# Student Move-in-Mate

## A roommate finder website

**CSCI 5253**

Dec 14, 2022

**Team members:**

Meghana Vasanth Shettigar
Ranajit Roy
Shreya Maitra

## 1. Overview:

Moving to a new location is difficult for everyone, but it can be particularly challenging for a student who has just been accepted to a university. Financial constraints force almost every student to share apartments and rooms and it is important to choose a suitable roommate. Our project is focused on creating a website that helps students in finding suitable roommates to live with, temporarily or permanently and has an exciting stay in the new city by collecting various student's major preferences.

## 2. Project Goal:

In this project we have developed a website with the goal to help students find a perfect match for roommates. This will assist mostly newly admitted students during their move-in process. We created a system to match users using preferences like location, food, pet allowance, smoke preferences etc. Users can search manually for roommates using various relevant filters. There is a communicating feature which users can use to communicate without sharing personal details.

Our main goals are here as follows -

- Help the newly admitted students easily find their perfect match as roommates.
- Give a platform to look at possible options for roommates and compare them.
- Provide them a place to discuss and communicate with fellow students.

## 3. Components Used (Hardware/Software):

### Google Protocol Buffers:

A language-neutral, platform-neutral, extensible methodology for serializing structured data in a way that is both forward- and backward-compatible is provided by protocol buffers. Similar to JSON, but smaller, quicker, and with native language bindings. For instances where data is transferred between numerous microservices in a platform-neutral manner, protobuf is more suited.

*Advantages:*

- Optimization: The payloads of messages sent over the network utilizing Protocol Buffers are serialized in binary. As a result, they are substantially smaller than XML or JSON. With so many network calls in a microservice design, you will save bandwidth and enhance network performance.
- Efficient parsing: Data is represented in a binary format with Protocol Buffers, reducing the amount of encoded messages and making parsing less CPU-intensive. Thus, even in IoT or mobile devices with inferior CPUs, message exchange occurs more quickly.

*Disadvantages:*

- A gRPC service cannot be directly called from a web browser because of how intensively HTTP/2 is used by gRPC. The control over web requests required to operate a gRPC client is not offered by any modern browser. Because of this, conversions between HTTP/1.1 and HTTP/2 need the use of a proxy layer and gRPC-web.

Considering all of these we chose to use this in our logging system with gRPC. All of the microservices are constantly giving out logs which if not well packaged or compressed can create a significant overhead on the VPC bandwidth. That's we chose this for logging system.

## REST and gRPC:

**REST**: Representational State Transfer, or REST for short, is an architectural design pattern for creating online services that communicate via the HTTP protocol. It became well-liked as a scalable and adaptable replacement for more conventional machine-to-machine communication techniques.

*Advantages:*
One of REST APIs' main benefits is the amount of freedom they offer. REST can support a variety of call types, return diverse data formats, and even alter architecturally with the proper implementation of hypermedia because data is not linked to resources or functions. They are therefore well-liked as a result of their ease of use, scalability, speed, and capacity for all data kinds.

*Disadvantages:*
Since there is no established contract between the service and the client, communication must take place in other ways, including through written materials or emails. There can't be any asynchronous calls because it relies on HTTP.

**gRPC:** gRPC is a cutting-edge, open source, highly effective Remote Procedure Call (RPC) framework that may function in any setting. With pluggable support for authentication, load balancing, tracing, and health checking, it can effectively connect services within and between data centers.

*Advantages:*

gRPC is designed for low latency and high throughput communication. gRPC is great for lightweight microservices where efficiency is critical. Point-to-point real-time communication: gRPC has excellent support for bi-directional streaming. gRPC services can push messages in real-time without polling.

*Disadvantages:*

It is not possible to directly call a gRPC service from a web browser because gRPC mainly relies on HTTP/2. No current browser offers the web request control necessary to support a gRPC client. Therefore, conversions between HTTP/1.1 and HTTP/2 must be handled by a proxy layer and gRPC-web.

