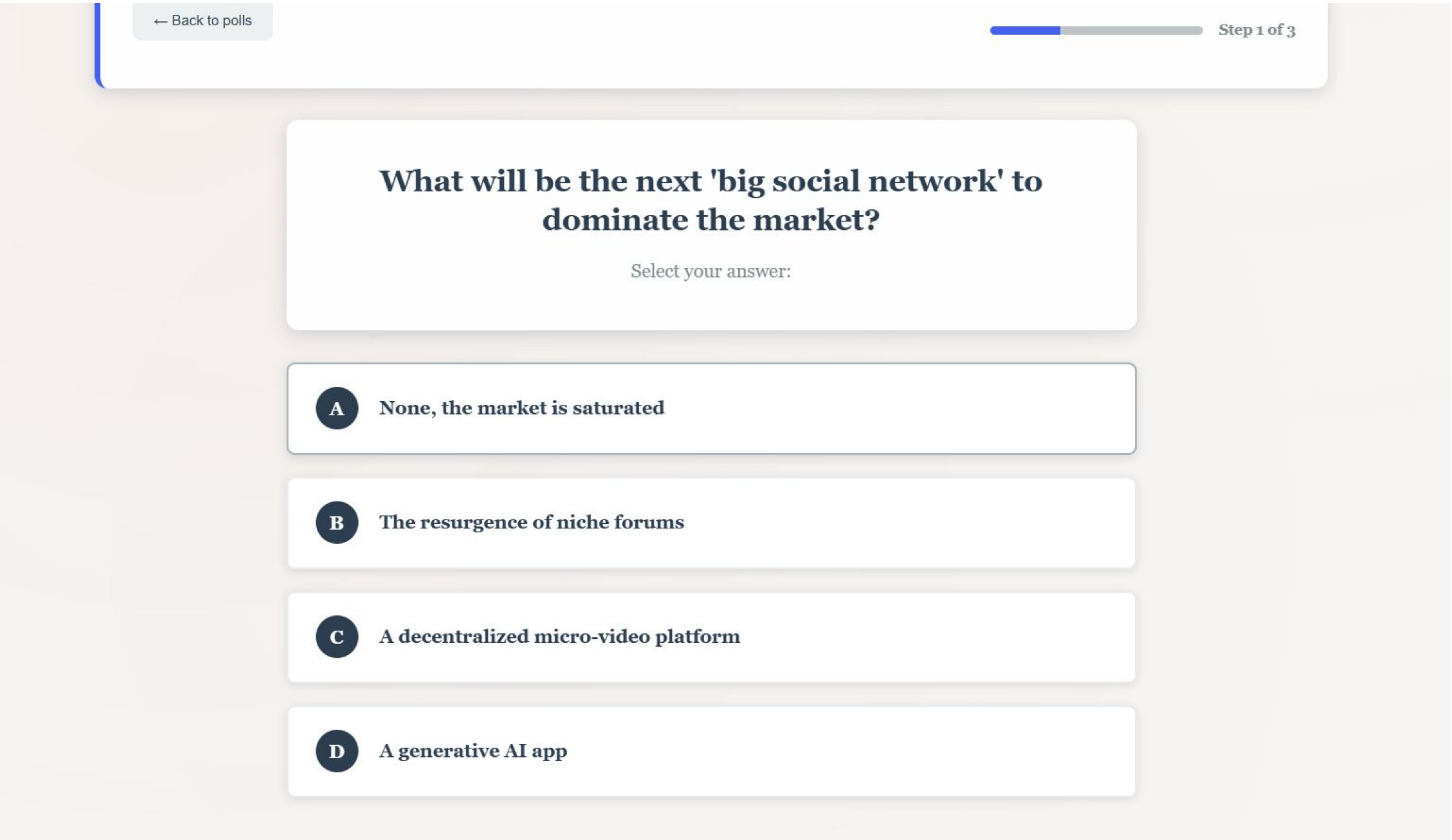
Vote now

Looking at the future of work, what's your ideal model? #2

- 100% In-person, the office is key
- Structured hybrid (fixed office days)
- Flexible hybrid (go to office when I want)
- 100% Remote, with total freedom

Vote now

Voting Live – Real-Time Updates



What will be the next 'big social network' to dominate the market?

