Collaborative Surveillance of Large Geographic Area by Fleet of Drones

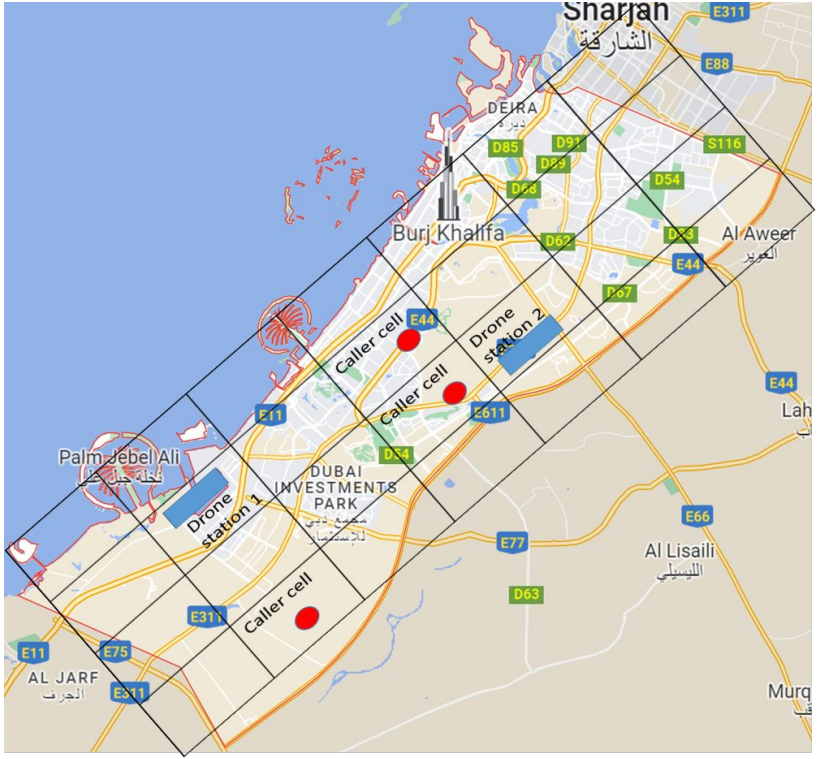
sdmay23-50

Advisor: Professor Goce Trajcevski
Graduate Student: Prabin Giri

Rowan Collins, Joe Edeker, Jaden Forde, Thomas Glass, Jacob Houts, Marcus Jakubowsky

Project Introduction

- Drone Fleet Algorithms
  - Events & Response
- Simulation vs Real-world
- Customizable Setup/Input & Algorithms
- Visualize Runs & Drone Paths
- Request Queue System

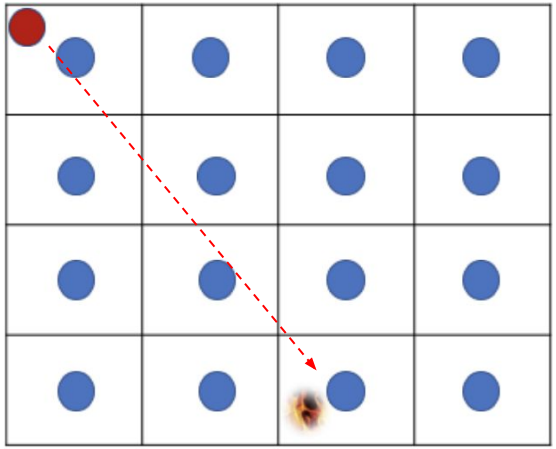


Final Report, Section 1.2

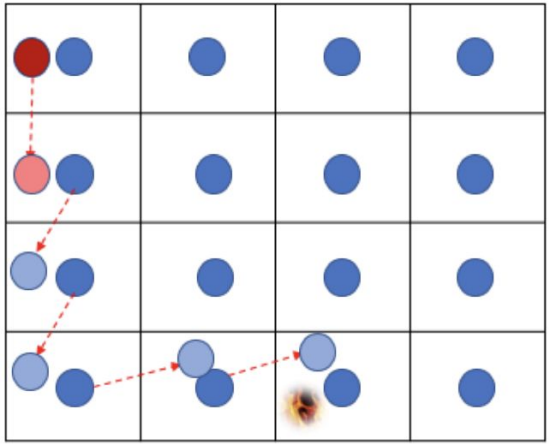
Drone Fleet Algorithms Overview

- Algorithms assume a square grid divided into zones, one drone per zone
- Additional roaming drones fly around to assist scanning
- Goal: When a drone detects an event, roaming drone flies to assist
- 2 Algorithms Provided
  - Naive
  - Coordinated Path-Hop (CPH)

