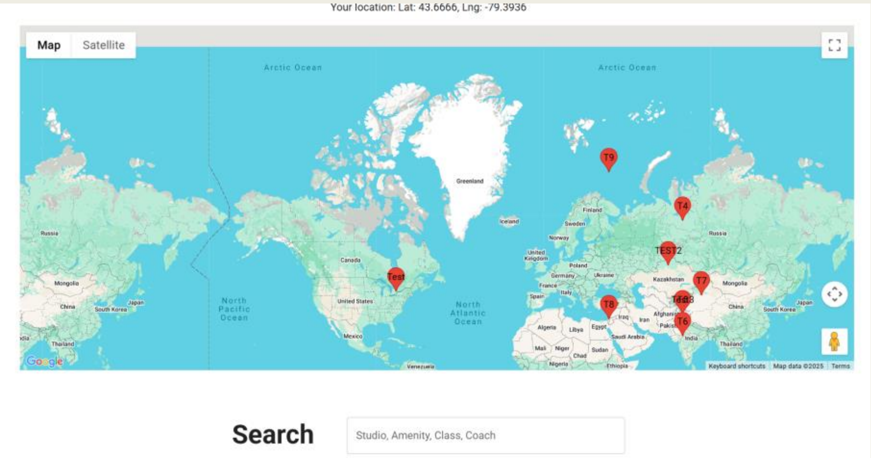
A SELF-ADAPTIVE FITNESS CLUB APPLICATION

Team1: Caules Ge, Ethan Li

What is it?

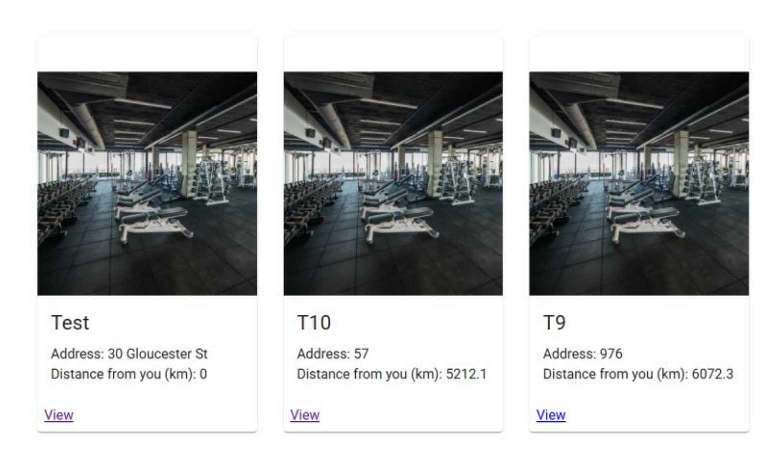
- ❑ A fictional web system for a chain of fitness clubs.
- ❑ It supports two user roles: administrators and members.
- ❑ Administrators can perform CRUD operations for studios and their affiliated amenities.
- ❑ Members can sign up, log in, view available studios, and schedule courses.

Features include an interactive map for studios, and the list is sorted by distance...



Your location: Lat: 43.6666, Lng: -79.3936

Search Studio, Amenity, Class, Coach



Test
Address: 30 Gloucester St
Distance from you (km): 0
View

T10
Address: 57
Distance from you (km): 5212.1
View

T9
Address: 976
Distance from you (km): 6072.3
View

Problem

The Canada Fitness Club system experiences significant traffic surges during promotions.

High traffic causes **slow responses** and **increased errors**, while low traffic results in **wasted resources** due to extra replicas.

Without self-adaptation, the system cannot remain efficient or provide a stable user experience.

