

# Dungeons & Dragons Game State Manager

SRS

Developed for Retro Rabbit by Optimize Prime



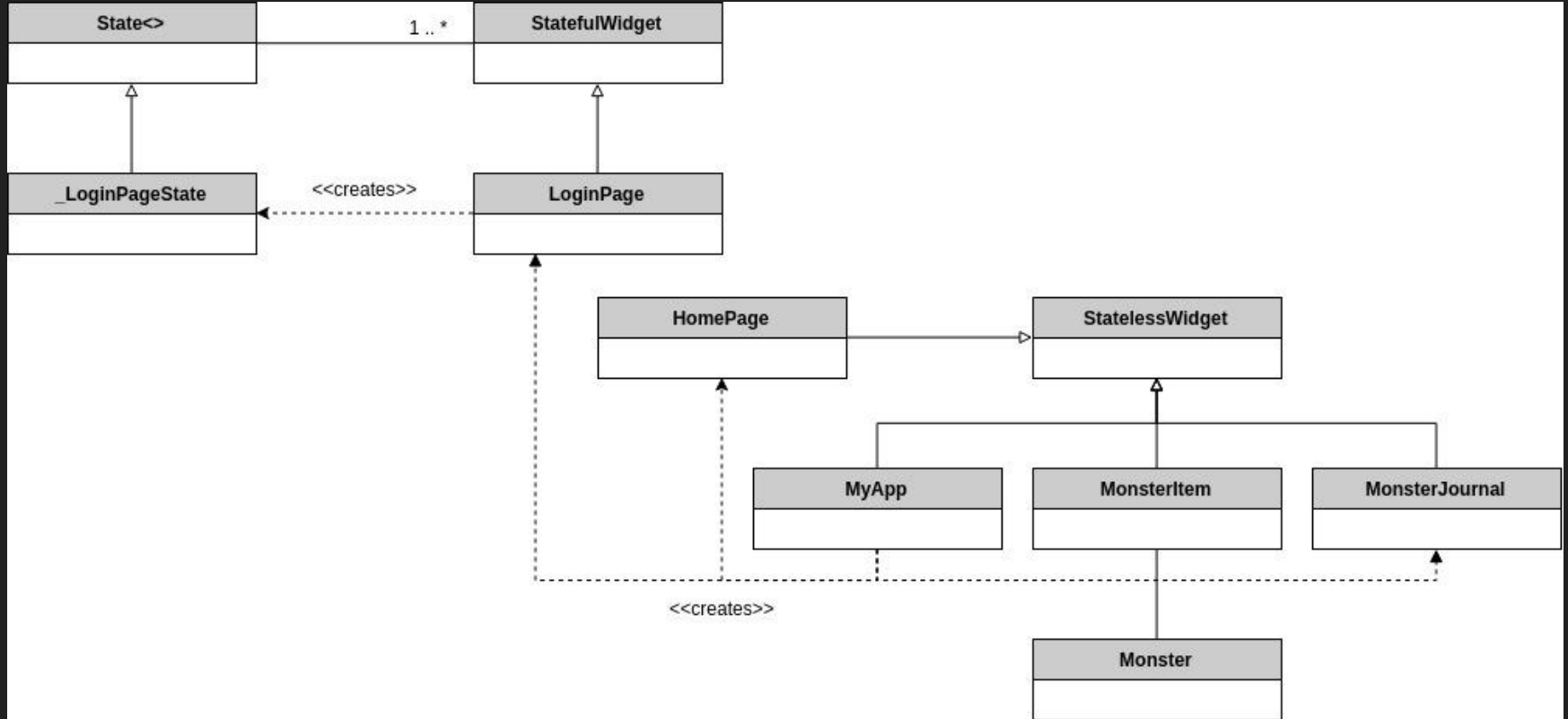
# Overview

The application is intended to act as a helper and assistant when users are playing a game of Dungeons & Dragons. The application tracks information and events of the current game session being played, to provide players with an easier way to see the current state of the game, compared to keeping track manually on paper. It will provide users with an easier way to keep track of their current characters, as well as creating new characters or updating information on existing characters. Users will also have the ability to look at a monster journal, which shows all monsters that have been encountered so far, and allows users to add monsters that they encounter later.

