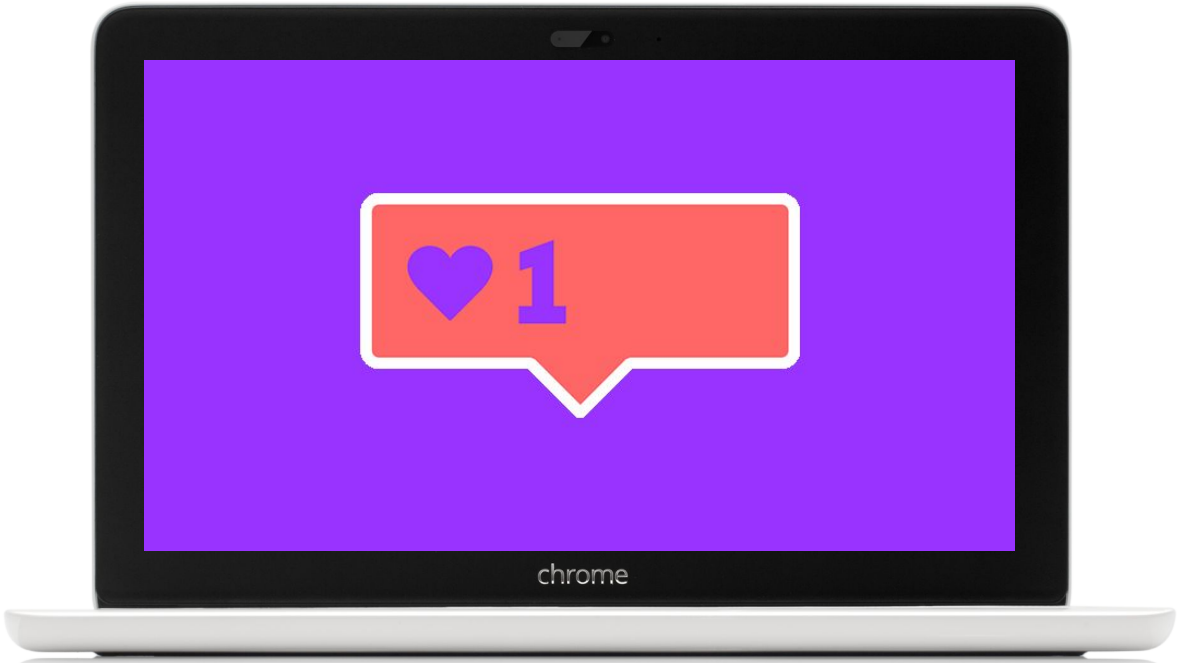
Fanlinc

Best Team

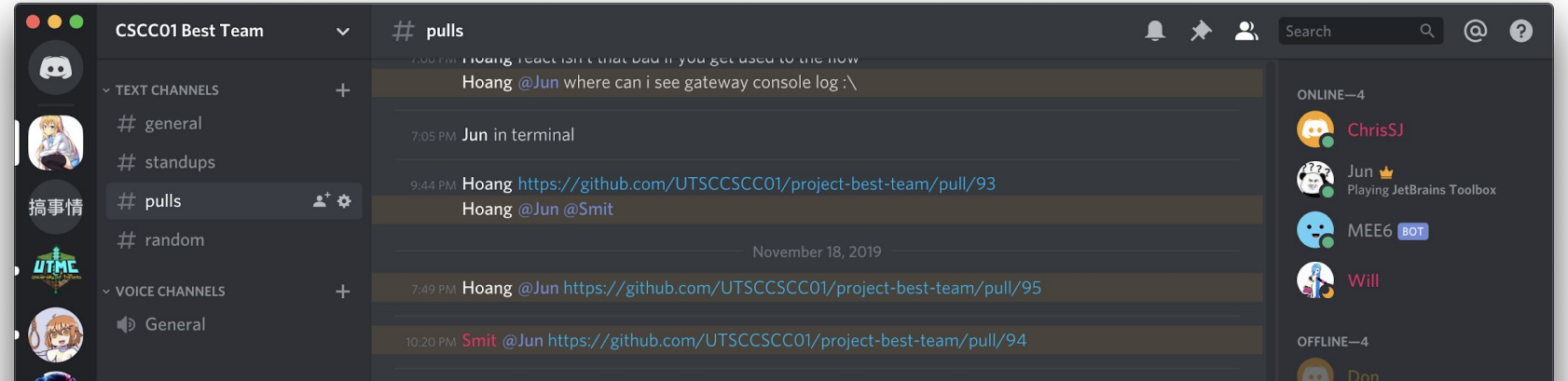
Jun Zheng
Seongjin “Chris” Hong
Donnie Siu
Smit Patel
Minh Hoang Nguyen
William Song



