**Project Proposal: Simulated Social Media Platform (SSMP)**

**📌 Overview**

**SSMP** is a large-scale, modular, backend-focused simulation of a time-accelerated digital world where autonomous agents (simulated users) interact with a dynamic social media platform. The system is built using a microservice architecture, allows deep observability, and is designed to test the evolution and stress behavior of complex distributed systems.

**🔥 Project Goals:**

* Build a **modular**, **scalable**, and **testable** backend system simulating a real-world social network.
* Introduce **autonomous behavior-driven users** that generate realistic load and usage patterns.
* Control and monitor the system using **accelerated virtual time** and **zone-based latency simulation**.
* Allow for **easy expansion** with new services (ads, messaging, search, moderation) without rewriting the core system.

**🏗️ System Architecture**

**🧩 Microservice Monorepo Structure**

graphql

CopyEdit

social-sim/

├── services/

│ ├── user-service/ # user accounts, profiles, simulated zones

│ ├── post-service/ # post creation, retrieval, engagement tracking

│ ├── comment-service/ # comment posting and management

│ ├── feed-service/ # feed generation and trending post logic

│ ├── sim-user-engine/ # sim agents w/ behavioral models and request logic

│ ├── zone-service/ # latency simulation and zone control

│ ├── admin-dashboard/ # metrics export, admin API, mod tools

│ └── gateway/ # centralized API router or reverse proxy

├── proto/ # shared API definitions (OpenAPI/gRPC)

├── shared/ # shared code: auth, logging, schemas

├── docker/ # container config, compose files

├── k8s/ # helm charts, manifests, service discovery

├── scripts/ # CLI tools, seeders, behavior tuners

└── README.md

**🔀 Communication Style**

| **Channel** | **Use Case** |
| --- | --- |
| REST | SimUser ➝ Gateway ➝ Backend services |
| gRPC | Inter-service communication (optional) |
| Redis | Lightweight cache and Pub/Sub |
| Kafka | Optional for future async events (e.g., post created ➝ update feed) |

**🧱 Core Services Breakdown**

**🧍‍♂️ user-service**

* Simulated user registration
* User behavior profile assignment (lurker, poster, troll, influencer)
* Zone assignment (affects latency)
* Token issuance for simulated auth

**Stack**: FastAPI, PostgreSQL, Alembic, optional gRPC client

**📝 post-service**

* Post creation + metadata tracking (likes, comments, author)
* Post retrieval
* Caching popular posts

**Stack**: FastAPI, SQLAlchemy, PostgreSQL, Redis

**💬 comment-service**

* Add/view comments
* Flag for moderation
* Future integration with NLP for toxicity detection

**📰 feed-service**

* Aggregates posts for a user feed (global or friend-based)
* Trending detection based on reactions/engagement
* Cache feeds for performance

**🌍 zone-service**

* Zones (locations) that determine simulated network conditions
* Simulates latency per zone (e.g. city vs rural)

**Example Zones**:

| **Zone** | **Latency** |
| --- | --- |
| A (City) | 20ms |
| B (Suburb) | 100ms |
| C (Rural) | 400ms |
| D (SlowNet) | 1000ms |

**👾 sim-user-engine**

* Runs a fleet of autonomous agents (SimUsers)
* Each has:
  + Behavior tree (e.g., post ➝ like ➝ scroll)
  + Action timing (based on think-time, randomness)
  + Zone-based latency simulation
* Sends real HTTP requests to the backend
* Logs behavior stats and metrics

**Simulated Behaviors**:

| **Type** | **Actions** |
| --- | --- |
| Lurker | scroll ➝ like ➝ idle |
| Poster | post ➝ like ➝ reply |
| Troll | comment ➝ report ➝ rage-quit |
| Influencer | post ➝ respond ➝ reply ➝ react |