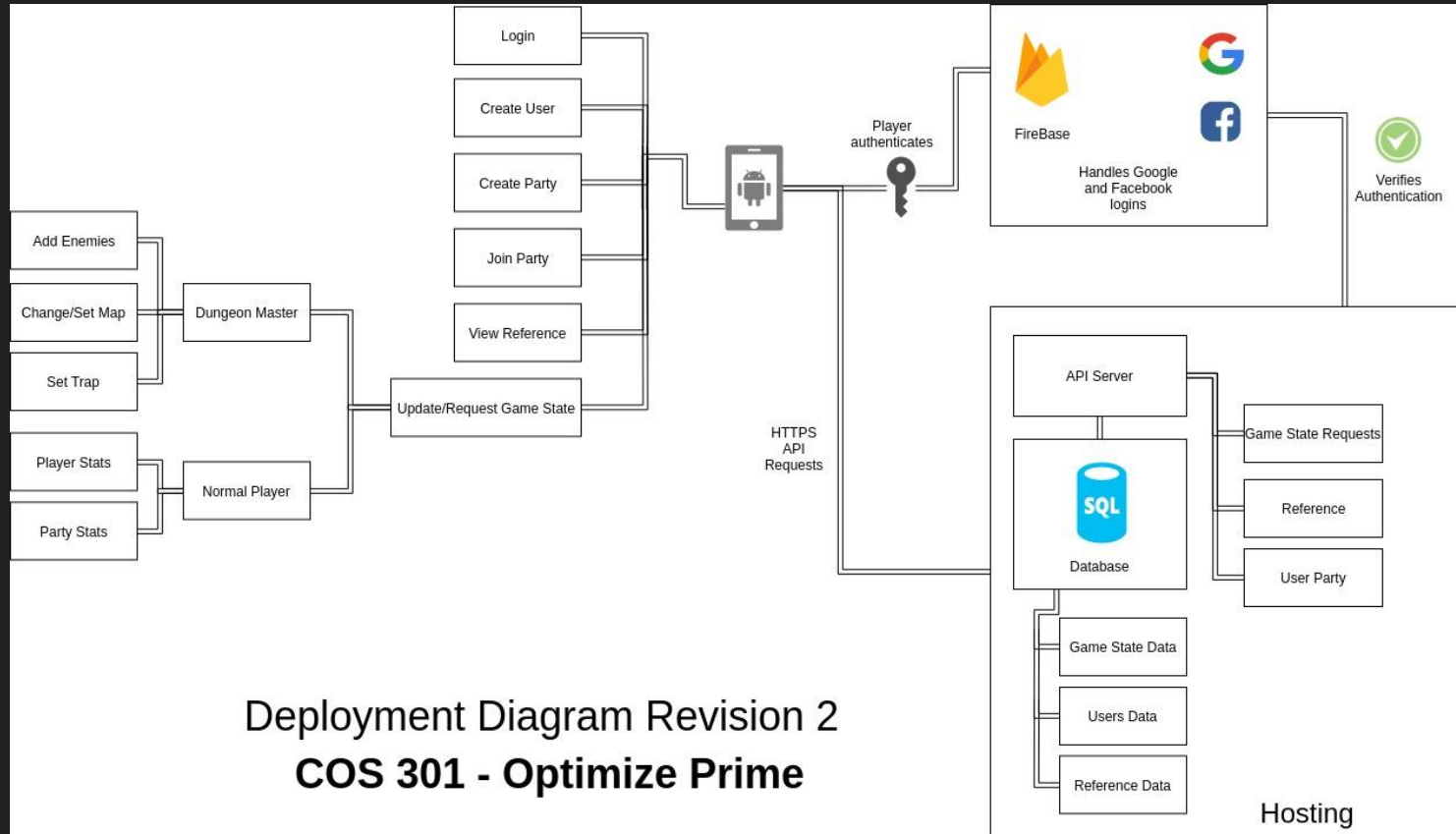
# Functional Requirements

1. Users should be able to create a game session in which they are either the player or game master.
2. Users should be able to view all their characters and create new ones or delete old ones.
3. Users should be able to view all monsters previously encountered and add to them in their monster journal.
4. Users should be notified of events that are triggered by the game master.
5. Users that are game masters should be able to set and trigger events.
6. Users should be able to view and manage their accounts.
7. Users should only be able to join a session that is near them.
8. Users should be able to add friends on the application and filter game sessions by friends.
9. The Application should provide users with informative displays about the current state of the game, such as their characters health.
10. The Application should allow the game master to track the condition of all elements of the game session, such as monsters and their stats.
11. Users should be able to create accounts and login with them or a social account.
12. Users should be able to sign out of their accounts.