Process

🎤 Communication

- **Discord**
- **Physical meetings** Monday at noon
- Daily **stand-ups** even during weekends
  - Lenient, can say did nothing, just provide an explanation
  - *Almost no blockers because microservice architecture!*



👀 **Pulls** and Code Review

- **No push to master** allowed, must issue pull requests.
- All pull requests must have at least **one assigned reviewer**.
  - If possible, code owner should be assigned as the reviewer.
  - otherwise choose anyone who is available and can review at that time.
- At **least one approval** is required before merging the PR.

👀 Pulls and **Code Review**

- Reviewer must ensure they can **run the code** on their computer.
- **Criteria of Rejection (CoR)**
  - Provable bug (must come up with way to reproduce)
  - Architectural level issues, for example, complex business logic in controller
- **approve the PR** otherwise
- *It takes time to review code, and a PR rejection can result in PR delayed for hours if not days.*

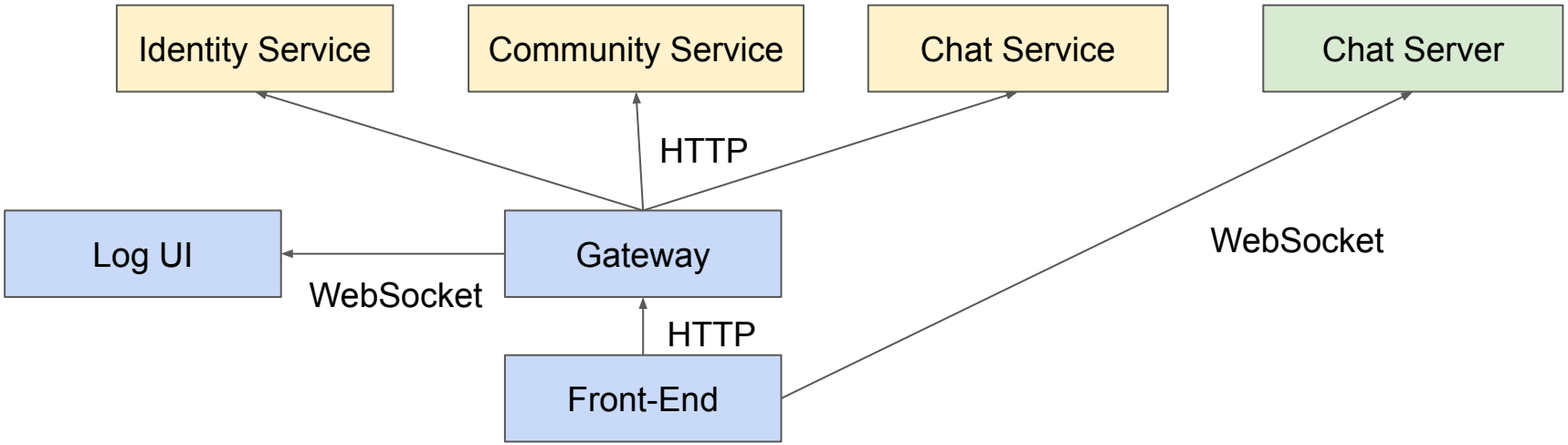
⌨️ Coding Standard

- **Java** - Google Style
- **JavaScript / TypeScript** - Prettier.io
- *We simply follow these standards, no special style requirements.*