Naive

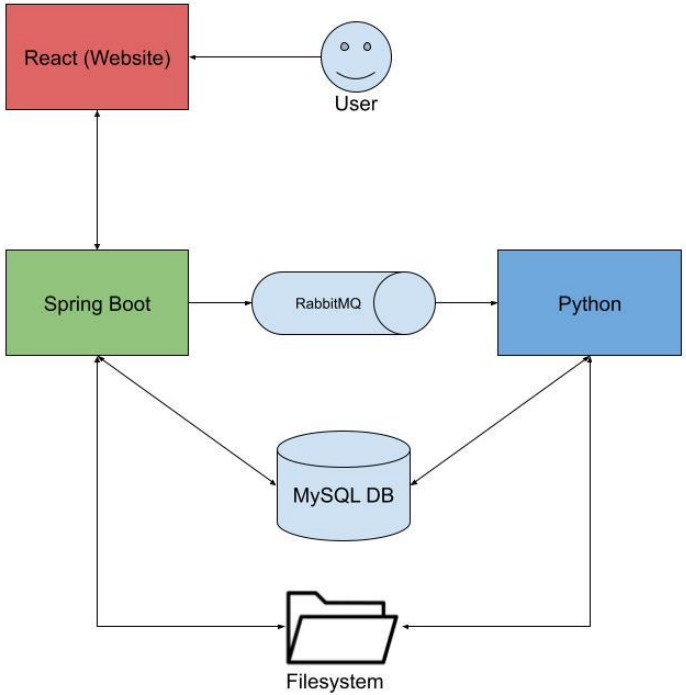


CPH



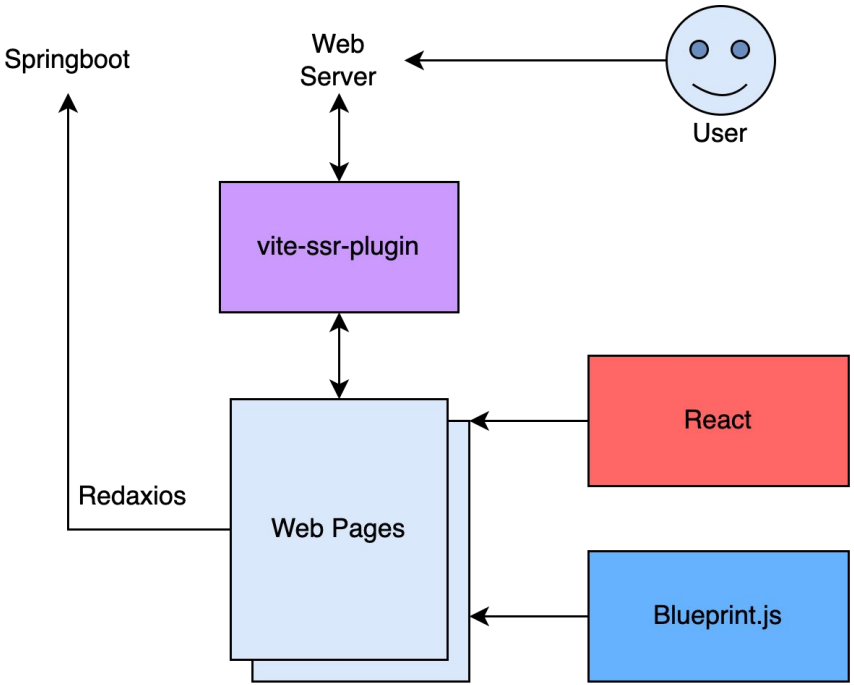
Implementation Architecture

Implementation Architecture (System)