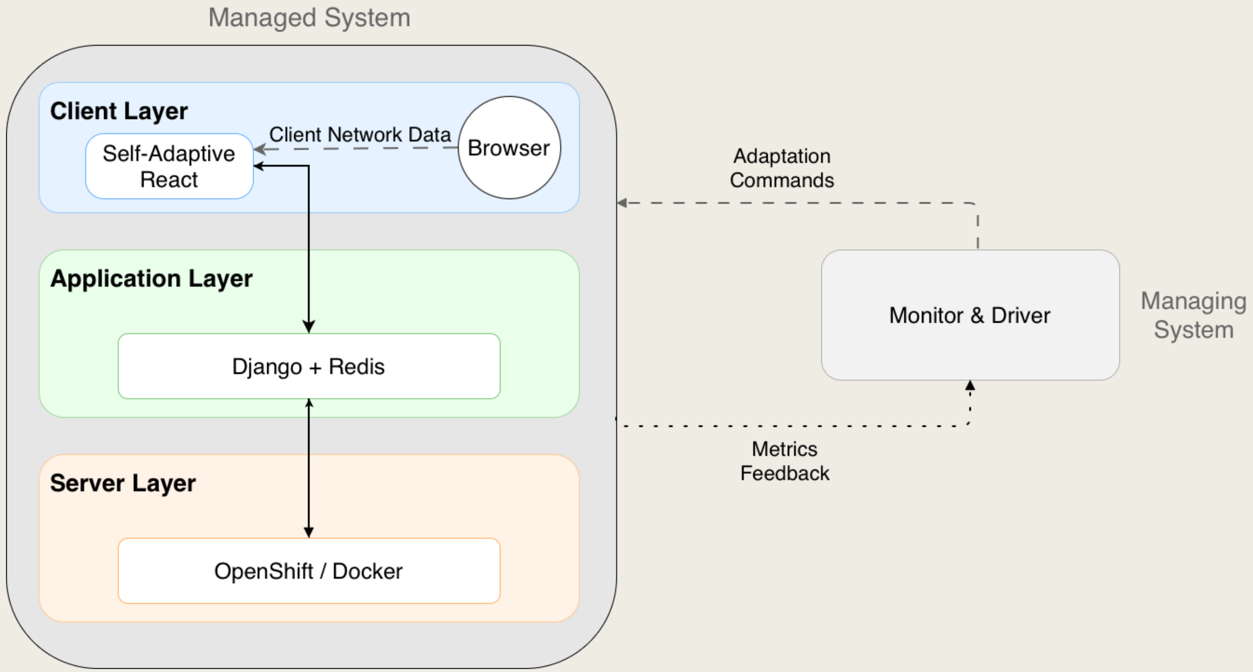
Object

3 Layers of Adaptation

- **Client**: simplifies the UI under poor network conditions
- **Server**: scales horizontally based on workload
- **Application**: switches between standard and degraded modes when the workload is extremely high

Goals

Improve response time, reduce errors, and avoid resource waste during low-traffic periods



Tools

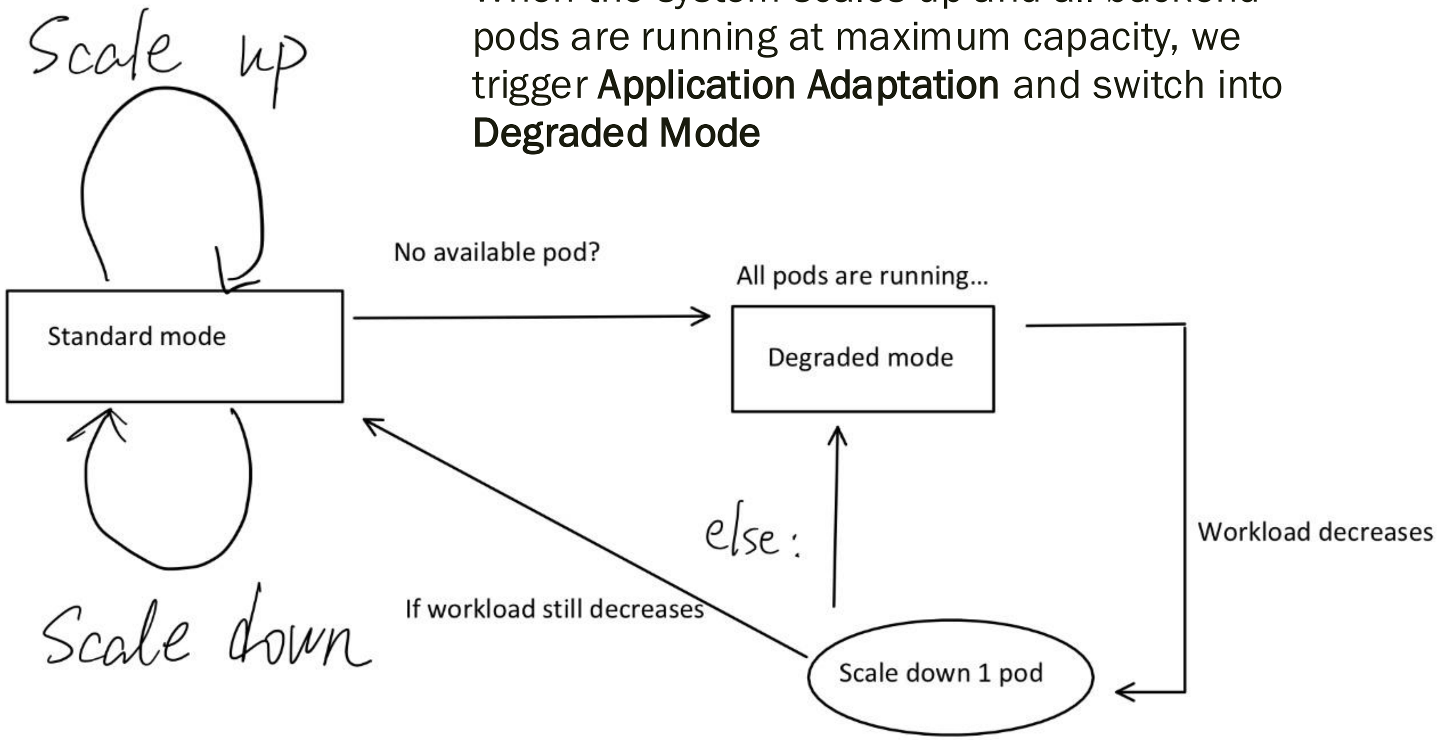
- Sysdig: collects CPU, memory, error rate, and p95 latency metrics.
- Redis: stores degraded-mode flags and cached responses of each endpoint.
- React: monitors the average FPS, network quality, and kbps download speed of the client. If client

