### **Background and Motivation**

This project aims to develop a specialized online platform for selling customized description services for private gaming servers. The primary product is a service that generates professional, tailored textual descriptions that server owners can use to attract players. This platform will be accessible via a responsive online website and mobile application(in future development, not covered in this project). The motivation for this project stems from the need to provide consumers with a dynamic, dedicated interface to customize their demands and persuade them for better community gaming experience.

### **Overall Objective and Features**

The overall objective is to build a highly available, scalable, and resilient platform that can sustain rapid growth. Key features include a frontend website for marketing; a backend server to record user credentials, account name, membership settings, and friend lists; an integrated digital wallet system to manage customer funds and handle purchases securely; and functionality to allow the buyer of a server description to share links to other friends via an external link, promoting organic user growth.

### **Limitations of Traditional Computing**

Traditional monolithic deployments often use a single server, making the entire service vulnerable during necessary maintenance or hardware failure.

It also **cannot scale rapidly** to handle **high concurrency inflow**. For players hosting servers using their personal server and home network, the latency and hardware bottleneck will destroy gaming experience when traffic exceeds the limits.

With traditional DIY deployment, only the owner of the server or ones with authorization can put the server online. The server will become unavailable if they are occupied or absent. Other players cannot play their save file anytime they want.

### **Benefits Brought by Cloud Computing**

Cloud computing technologies directly address the shortcomings of traditional solutions, making them a perfect fit for this project.

**1.** Managed cloud services provide auto-scaling features, allowing the platform to instantly scale resources up during peak gaming hours and scale down during off-peak times, efficiently handling the **high concurrency inflow**.

**2.** Cloud deployment across multiple zones guarantees that maintenance on one server or data center will not take the entire application down, ensuring continuous service and fault tolerance.

**3.** Utilizing fully managed services (like Cloud SQL and Cloud Load Balancing) minimizes operational overhead and allows the development team to focus entirely on application features rather than infrastructure maintenance.

### **Cloud Technologies Used**

**1. Kubernetes (GKE)**: Provides a managed container orchestration platform for the Microservices Architecture.

**2. Backend Data Storage (NoSQL):** A fully managed, highly scalable NoSQL database used for flexible, high-read/write data where immediate consistency is less critical than speed and scale.

**3. Relational Data Storage (SQL):** Used for strictly relational and critical data requiring ACID compliance.

**4. Load Balancing (HTTP(S))**: Ensures high availability and distributes incoming traffic effectively across the GKE cluster.

### Cost estimate

Since it is a small pilot project, I choose 4 CPU, 8GB memory and 25GB of persistent disk as the spec of our VM. The monthly cost comes to $89.05, which is very affordable.( See appendix 1)

### Architecture Design

1. **Client Access:** The user (web or mobile) initiates a request via the application domain.
2. **Load Balancing:** **Cloud Load Balancing** directs the request to the highly available GKE cluster.
3. **API Gateway:** An internal service within GKE acts as an API gateway, handling authentication and routing the request to the correct microservice.
4. **Microservices Execution:**
   * **User Service:** Interacts with **Cloud Firestore** to handle account login, friend lists, and membership status.
   * **Wallet Service:** Interacts with both **Cloud Firestore** (current balances) and **Cloud SQL** (transaction history) for secure payment processing.
   * **Description Service:** Executes the core logic, fetching templates from **Cloud SQL** and returning the customized description.
5. **Data Persistence:** Data is stored in the appropriate managed service: **Cloud Firestore** for highly concurrent, flexible user/wallet data, and **Cloud SQL** for strict relational and financial records (transaction history).

### 

Appendix 1: Google VM cost estimate