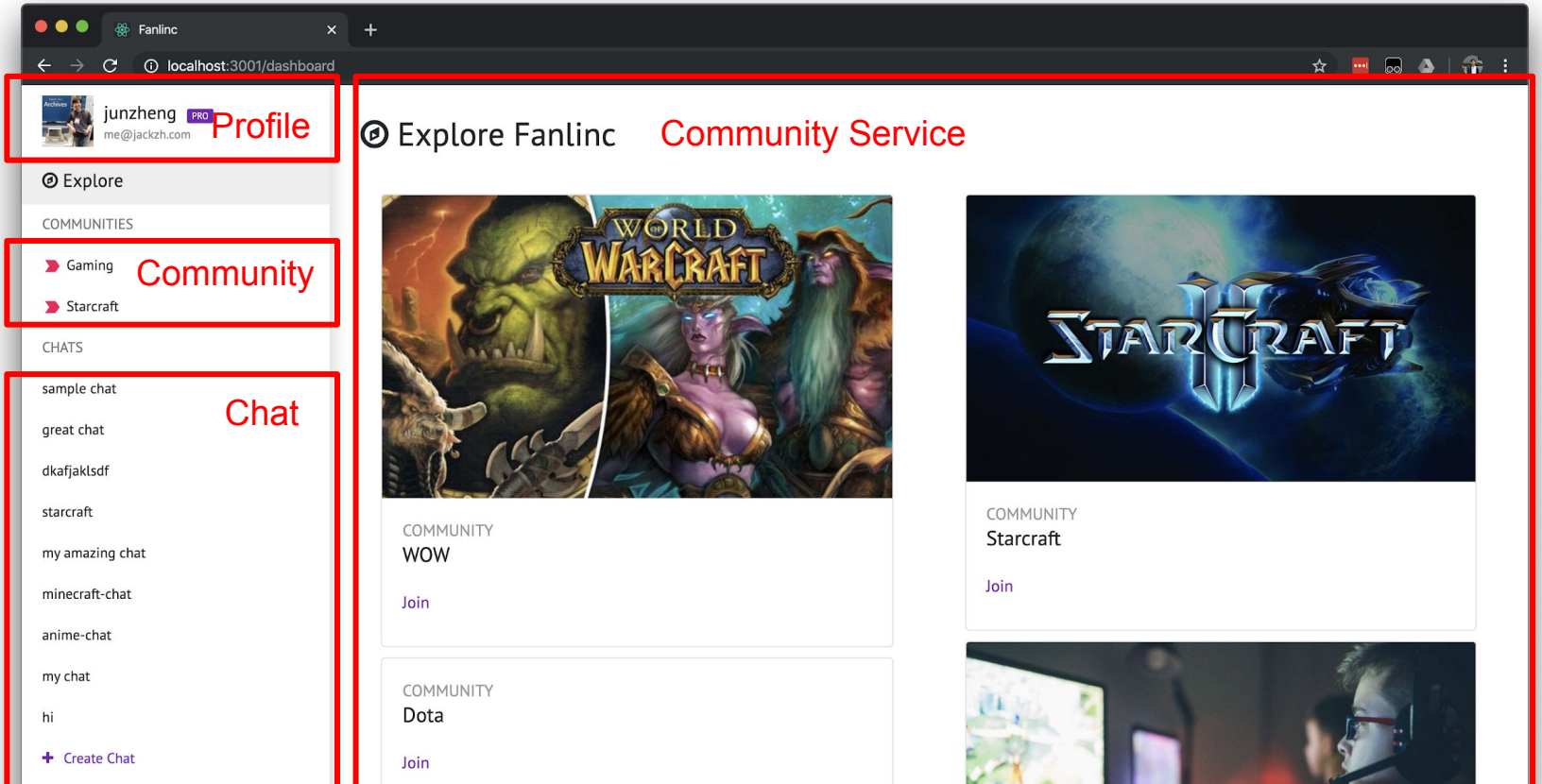
Architecture

Microservices

- **Aggregator** microservice pattern.
  - Parallel development, allows us to use/learn many different technologies.
- Gateway exposes **REST interface** that can be used by front-end.