***REST vs gRPC*:** When receiving data, gRPC outperforms REST by a factor of around seven, and when sending data of any payload, it outperforms REST by a factor of ten. This is mostly caused by gRPC's use of HTTP/2 and the tightly packed Protocol Buffers.

Implementation with REST API is much faster than gRPC. But the advantages with gRPC comes into life when it is being used more. That's we used REST to create all the microservices and used gRPC to ensure less overhead during logging.

## Virtual Machine and Kubernetes:

A virtual machine is a computer file that functions like a real computer. These files are generally referred to as images. It can act as the user's complete computer experience, as is typical on many people's work PCs, or it can run in a window as a separate computing environment, frequently to run a different operating system.

*Advantages and Disadvantages of Virtual Machines:*

- Users can run several OS instances on a single piece of hardware because the software is distinct from the actual host machine, saving a business time, money, and physical space.
- Due to their indirect hardware access, virtual machines are less effective than physical ones. When virtual machine software is running on top of the host operating system, the physical device must be asked for access to its storage and memory.

Kubernetes streamlines application management by automating operational activities associated with container management and providing built-in commands for application deployment, rollout of updates, scaling up and down to accommodate changing requirements, monitoring, and more.

*Advantages and Disadvantages of Kubernetes:*

- Users can run several OS instances on a single piece of hardware because the software is distinct
- Although Kubernetes offer services like run, build, and scaling the containerized application faster, it's not a PaaS-specific system.

Even if Kubernetes would have been the better choice to host the web services, we chose VM because in GCP environment deploying and working with load balancer becomes much easier. So it was our first choice for web service and for the other micro services we user Kubernetes.

## **Google Cloud Storage**:

Data can be stored and accessed on the Google Cloud Platform infrastructure using Google Cloud Storage, a RESTful online file storage web service. The service combines cutting-edge security and sharing features with the performance and scalability of Google's cloud. It is a cloud-based infrastructure service (IaaS).

*Advantages:*
By using cloud storage, you can make data accessible whenever you need it, from anywhere. As long as they have an internet connection, anyone can access data from anywhere in the globe using any device, rather than being restricted to a particular place or device.

*Disadvantages:*
The cloud providers may occasionally experience technical issues that might occur for a variety of reasons, including power outages, poor Internet access, the need for maintenance at data centers, etc. Data

security and privacy issues resulting from breaches or leaks, outages, and a loss of control over your data are a few hazards associated with cloud storage. The sensitivity of the data kept in the cloud and the security precautions implemented by cloud service providers affect a number of hazards.

We used this object store for images that users uploaded for profile pictures. Also we used google cloud storage to periodically store new log files.

## ReactJS and ExpressJS:

**ReactJS:** An open-source JavaScript framework and library is called React.js. React. js is a compact JavaScript framework that may be used to design user interfaces and to make reusable User Interface components.

*Advantages:*
React. js is a declarative, effective, and versatile JavaScript library that aids in creating quick and easy-to-use online apps.

*Disadvantages:*
Only the UI Layers of the application are covered by ReactJS. To obtain a full toolkit for project development, you must still select a few other technologies.

**ExpressJS**: Express is a node js framework for online and mobile applications that offers a wide range of functionality. It is used to create web applications that are single page, multipage, and hybrid. It's a layer that was added on top of Node js to manage servers and routes.

*Advantages:*
The advantage of using Express.JS for backend development is that the application can be scaled easily. With aid of the same language, JavaScript, we are able to write code for both the frontend and the backend.

*Disadvantages:*
Slow Javascript code makes it harder to run pure processing task in the service itself. Lack of threading is another issue.

Despite the challenges, coding both frontend and backend with same programming was a big advantage. It reduces code complexity and development time. That’s why choosing these were a no-brainer.