Client-level adaptation:

When the average FPS of the last 5 seconds drops or the network is weak (eg, 3G), the UI switches to a lightweight mode with reduced pagination size, hides the map, and uses lower resolution images

System and Application Level

- When the system scales up and all backend pods are running at maximum capacity, we trigger **Application Adaptation** and switch into **Degraded Mode**



- **Reason**: avoiding returning to standard mode while all pods are still overloaded, which would harm performance

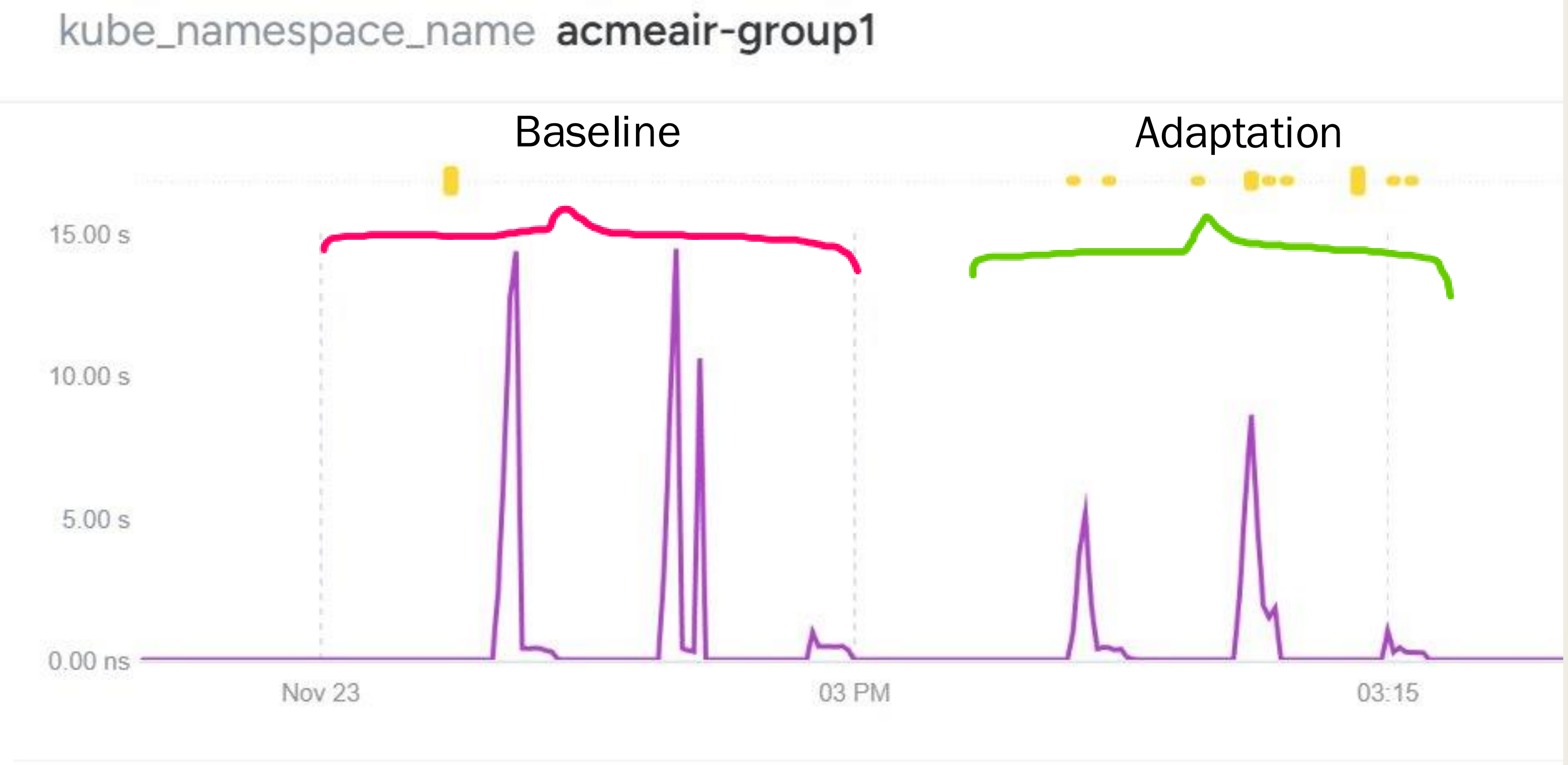
Degraded mode:

Disabling expensive user-specific features, such as distance-based sorting, allowing Redis caching for pages that normally cannot be cached

For other pages that have been cached, it increases the cache TTL.

Reducing the pagination size from 9 to 6 (fewer DB queries per page)

Result: System and Application Level



Average response time under three workload levels, from high to low