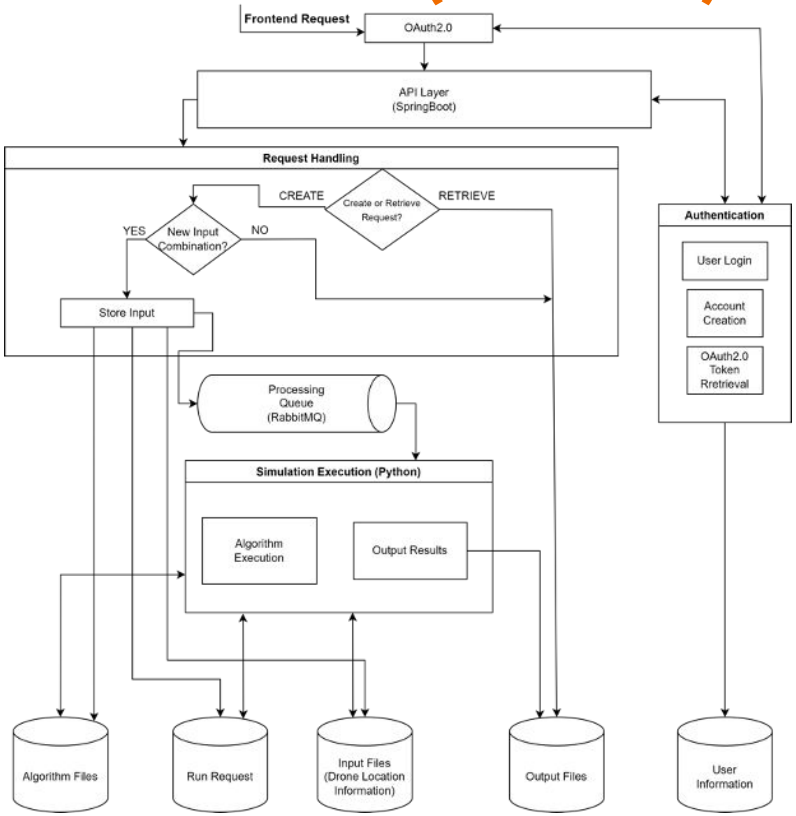


Final Report, Section 2.1

Implementation Architecture (Frontend)



Implementation Architecture (Backend)



Final Report, Sections 1.5, 2.2

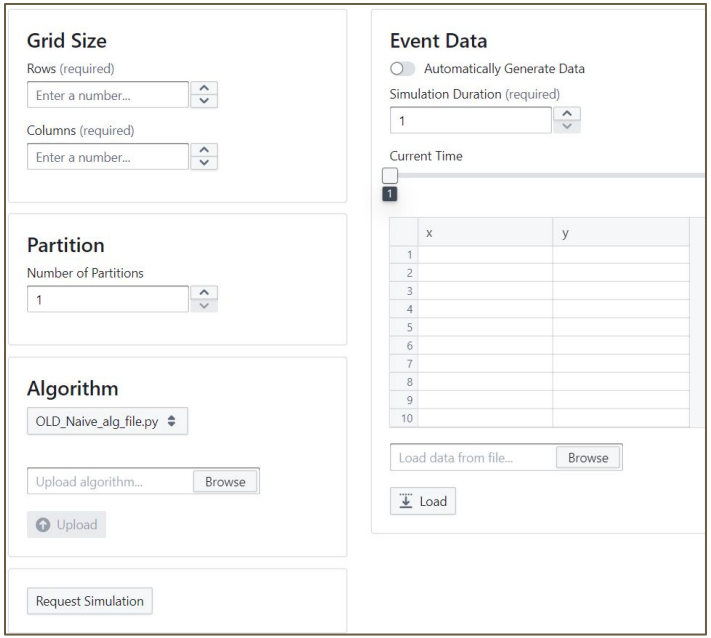
Work Accomplishments

Work Accomplishments (Frontend)

Simulation Setup:
- Ability to upload algorithms
- Send run request with specified parameters
- Import data from csv

Simulation Visualization:
- Grid to view drones
- Algorithm comparison

Dashboard:
- Web application navigation
- Display request queue and completion data