## MongoDB database:

MongoDB is a non-relational document database that supports JSON-like storage. The MongoDB database includes a flexible data schema that enables you to store unstructured data. It also has full indexing support, replication, and extensive and user-friendly APIs.

*Advantages:*

- Full cloud-based developer data platform.
- Flexible document schemas.
- Widely supported and code-native data access.
- Change-friendly design.

- Simple installation.

*Disadvantages:*
The lack of transaction functionality in MongoDB is one of its drawbacks. Although transactions are becoming less and less common in applications, some still need them in order to update numerous documents or collections. MongoDB shouldn't be used if your team need that feature.

We needed a simple enough database which didn’t need a pre-defined schema. And MongoDB helped cut down development and focus on the main functionalities of the project.

## Load Balancer:

A load balancer serves as the single point of contact for clients.  It splits up incoming application traffic among many targets, like EC2 instances in various Availability Zones. The application is now more readily available as a result.

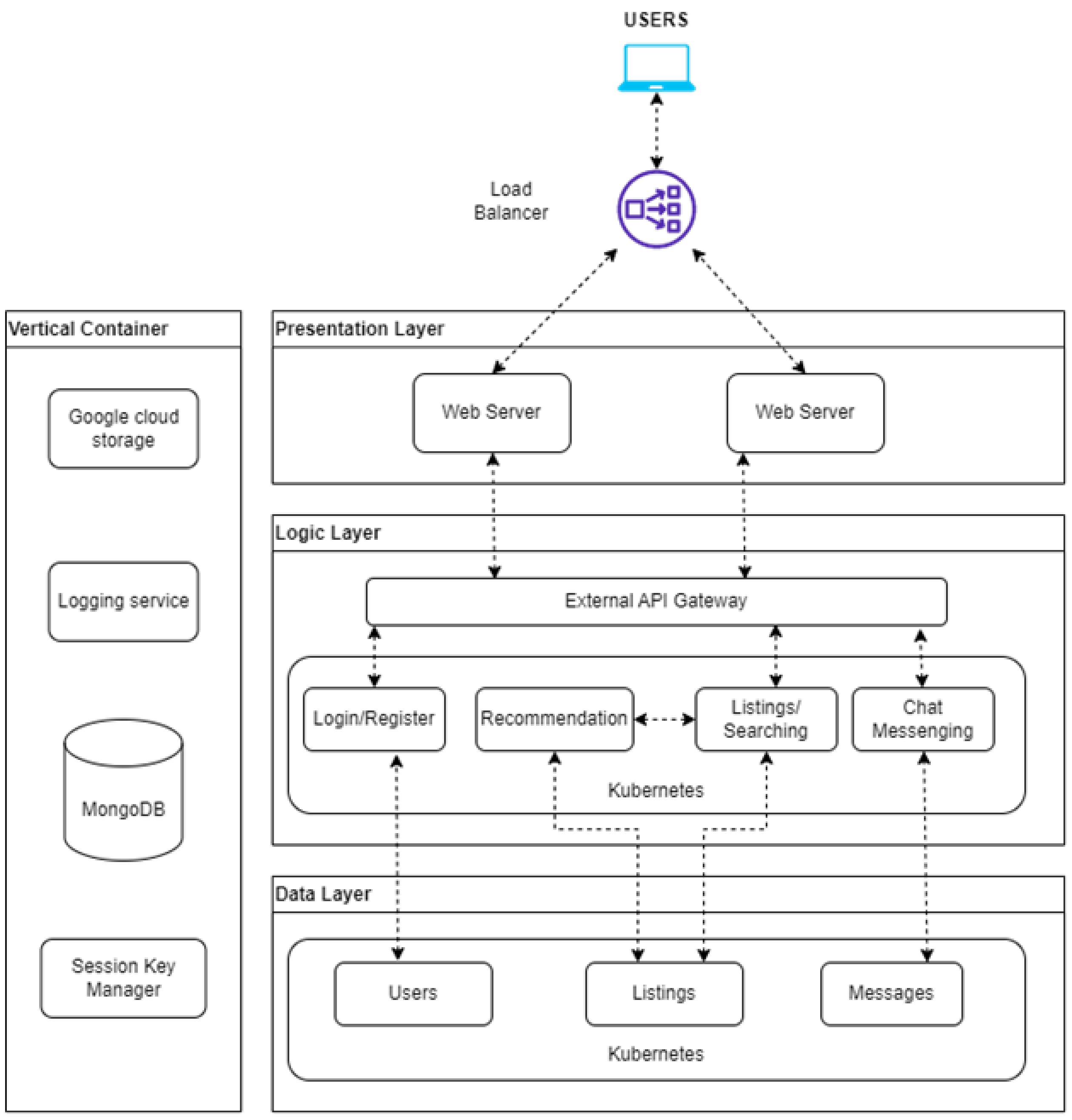
*Advantages:*
Load balancing lets you evenly distribute network traffic to prevent failure caused by overloading a particular resource. This strategy improves the performance and availability of applications, websites, databases, and other computing resources. It also helps process user requests quickly and accurately.