**🧑‍💻 admin-dashboard**

* Exposes observability and admin control via API
* **Metrics** (Prometheus-exported):
  + active\_users
  + requests\_per\_second
  + average\_post\_latency\_by\_zone
  + new\_users\_per\_hour
  + engagement\_per\_user
  + moderation\_events\_triggered
* Optional: Use Grafana to visualize

**🔀 gateway**

* Routes external traffic to services
* Applies auth checks
* Injects headers like user ID or simulated delay

**🧪 Simulation Engine (SimUser System)**

**🧠 Components:**

* Time-control system:
  + Virtual clock w/ adjustable speed (1x, 10x, pause)
* Behavior engine:
  + FSM or behavior trees per user
* Request engine:
  + Latency injection based on zone
  + Async HTTP client
* User onboarding:
  + Adds new SimUsers gradually or on trigger

**🔎 Observability**

**🛠 Stack:**

* Prometheus for metrics
* Grafana for visualization
* Loki (optional) for logs
* /metrics endpoint on every service

**📊 Key Metrics:**

| **Name** | **Description** |
| --- | --- |
| requests\_total | Count of HTTP calls made by users |
| latency\_by\_zone | Histogram of latency per user zone |
| posts\_per\_tick | Rate of post creation over time |
| user\_growth\_rate | New users per virtual hour |
| comment\_toxicity\_score | Placeholder metric for moderation |
| feed\_generation\_time | Feed service performance |

**🧱 Data Schema Overview (PostgreSQL)**

**Users**

sql

CopyEdit

Users (

id UUID PRIMARY KEY,

name TEXT,

behavior\_type TEXT,

zone\_id UUID,

created\_at TIMESTAMP

)

**Posts**

sql

CopyEdit

Posts (

id UUID PRIMARY KEY,

user\_id UUID,

content TEXT,

created\_at TIMESTAMP,

like\_count INT,

comment\_count INT

)

**Comments**

sql

CopyEdit

Comments (

id UUID PRIMARY KEY,

post\_id UUID,

user\_id UUID,

content TEXT,

created\_at TIMESTAMP

)

**Likes**

sql

CopyEdit

Likes (

id UUID PRIMARY KEY,

post\_id UUID,

user\_id UUID,

created\_at TIMESTAMP

)

**Zones**

sql

CopyEdit

Zones (

id UUID PRIMARY KEY,

name TEXT,

latency\_sim INT

)

**🧱 Infrastructure**

**🧰 Local Dev:**

* docker-compose for:
  + All services
  + Redis
  + PostgreSQL
  + Prometheus
  + Grafana
* Shared .env files
* Simple startup via Makefile or run.sh

**🚀 Future Expansion Ideas**

| **Feature** | **Description** |
| --- | --- |
| 🔒 Auth Service | Real JWT-based auth and OAuth |
| 🗨️ Messaging | DMs between users |
| 🔍 Search | Meilisearch or Elastic for posts/users |
| 📣 Ad System | Simulated targeting, engagement |
| 👮 Moderation | NLP-based comment moderation |
| 🌐 Web Simulator | Fake websites for users to browse |
| 📱 Mobile Agents | Mobile vs desktop behavior simulation |
| 🎥 Media Upload | Simulated video/image posting |
| 🧪 AB Testing Framework | Internal testing platform for features |

**🧑‍💻 How It Helps You in Interviews**

* You’ve designed **a distributed, event-driven social simulation**.
* You’ve built:
  + Microservices
  + Scalable data schemas
  + Prometheus-based observability
  + Simulated load using behavioral agents
* You can show your code, show the Grafana dashboard, and talk systems like a staff engineer.

**✅ MVP Milestone Plan**

**🧱 Week 1–2**

* Setup repo, Docker Compose, FastAPI boilerplates
* Build user-service, post-service, comment-service

**🧠 Week 3**

* Build sim-user-engine with behavior loop
* Add zone-service and inject request latency

**📊 Week 4**

* Add Prometheus metrics to all services
* Create admin-dashboard and Grafana dashboard

**🔄 Week 5+**

* Add feed-service, moderation, scaling tools, chaos simulation