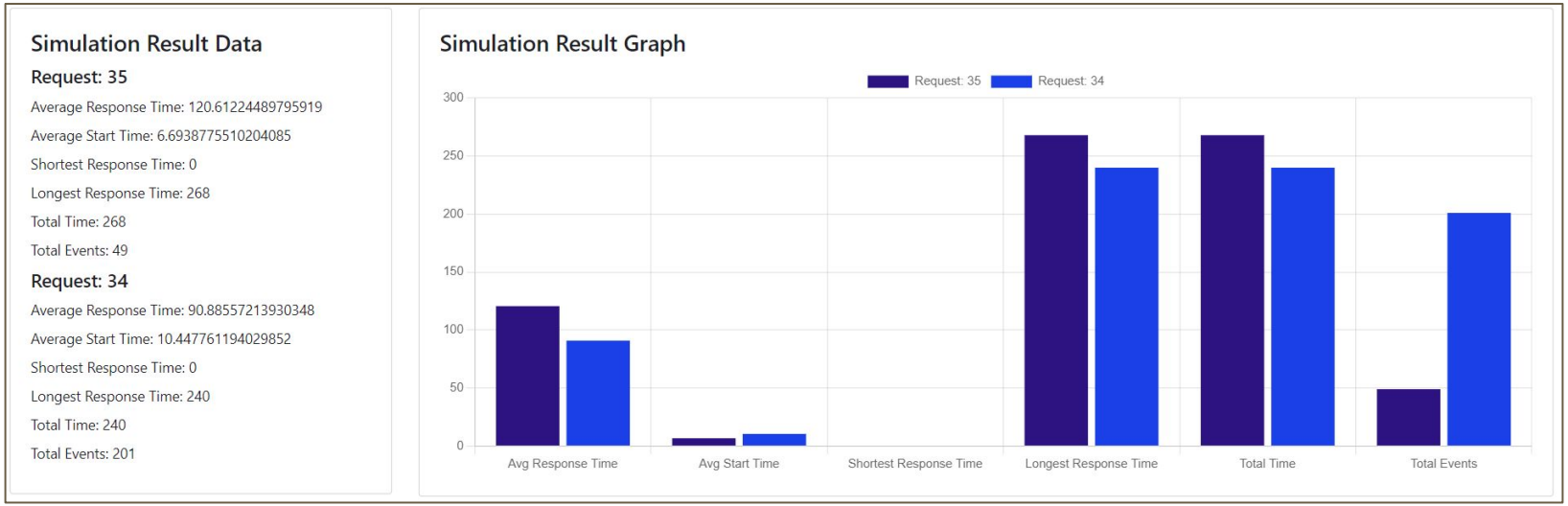


Final Report, Section 2.3

Work Accomplishments (Frontend)

Simulation Comparison:

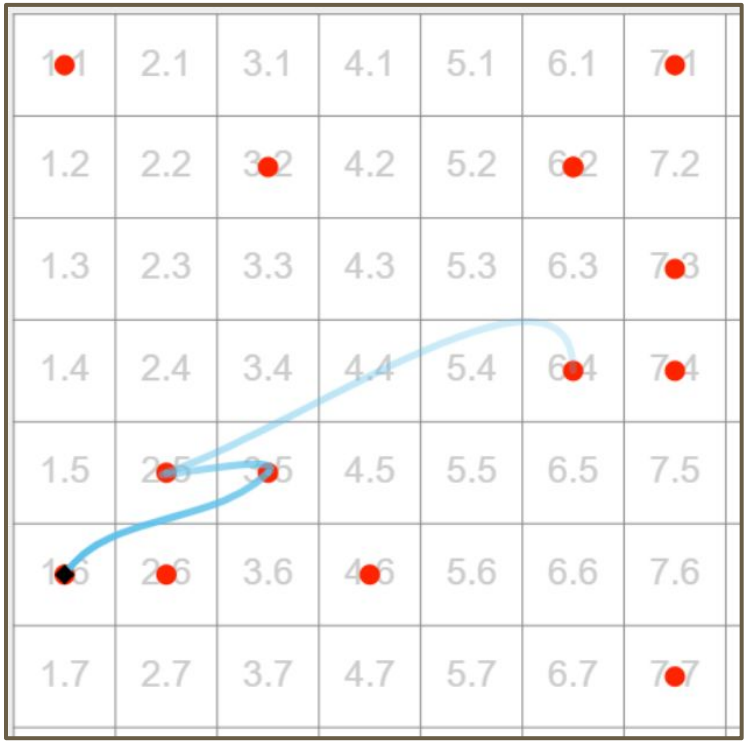
- Comparing drone simulation outputs side by side.
- View difference in success criteria for 1 to many Outputs.



