

# The Game of Set



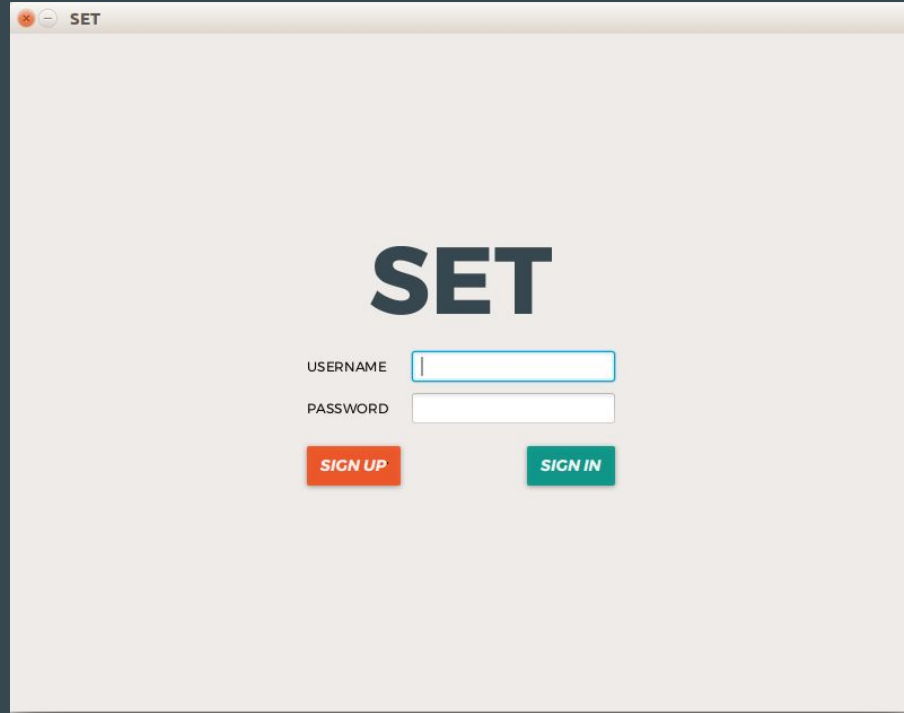
Yash Sharma, Sahil Patel | Shalin Patel, Kevin Sheng

# Design Goals

Goals:

- Robust & Reliable
- Smooth User Experience
- Modularity

# Authentication



A screenshot of a web browser window with the title 'SET'. The page features a large, bold 'SET' logo in the center. Below the logo are two input fields: 'USERNAME' and 'PASSWORD'. The 'USERNAME' field is highlighted with a blue border. Below the input fields are two buttons: a red 'SIGN UP' button and a green 'SIGN IN' button.