# Domain Model of the D&D Application:



# Deployment Diagram:



# Architectural Design Explanation:

Our system uses a custom architecture.

The application serves as the Actor at all times and is controlled by a human.

The application interacts with an external authentication service, Firebase, using an Interactive (“client-server”) architecture to authenticate existing and register new users.

After this authentication process and the user has been verified, the application interacts with the game server.

The application then only interacts with the authentication server again once a sign-out has been requested by the user. The application then communicates with the game server in regards to session requests and server responses.

In this way the game server is similar to an Event-Driven architecture in that it maintains a virtual state for each game session and responses are pushed to clients when an event is triggered.

A more detailed approach of each subsystem of our custom architecture follows.

# Explanation Cont:

## D&D Application:

- The **D&D application** performs a basic authentication request with the users credentials via http to the **Firebase Server**.
- The **Firebase Server** then responds with an **Authentication Token (AuthToken)** or with an empty response, indicating incorrect credentials.
- The **D&D application** also communicates with the **Game Server** to create,join or leave a game session.
- The **D&D application** also requests the users **Monsters** and **Characters** from the **Firebase Database** once the user is verified.
- The **D&D application** is “made up” of user interactive **Screens** populated with **Widgets**.
- The **D&D application** makes queries to the **Firebase Database** to be able to populate itself with **Characters** and **Monsters** specific of the current **User**.
- The **D&D application** receives pushed events when they occur during the game, and updates the game **Screen** as necessary.

# Explanation Cont:

## Firebase:

- **Firebase** is Google's free authentication and database management system, complete with an API for communicating securely with the service.
- **Firebase** interactions take place in a "Client-Server" manner and therefore this subsystem follows an Interactive architecture with the **D&D application**.
- **Firebase** is used by the **D&D application** to authenticate users with received credentials via http requests.
- **Firebase** then attempts to find a matching user in its user base and return that **Users** unique **Authentication Token (AuthToken)**.
- **Firebase** also holds the databases for **Monsters** and **Characters** created by Users of the **D&D application**.
- These **databases** contain data on the item (JSON files) as well as the **Authentication Token** of the **User** who created it. This applies to **Monsters** and **Characters**.
- This **Firebase** Database service returns these items to the **D&D application** when they are requested.