Work Accomplishments (Backend)

Spring Boot:
- User signup & login
- Algorithm file upload
- Run simulation requests
- Enqueue requests
- Simulation status, output & pathing

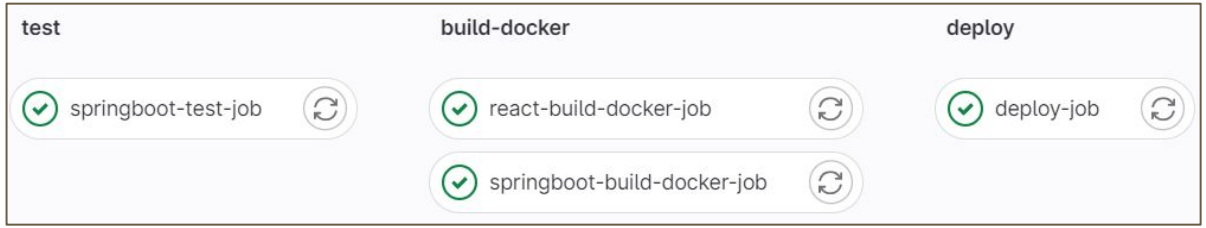
Python:
- Receive queued requests
- Drone algorithm execution
- Save output to database
- Create flight paths

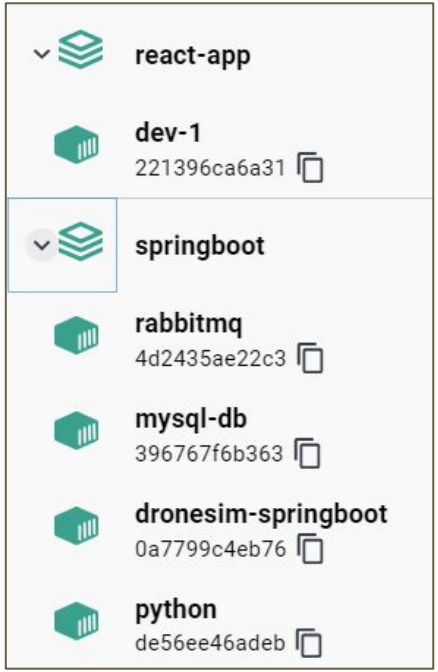


Work Accomplishments (Other)

Other:

- Tech stack running in Docker
- CI/CD pipeline automatically deploys changes
- RabbitMQ initialization for queue system
- Token authentication w/Spring Security