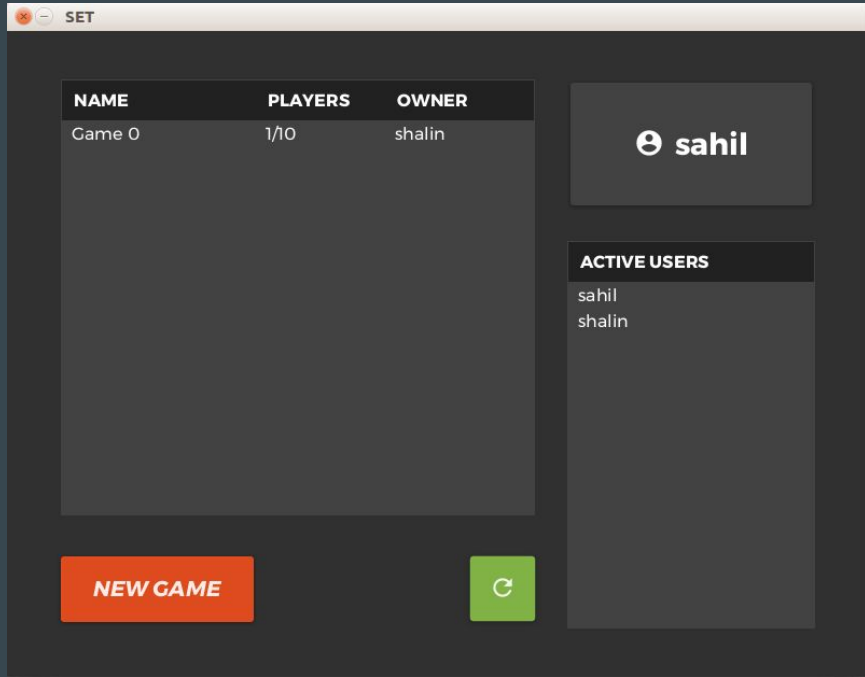
SET

USERNAME

PASSWORD

[SIGN UP](#) [SIGN IN](#)

# Lobby + Room



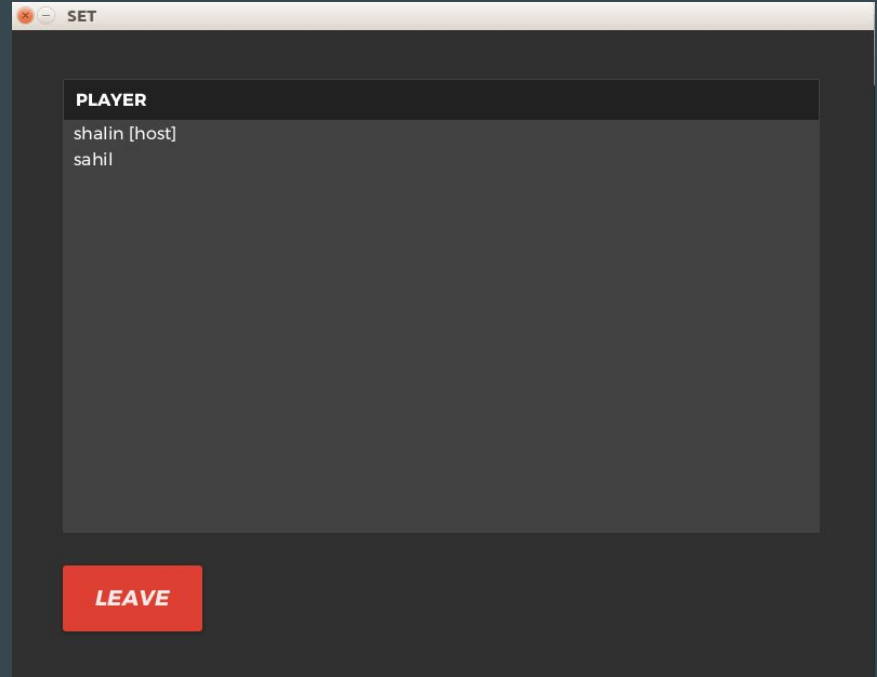
The lobby interface displays a table with columns for game name, player count, and owner. It also shows the current player's name 'sahil' and a list of active users.

NAME	PLAYERS	OWNER
Game 0	1/10	shalin

**ACTIVE USERS**

- sahil
- shalin

Buttons: **NEW GAME** (orange), **Refresh** (green)



The room interface shows a list of players in the current room.


















**PLAYER**

- shalin [host]
- sahil

Button: **LEAVE** (red)

# Game

SET

**SCORE**

shalin	0
sahil	0

**SURRENDER** **SET**

# Client-side Decisions

- JavaFX8 vs Swing
- Minimizing Logic
  - Client avoids logic as much as possible to avoid inconsistency.
  - UI elements send messages and updates are performed based on data in received responses.
- Multithreading (Task and runLater)
  - All game and window state updates are handled in one *Task*, which runs outside of the main GUI thread.
    - Allows updates without freezing UI.
    - Task is within its own class, separating UI from logic and data.
    - Manipulates 4 scenes, each its own object.
  - Within this task, UI updates are called with *Platform.runLater*, which allows adds the update into a queue for the main GUI thread to run as soon as possible.