# Explanation Cont:

## Firebase Database:

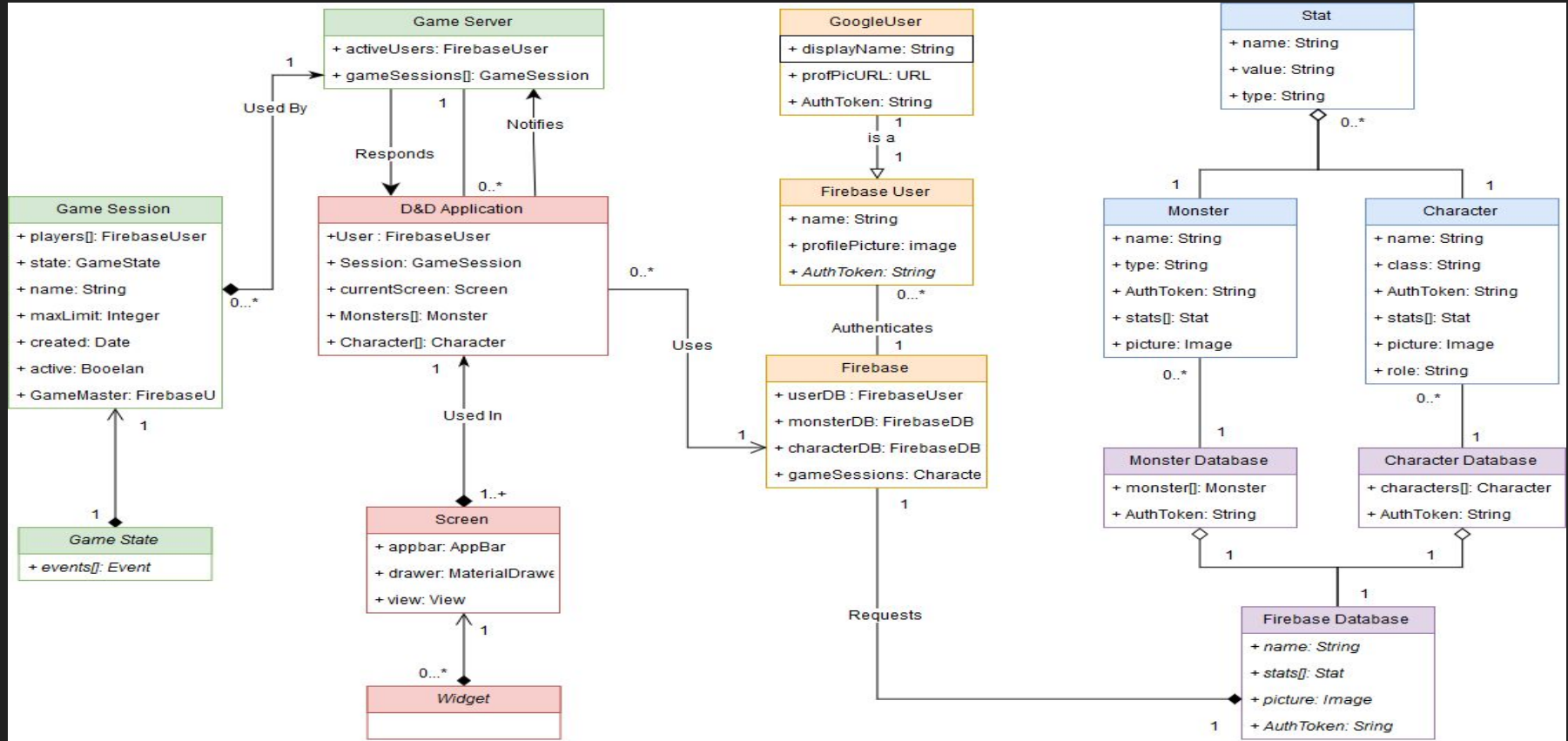
- The **Firebase Database** holds a database of **Users** and their credentials as well as their **Authentication Token (AuthToken)** which is handled internally in **Firebase** systems.
- The **Firebase Database** holds two important **databases**; one being for **Characters** and the other for **Monsters**.
- Each entry into either **database** consists of the **Authentication Token** of a **User** and a JSON file which encapsulates the data of the actual item.
- This User-Data pairing allows only select **Characters** and **Monsters** to be available to a specific User.
- The **Firebase Database** interacts with the **D&D application** using the **Firebase API** to query the database.

# Explanation Cont:

## Game Server:

- The **Game Server** does not respond in typical interactive “Client-Server” style and instead follows a virtualized Event-Driven architectural approach. That is to say that each **Game Session** is treated as a virtual server that pushes information to Users **D&D applications** when an event is triggered or a criteria met.
- The **Game Server** responds directly only on select requests, such as a list of available **Game Sessions**, and this response depends on the current state of the communication between the **D&D application** and the **Game Server**.
- The **Game Server** creates, clears and manages **Game Sessions** as well as the **Users** that are connected to them.
- The **Game Server** must ensure only a specific amount of **Users** may enter a specific **game session**.
- The **Game server** must also enforce joining policies if they are in place for each session (like a password).

# Low-Level Architectural Overview:



# Architectural Model Postface:

The architectural diagram is completed with detail to allow one to follow the workings and reasoning for each subsystem in the architecture. This is because one can always identify the higher level concept from a lower level diagram but the opposite is not always the case. In this mindset we have provided a diagram that allows one a glimpse into the basic workings of our custom architecture in the hopes that it will help one to follow the flow of information and the order of interaction between each subsystem whilst the user traverses and utilizes the application.\*

\*This diagram is based on the current state of the application and will be updated as the application evolves to meet the changing requirements.