Pop Up – Full Stack Application

Benjamin Michael, Gerardo Cano, Clark He • Vida Vakilian • COMP-499

What is Pop Up?

Pop Up is a full-stack application with a mobile app as the frontend. The purpose of the app is to allow users to easily find pop up events near them. These can be anything from garage sales, food trucks, sporting events, or anything else that is happening for a short period of time.

Why?

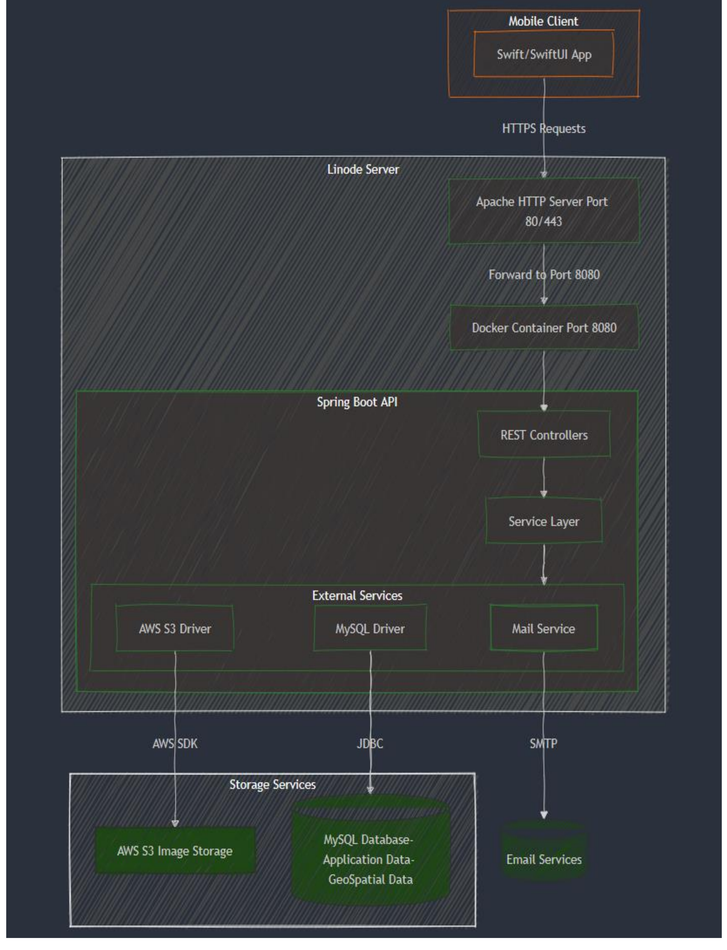
Currently, there isn't an app that is a household name used for finding pop up events, and Google Maps does not work well for short-term or mobile events. It is common to see street posts plastered with signs that advertise "Yard Sale", but oftentimes those events are already over. We aim to use modern technology to bring an application to users which would make advertising or finding events like this much easier and more intuitive. There is no reason that you should have to spend an hour stapling signs to street posts or posting on Facebook– within a few quick steps you can instead upload your event to Pop Up, and anyone around town can see it.

Backend Technologies

Overview of the main technologies used in development of the backend API:

- Spring Boot framework and Java were used to develop the REST API. Spring Boot is a powerful framework that supercharges Java development by eliminating the boiler plate configuration and providing production-ready features out of the box. In addition, it has integrated drivers (JDBC) for communicating with databases and is containerization friendly.
- To run the backend API we leveraged Docker to containerize the server and easily deploy it. Using docker allowed us to integrate the deployment of our server with GitLab CI/CD.
- For persistence, multiple facets of storage were required. Not only do we need to store general user information and event data (which can be easily translated to an efficient schema) but we also need the ability to store and server images. Images are an integral part of Pop Up, but databases are not very efficient for images. Therefore, we settled on two storage services
  - MySQL
    - A common relational database, MySQL was used to store the entities needed for the user/post layout
  - AWS S3
    - S3 was used for storing and serving images. It is very efficient storage service for images and highly scalable. In addition, with developer friendly APIs, all that was required of us was to create a driver to interact with S3 for our image needs.

Architecture

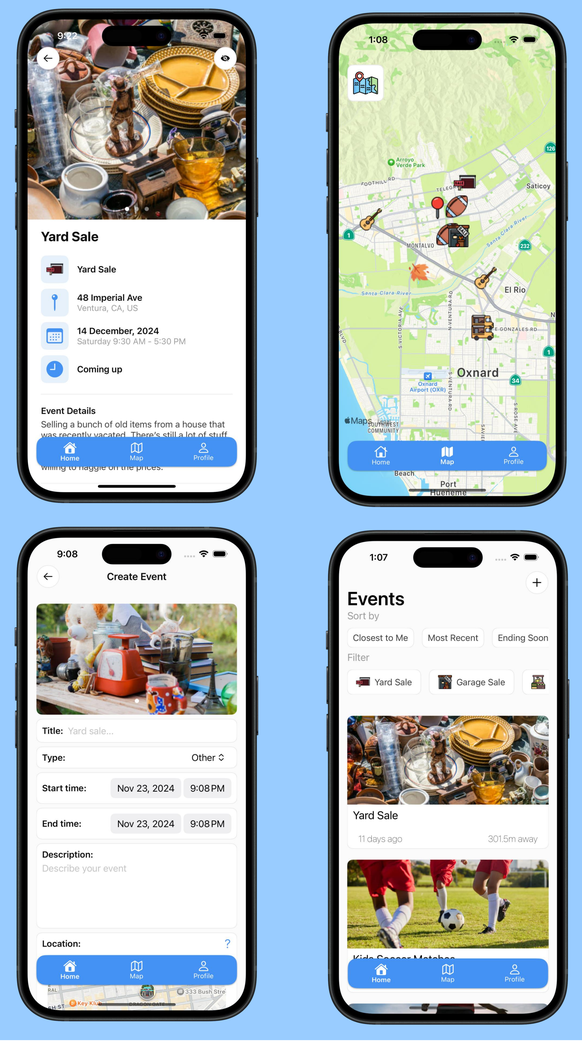


Architecture Notes

- The mobile application talks to the REST API via https requests
  - A custom API and HTTP client wrote in Swift handle this
- REST API runs on a Docker container on private server
  - Configured server to use Apache2 HTTP server
  - Port forwarding rules to forward any HTTPS traffic to the docker container that is running on port 8080
- REST API Layers:
  - Presentation Layer
    - Includes the controllers (endpoints open on the API)
    - What the public can use and interact with
  - Business Layer
    - Includes the services that handle the logic of taking data from the user via the controllers and talking to the data layer
  - Persistence (Data) Layer
    - Handles persisting data

Mobile App UI

Snippets of the more important screens of the UI (does not include all screens in the app)



Future Improvements

- Adding scalability to the backend
  - Currently, we only utilize one server to handle requests. While this is perfectly sustainable for the size of the project, implementing a scalable structure that can deploy new instances when traffic throughput is high would be crucial if the application was deployed in production.
- Database sharding
  - Although more complex, sharding would allow us to reduce the stress on a single database instance and split the load between multiple instances