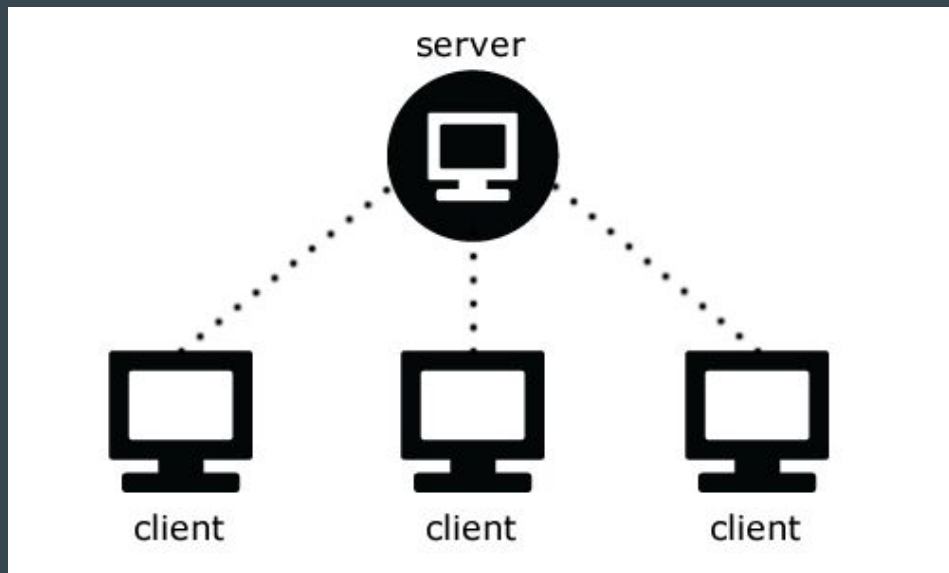
# Database Decisions

- MySQL
- Hand out connection for every interaction
- Created objects to simplify queries into function calls



# Server-side Decisions

- Multithreaded design
  - Main process
  - Player thread
  - Game thread
  - Thread communication via pipes
- Class to store player info
  - username
  - Input/Output stream objects
  - References to pipes between player & game threads





# Server-Client Communication Decisions

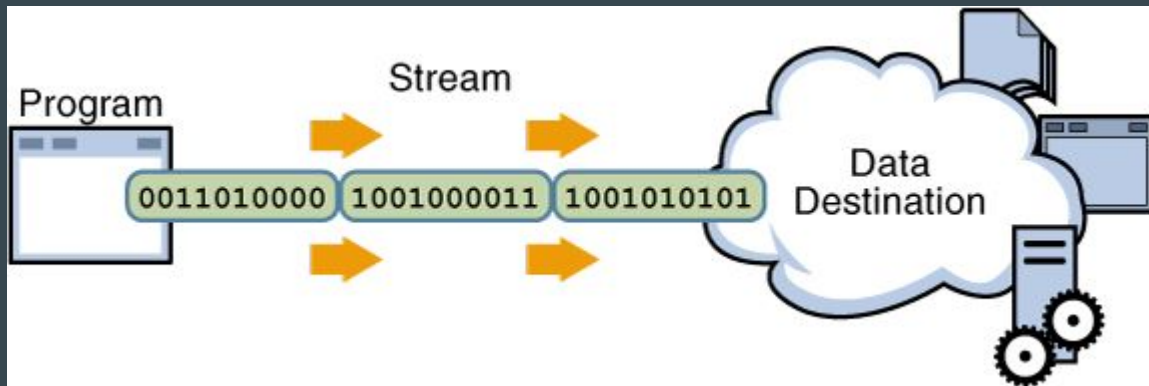
Publish-Subscribe:

- Server *publishes* messages to users *subscribed* to that specific event
- Ex: When a user clicks on the *SET* button, the user sends out a `SetSelectMessage`, and all users currently in the game receive a `SetSelectResponse`.

# Server-Client Communications

Messages/Responses:

- Object Hierarchy approach
  - Each Message is its own class that inherits from a *Sendable* interface
  - Allows for easy sending to clients/server



Questions?