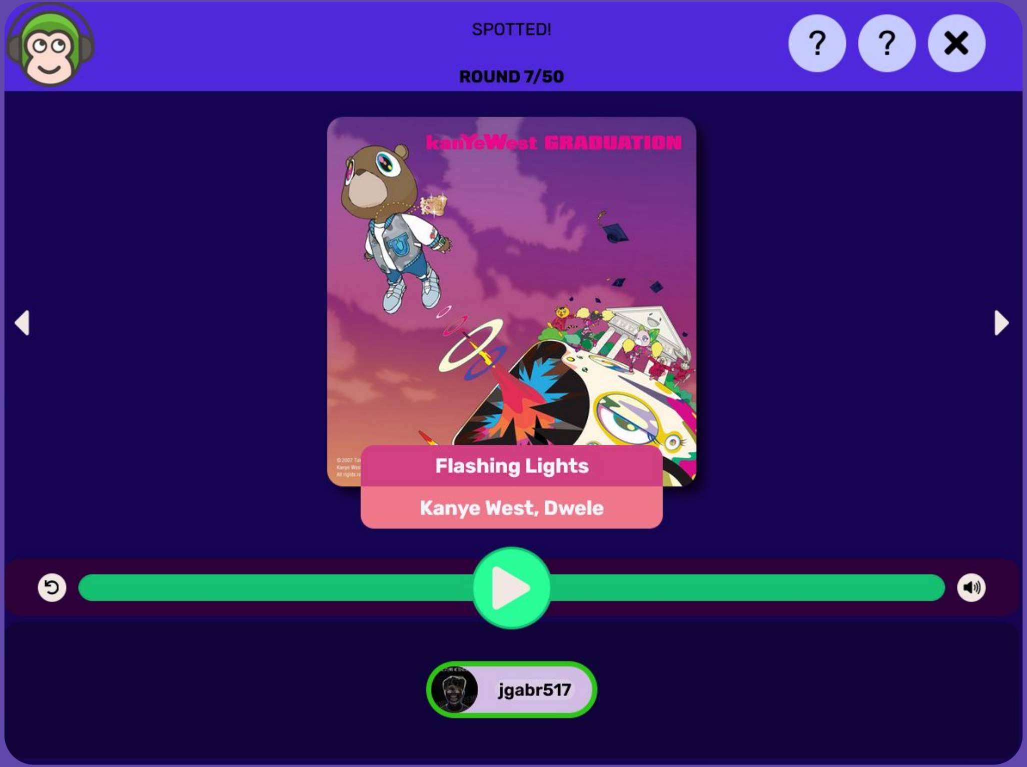
SPOTTED

By: Jesse Gabriel & Drew Schiller

What Is SPOTTED?



- A game that involves your music taste and how well you know your friends.
- Each Player joins by linking their Spotify Account.
- Each round, a new song is selected. The goal is to guess which one of your friends has the song in their playlist.

FEATURES: Multiple game-modes & adjustable settings

DEVELOPMENT PLAN

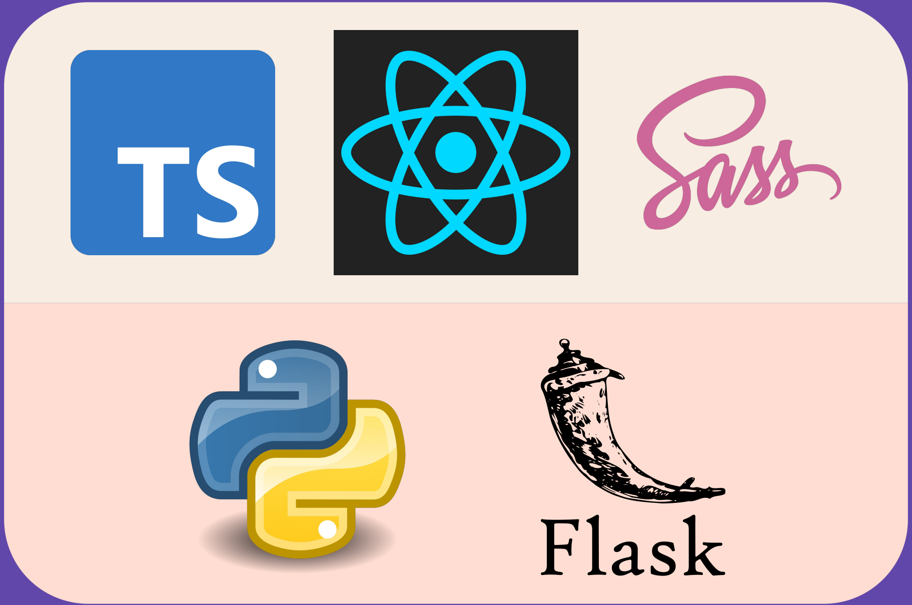
Front-end: Jesse

- Directory and component structure
- Design and styling
- Both involved with linking and general vision for the project