Select your answer:

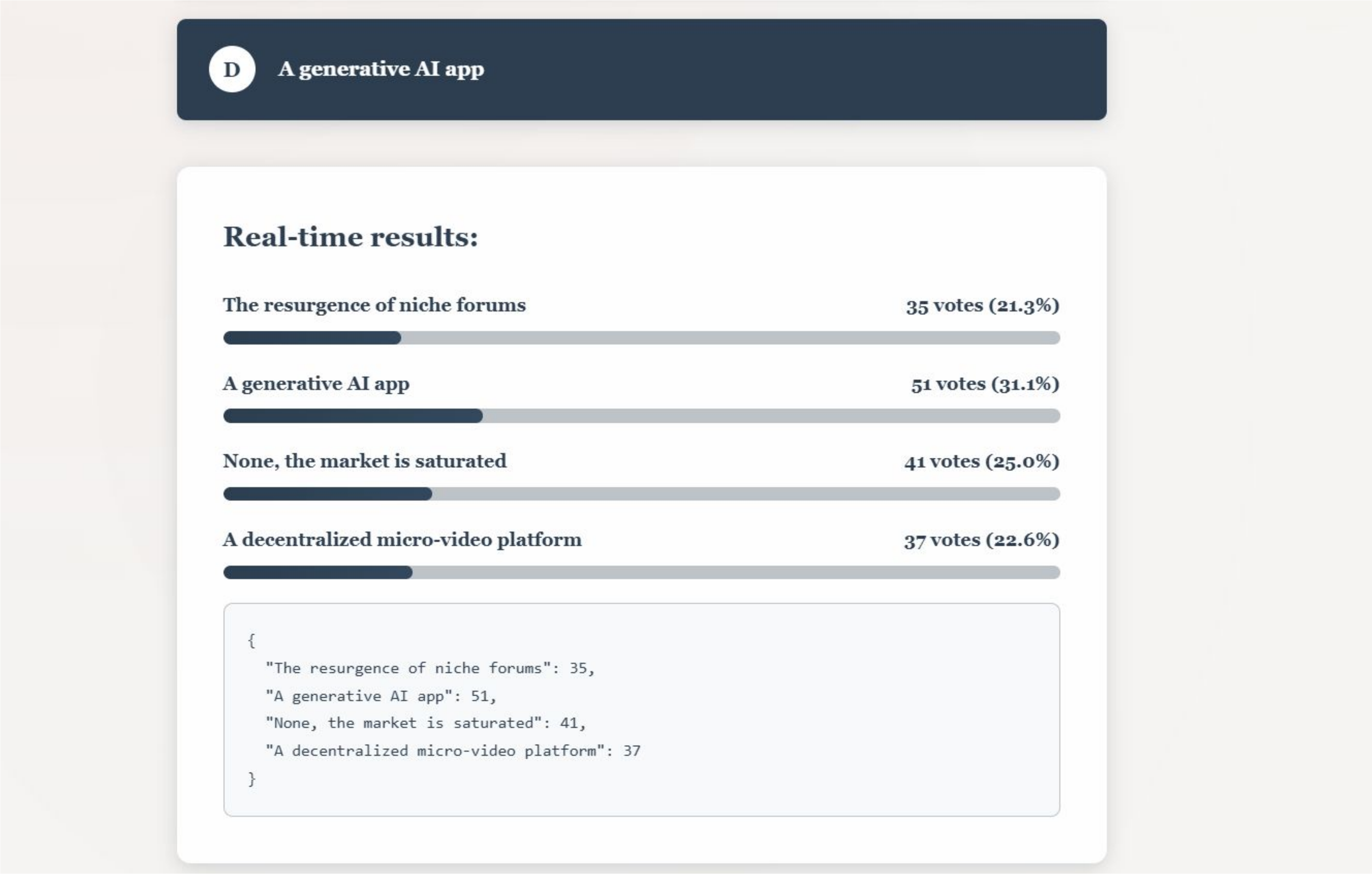
A None, the market is saturated

B The resurgence of niche forums

C A decentralized micro-video platform

D A generative AI app

Voting Live – Real-Time Updates



D A generative AI app

Real-time results:

The resurgence of niche forums 35 votes (21.3%)

A generative AI app 51 votes (31.1%)

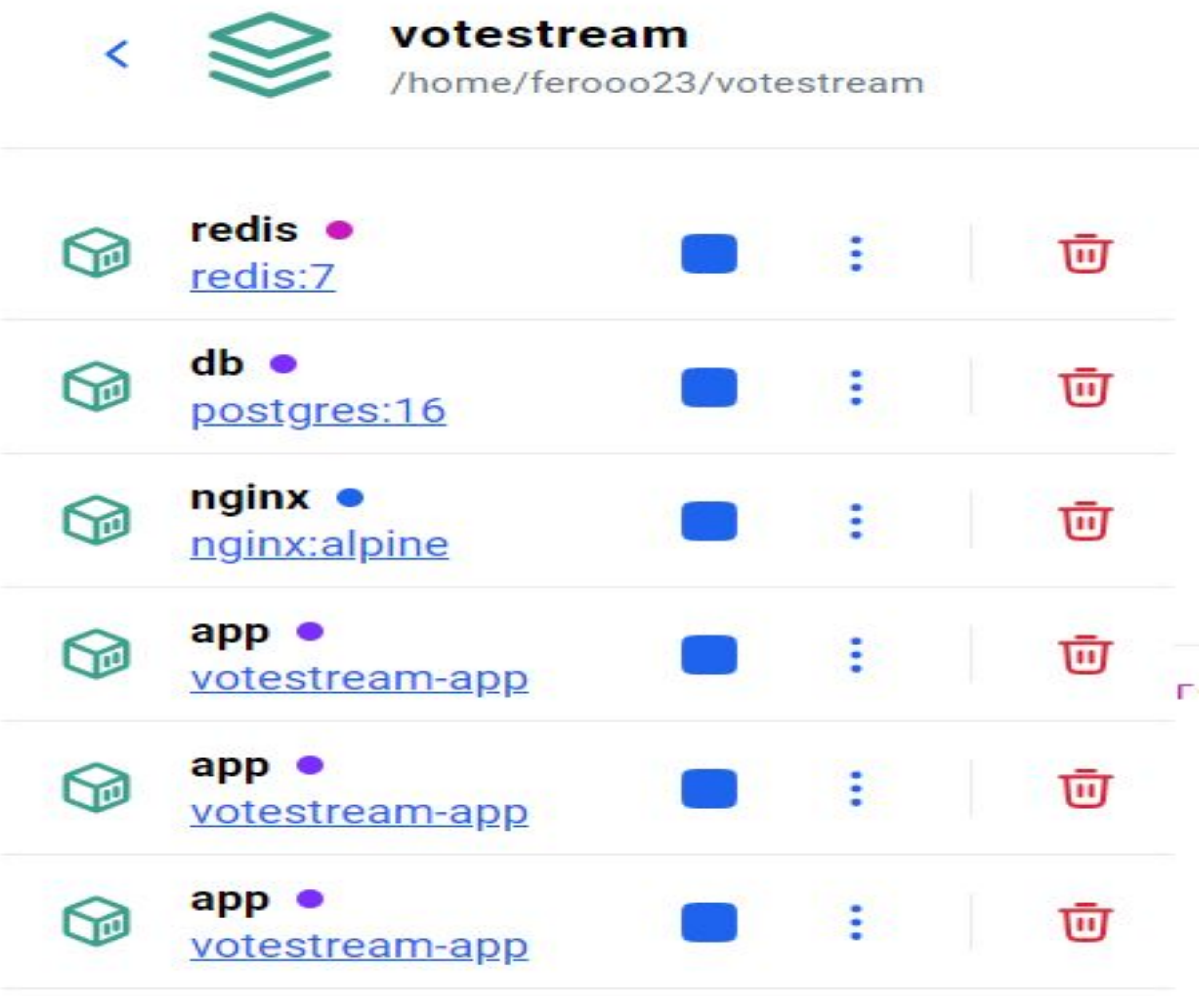
None, the market is saturated 41 votes (25.0%)

A decentralized micro-video platform 37 votes (22.6%)

```
{
  "The resurgence of niche forums": 35,
  "A generative AI app": 51,
  "None, the market is saturated": 41,
  "A decentralized micro-video platform": 37
}
```

- When a user selects an option, the vote is saved to PostgreSQL.
- Results are instantly updated for all active users via Redis Pub/Sub and WebSockets.
- This showcases real-time updates and concurrent connection handling.