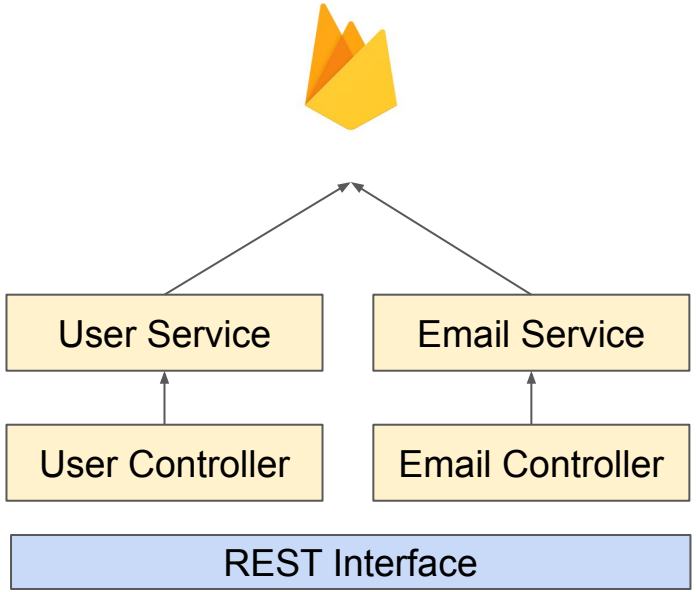


Microservices



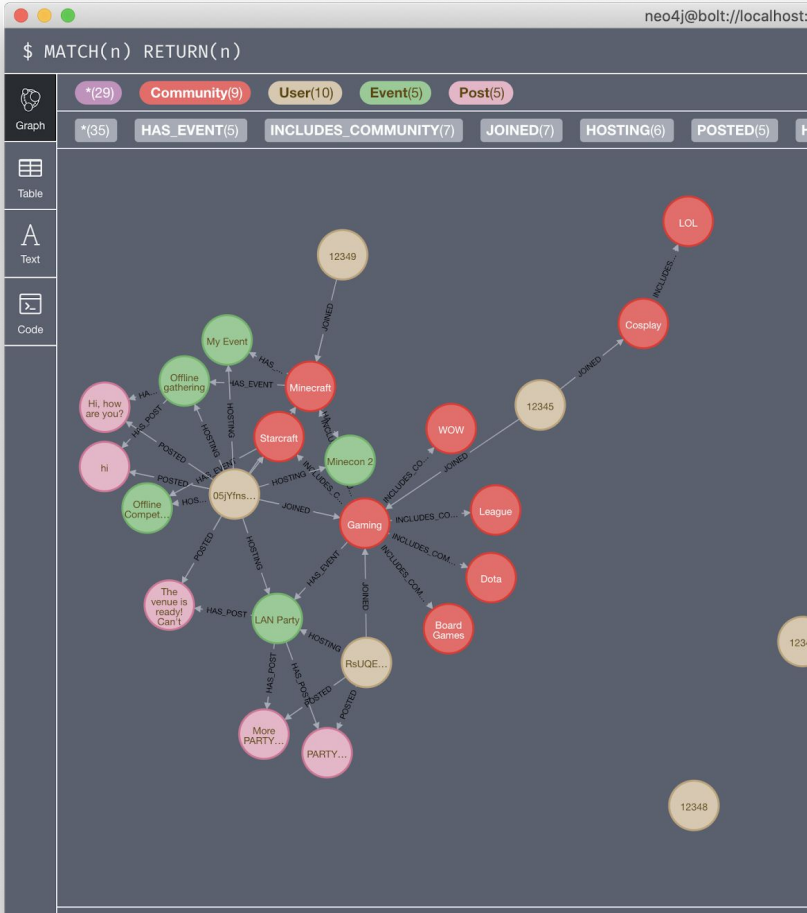
Identity Service

- Responsible for
  - Login (JWT issuance)
  - Signup
  - Authorization (JWT verification)
  - Profile information storage
- Technologies used