*Disadvantages:*

No native failure detection or fault tolerance and no dynamic load rebalancing. No capability other than round-robin. No way to ensure connection to the same server twice, if required.

This created complexity in our code, but the end result is equally wonderful. The scalability of our project was increased a lot due to this. The ability to upscale and downscale ensures efficient use of resources and costs with still maintaining the responsiveness and availability.

## Interaction with Components:



## Presentation Layer

***Web Server:*** This serves all the requests for the web pages as well as handles all the user interactions with our services. It uses REST API to communicate with clients. Virtual machine instances were used to run these web servers. We have used ReactJS for front-end and ExpressJS for backend services.

### Logic Layer

All the microservices in this layer is managed by Kubernetes.

**Login/Register:** This microservice is being used to login users using their credentials and also register new users. On each login attempt it interacts with Session Key Manager to generate a session key and authenticate users for further API calls in their login session. The credentials are encrypted for more security.

**Listings/Searching:** This a service which is being used to do custom search over the full database. This contains business logics for filtering the user data and their listed preferences. This service will also return a list of users.

**Chat/Messaging:** This service is used for users to communicate between them. This will handle all the requests of sending and receiving of messages. This will provide all chat history.

## Data Layer

These services are provided using one microservice which interacts with databases to provide data as needed by other microservices. This microservice is managed by Kubernetes.

**Users:** This service generally interacts with Login/Register service and Session Key manager. This helps to create new user entry, accessing existing entry from database.

**Listings:** This service generally interacts with  Listings/Searching service. Contains logic pagination and filters.

**Messages:** This service communicates with Chat/Messaging service. It creates and accesses all the messages.

## Shared Components

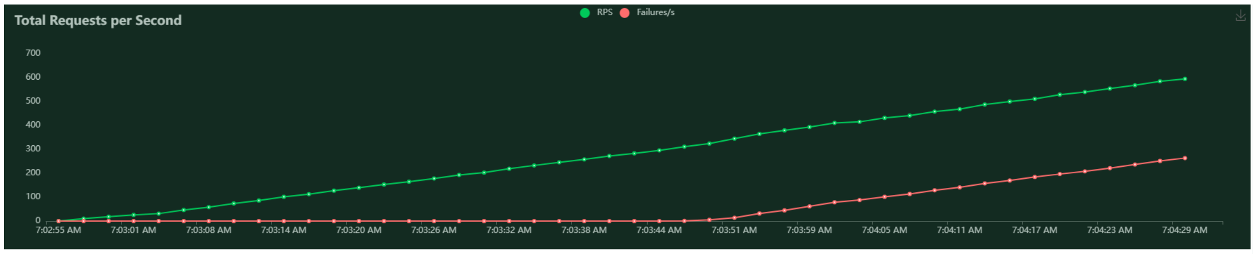
Google Cloud Storage: This is being used to store all the static objects in our project. This has all the images uploaded by the user and also stores the log files. This ensures a single fully managed service that can be maintained and accessed online for accessing our data.
**Logging:** This service is being used to log all the events going on in each of the microservices. The logging messages will have different levels of criticality like debug, info, warning, etc. We are using gRPC to interact with this service.