Scaling the System – Load Balancing



votestream
/home/ferooo23/votestream

redis
redis:7

db
postgres:16

nginx
nginx:alpine

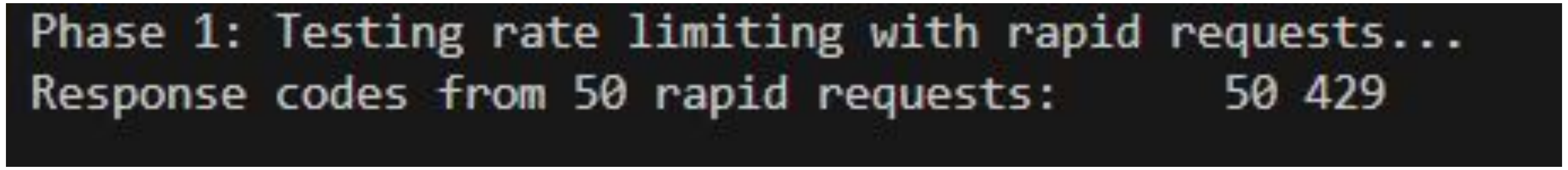
app
votestream-app

app
votestream-app

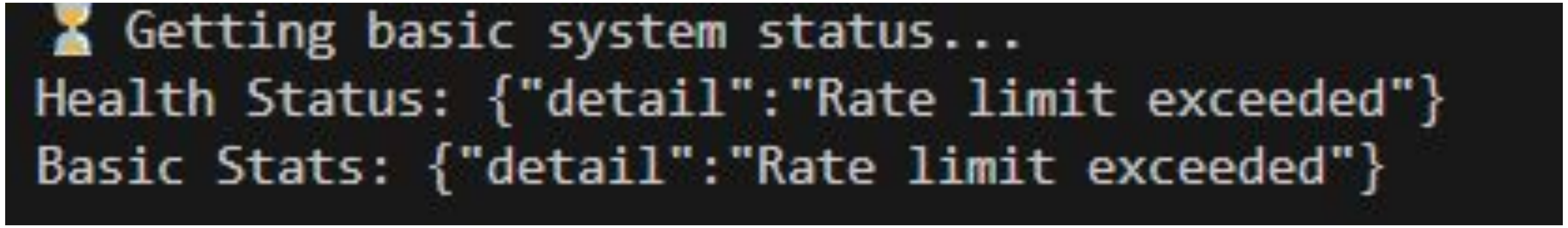
app
votestream-app

- Our stateless architecture allows multiple backend instances to run in parallel.
- Nginx acts as a load balancer and distributes incoming traffic evenly.
- Here we demonstrate multiple containers or Gunicorn workers handling requests concurrently.

Scaling the System – Load Balancing



```
Phase 1: Testing rate limiting with rapid requests...
Response codes from 50 rapid requests:      50 429
```



```
⏳ Getting basic system status...
Health Status: {"detail":"Rate limit exceeded"}
Basic Stats: {"detail":"Rate limit exceeded"}
```

- A custom rate limiter restricts the number of requests per client to prevent abuse.
- A circuit breaker pattern protects Redis and the database from cascading failures under stress.
- This ensures system resilience and graceful degradation during high load.

Conclusion

- VoteStream successfully meets all project requirements, demonstrating a robust, scalable, and resilient architecture.

- We have implemented state management, vertical/horizontal scaling, overload mitigation, caching, and asynchronous patterns.

- The project is fully containerized and easy to deploy.

🔗 GitHub Repository: https://github.com/ferooo23/votestream.git