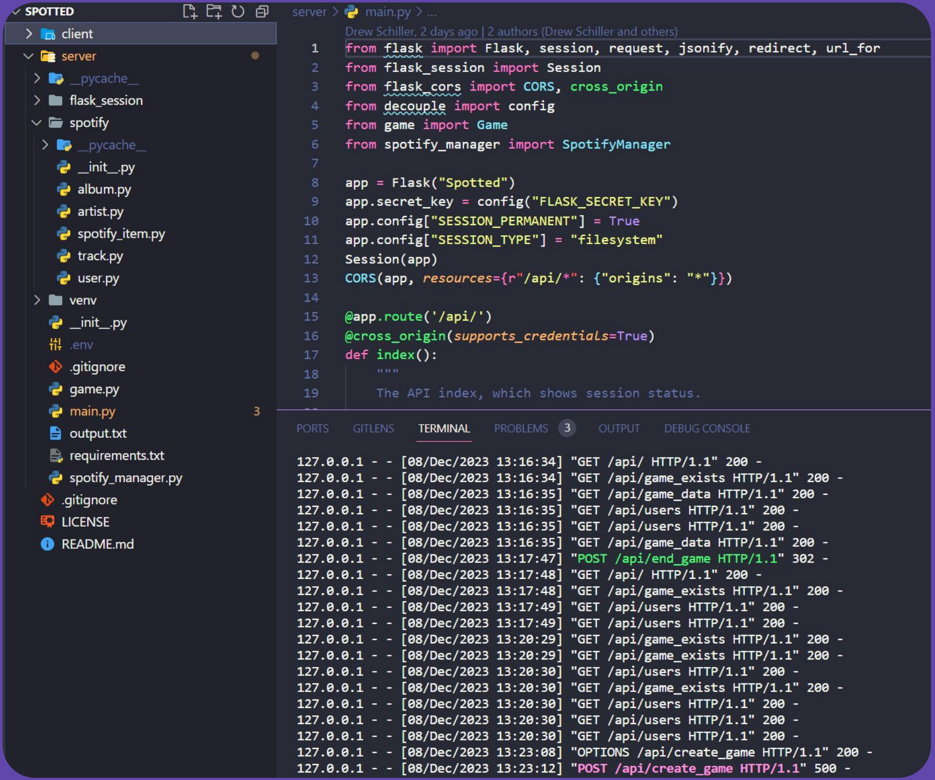
Back-end: Drew

- Connection to Spotify API
- Game Session Management
- Custom API for frontend
- Both involved with linking and general vision for the project

TECH STACK



THE BACKEND (API)



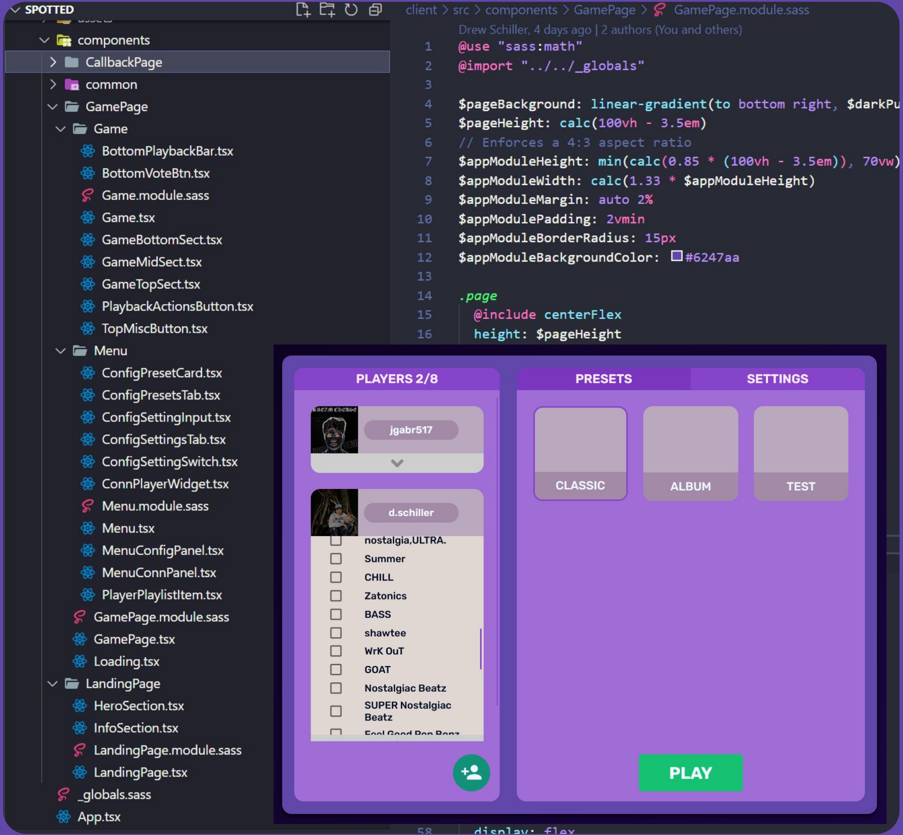
```python
from flask import Flask, session, request, jsonify, redirect, url_for
from flask_session import Session
from flask_cors import CORS, cross_origin
from decouple import config
from game import Game
from spotify_manager import SpotifyManager

app = Flask("Spotted")
app.secret_key = config("FLASK_SECRET_KEY")
app.config["SESSION_PERMANENT"] = True
app.config["SESSION_TYPE"] = "filesystem"
Session(app)
CORS(app, resources={r"/api/*": {"origins": "*"}})

@app.route('/api/')
@cross_origin(supports_credentials=True)
def index():
    """
    The API index, which shows session status.
```

```
127.0.0.1 - - [08/Dec/2023 13:16:34] "GET /api/ HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:16:34] "GET /api/game_exists HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:16:35] "GET /api/game_data HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:16:35] "GET /api/users HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:16:35] "GET /api/users HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:16:35] "GET /api/game_data HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:17:47] "POST /api/end_game HTTP/1.1" 302 -
127.0.0.1 - - [08/Dec/2023 13:17:48] "GET /api/ HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:17:48] "GET /api/game_exists HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:17:49] "GET /api/users HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:17:49] "GET /api/users HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:20:29] "GET /api/game_exists HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:20:29] "GET /api/game_exists HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:20:30] "GET /api/users HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:20:30] "GET /api/game_exists HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:20:30] "GET /api/users HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:20:30] "GET /api/users HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:20:30] "GET /api/users HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:23:08] "OPTIONS /api/create_game HTTP/1.1" 200 -
127.0.0.1 - - [08/Dec/2023 13:23:12] "POST /api/create_game HTTP/1.1" 500 -
```