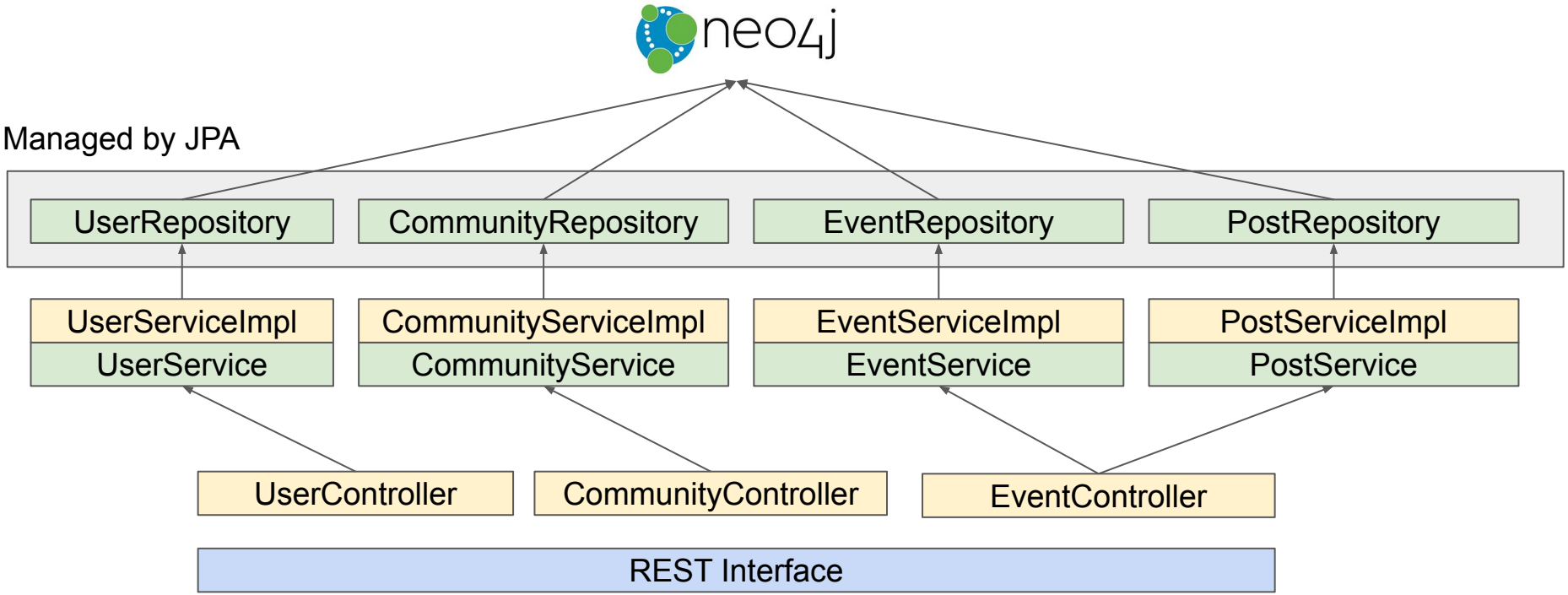


Community Service

- Responsible for
  - Joining communities
  - CRUD on community graph
  - CRUD on events and posts
  - Linking chats with communities
- Technologies used