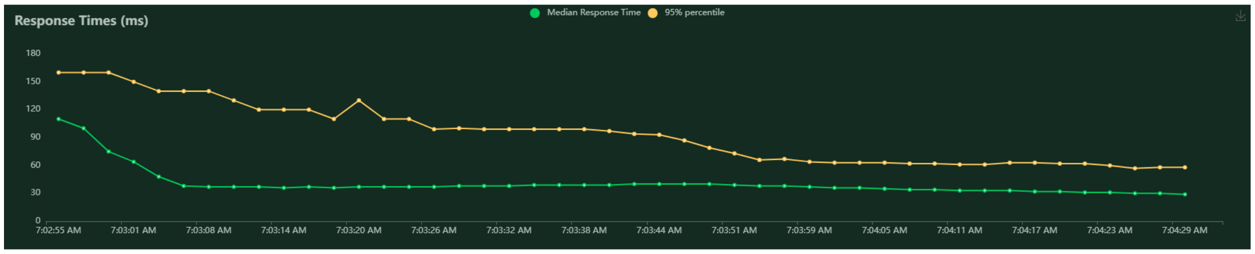
**MongoDB:** We are using MongoDB to store all our dynamic data like user information, listings, etc. It does not impose schemas and stores data in the collections as JSON-based documents.

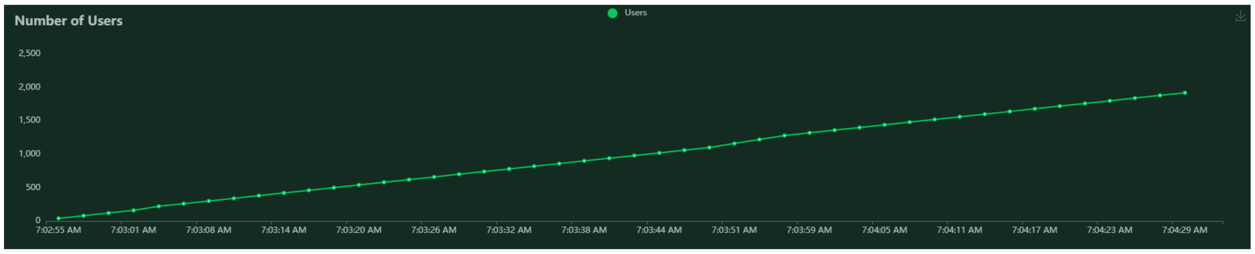
**Session Key Manager:** This is used for creating as well as storing all the active session keys to authenticate each API call after login. All the active sessions have TTL timers so that after a certain period of time the session can expire.

**Load Balancer:** It was used to distribute traffic amongst multiple VMs hosting web services. Load balancers enhance the responsiveness and availability of applications while preventing server overload. Each load balancer is positioned in-between client devices and the backend servers. It receives incoming requests and then distributes them to any server that is accessible and able to process them. It checks for system utilization and starts or stops VMs. It also does Health Checkups on each to see if any of them crashed or not and restarts them as required.

## 4. Capabilities and limits







We used *locust* python module to calculate these metrics posted in the above figure.

Here, we can see that even if we are increasing the number of user and also the requests, the average response times were not changing. The 95- percentile line flattened out at 70ms.

We can also see that the requests were starting to fail after around 1200 simultaneous users. These users are designed to send 1 request every 2 seconds. After this mark the failure rate increased with the increase in number of users.