- Backend uses Spotipy library for API calls to Spotify
- Stores game and Spotify data in sessions
  - Sessions are encrypted and cached per instance of the site
- Client manages backend game state using Flask API calls
- Currently, session data is stored on our filesystem, we have plans to implement a database

THE FRONTEND

- Mainly built with React and SASS
- Reads session data using calls to backend Flask API
- Large number of components required the development of specific organization schema involving file structure, and naming convention
- SASS modules used for styling UI with modularity and level scoping of variables



```
SPOTTED
  components
    CallbackPage
    common
    GamePage
      Game
        BottomPlaybackBar.tsx
        BottomVoteBtn.tsx
        Game.module.sass
        Game.tsx
        GameBottomSect.tsx
        GameMidSect.tsx
        GameTopSect.tsx
        PlaybackActionsButton.tsx
        TopMiscButton.tsx
      Menu
        ConfigPresetCard.tsx
        ConfigPresetsTab.tsx
        ConfigSettingInput.tsx
        ConfigSettingsTab.tsx
        ConfigSettingSwitch.tsx
        ConnPlayerWidget.tsx
        Menu.module.sass
        Menu.tsx
        MenuConfigPanel.tsx
        MenuConnPanel.tsx
        PlayerPlaylistItem.tsx
      GamePage.module.sass
      GamePage.tsx
      Loading.tsx
    LandingPage
      HeroSection.tsx
      InfoSection.tsx
      LandingPage.module.sass
      LandingPage.tsx
  _globals.sass
  App.tsx
```

client > src > components > GamePage > GamePage.module.sass

Drew Schiller, 4 days ago | 2 authors (You and others)

```
@use "sass:math"
@import "../../_globals"

$pageBackground: linear-gradient(to bottom right, $darkPu
$pageHeight: calc(100vh - 3.5em)
// Enforces a 4:3 aspect ratio
$appModuleHeight: min(calc(0.85 * (100vh - 3.5em)), 70vw)
$appModuleWidth: calc(1.33 * $appModuleHeight)
$appModuleMargin: auto 2%
$appModulePadding: 2vmin
$appModuleBorderRadius: 15px
$appModuleBackgroundColor: #6247aa

.page
  @include centerFlex
  height: $pageHeight
```

□ nostalgia,ULTRA.
□ Summer
□ CHILL
□ Zatonics
□ BASS
□ shawtee
□ WrK OuT
□ GOAT
□ Nostalgiac Beatz
□ SUPER Nostalgiac Beatz

DEMO

CURRENT LIMITATIONS

- Liked song fetching limited
- Doesn’t work with other Spotify accounts not in the developer console
- Artist info limited
- Not many presets yet
- Session data is currently stored on filesystem

WHAT WE LEARNED/CHALLENGES

We Learned A LOT

- How to structure a full-stack web application
- A Significant amount of CSS (SASS)
- Working with React (Hooks, Contexts, useRef...)
- Using Git for productivity (Branches, merging & issues)

Challenges Faced:

- Getting Authentication to work
- CSS Challenges: Scrolling text animation, container aspect ratio

FUTURE PLANS/GOALS

Bug Fixes

- Handling Errors Better, Fix current limitations
- Reorganizing Codebase

Features:

- Game Modes involving:
  - Selection Voting: “Who doesn’t listen to __”
  - Statistics: “Who listens to artist/song the most?”
- Design: finishing up styling, animations …
- And much more.

THANK YOU!

Questions?