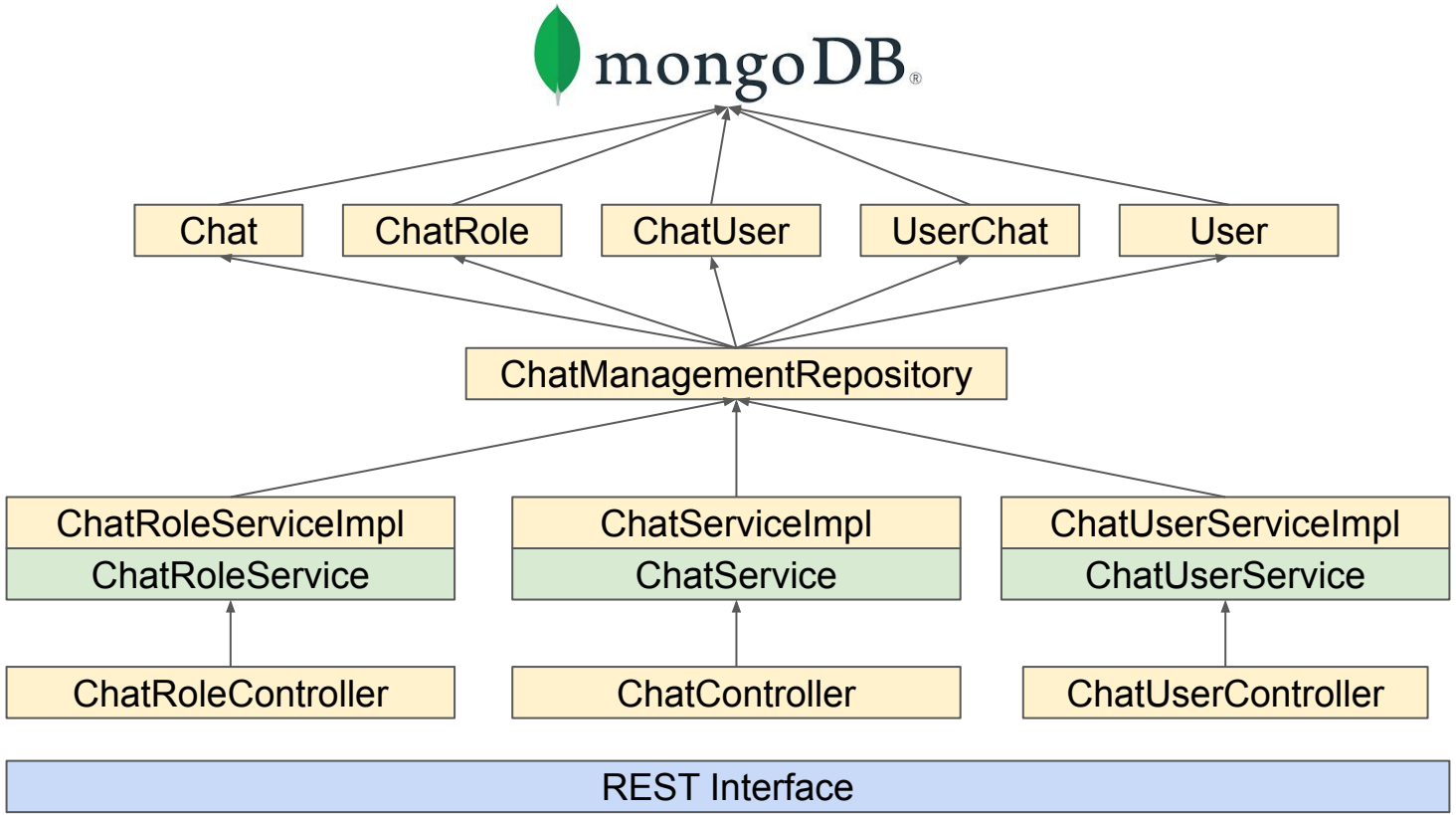
Community Service



Chat Service

- Responsible for
  - Create chats
  - Make chat private/public
  - Joining chats
- Technologies used

Chat Service



Chat Server

- The actual server that hosts chats
- Technologies used

Gateway

- Acts as aggregator of all underlying services.
- Provides a single interface for front-end to access.
- Technologies used

⚙ Front-End

- The actual application you see!
- Technologies used

- Custom state management, similar to Redux, but simpler.

🕐 Recap

- Microservice - Aggregator
- Languages
  - Python, Java, TypeScript
- Databases
  - Firebase, Neo4j, MongoDB
- Frameworks
  - Spring, Flask, Express, React
- MVC within services

💧 Challenges

- Initial integration with front-end
  - Initially we didn’t have gateway
  - Need a way to adapt to future changes once gateway is implemented
- Different team members knows different languages
  - Microservice architecture!
- Hard to setup
  - Solved by building our own process management service

📺 Demo!

User Profile

Communities & Chats

Events & Posts

❓ Questions?

Users & Profiles

feature 1

- login
  - identity
  - security
  - email verification
- profile
  - profile picture
  - display name
  - PRO designation
  - chat roles
  - private information

Communities

feature 2

- hierarchy of communities
- join/leave with prophecies
- dynamically scales
- roles

Chats

- real time messaging
- user-created private channels independent of communities

Events & Posts

feature 3

Fanlinc

as presented by the
Best Team
Jun Zheng
Seongjin “Chris” Hong
Donnie Siu
Smit Patel
Minh Hoang Nguyen
& William Song