Web Application Visuals

Dashboard

Dashboard | File | Logout

Simulation

Setup a new Simulation

View Simulation execution

Compare Simulations

Completed Simulations

Simulation Request #2

Algorithm File ID: 10

Request Time: 4/25/2023, 12:10:19 AM

Execution Start Time: 1/1/1970, 12:00:00 AM

Execution Completion Time: 4/25/2023, 12:10:24 AM

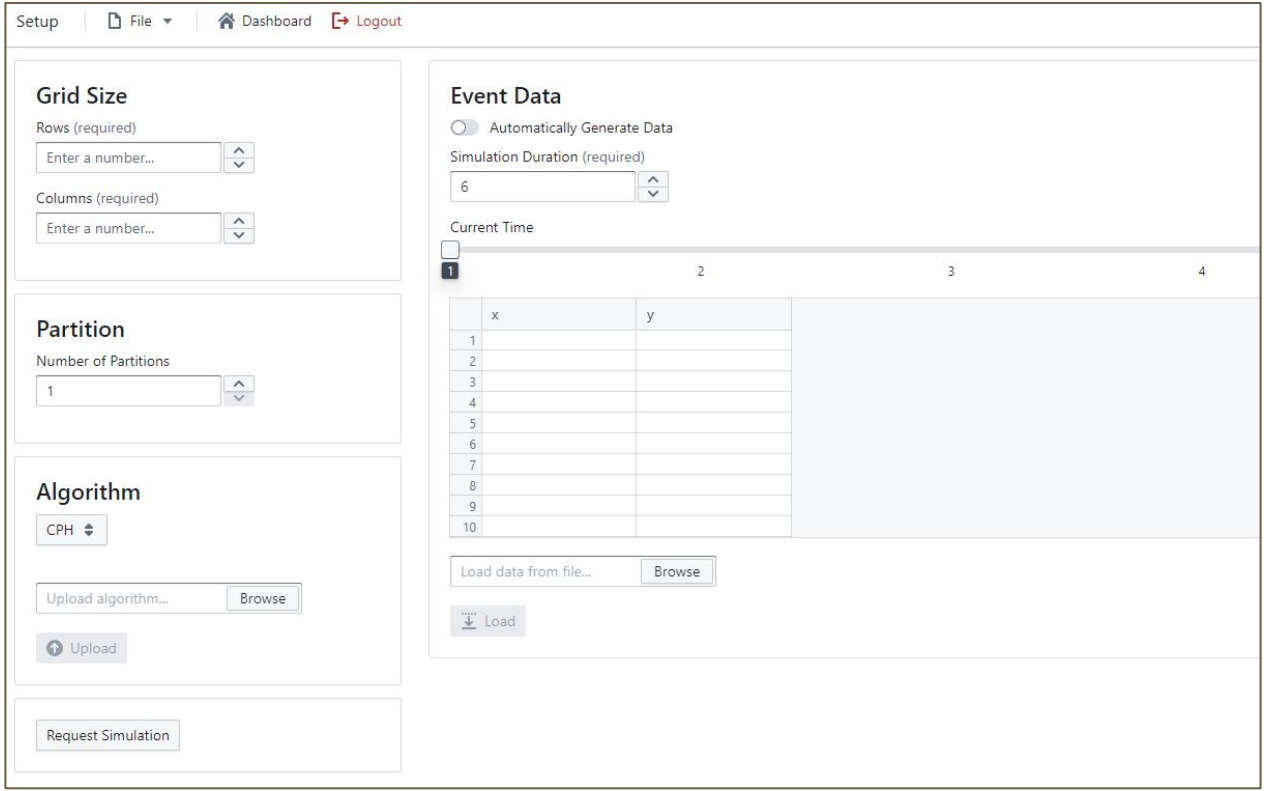
← Previous | 1 of 2 | → Next

Queued Simulations

No Data

No data is available at this time.

Simulation Setup



Simulation Comparison

View | File | Dashboard | Logout

Simulation Result Data

Request: 2

Average Response Time: 5

Average Start Time: 1

Shortest Response Time: 0

Longest Response Time: 10

Total Time: 10

Total Events: 2

Request: 1

Average Response Time: 3.5

Average Start Time: 0.5

Shortest Response Time: 0

Longest Response Time: 7

Total Time: 7

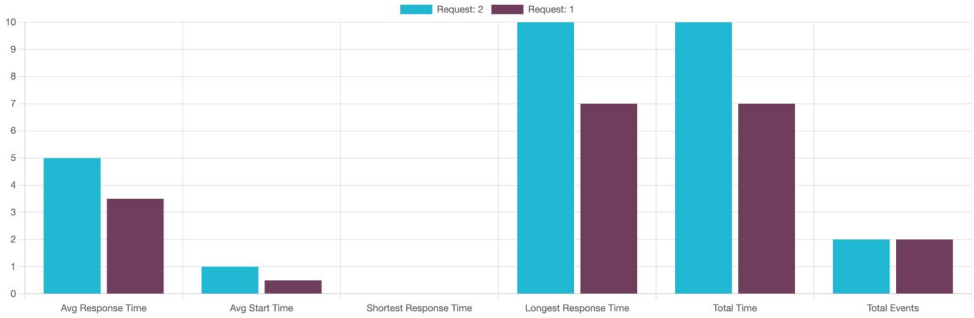
Total Events: 2

Load data from file... Browse

Add Result Data

Reset

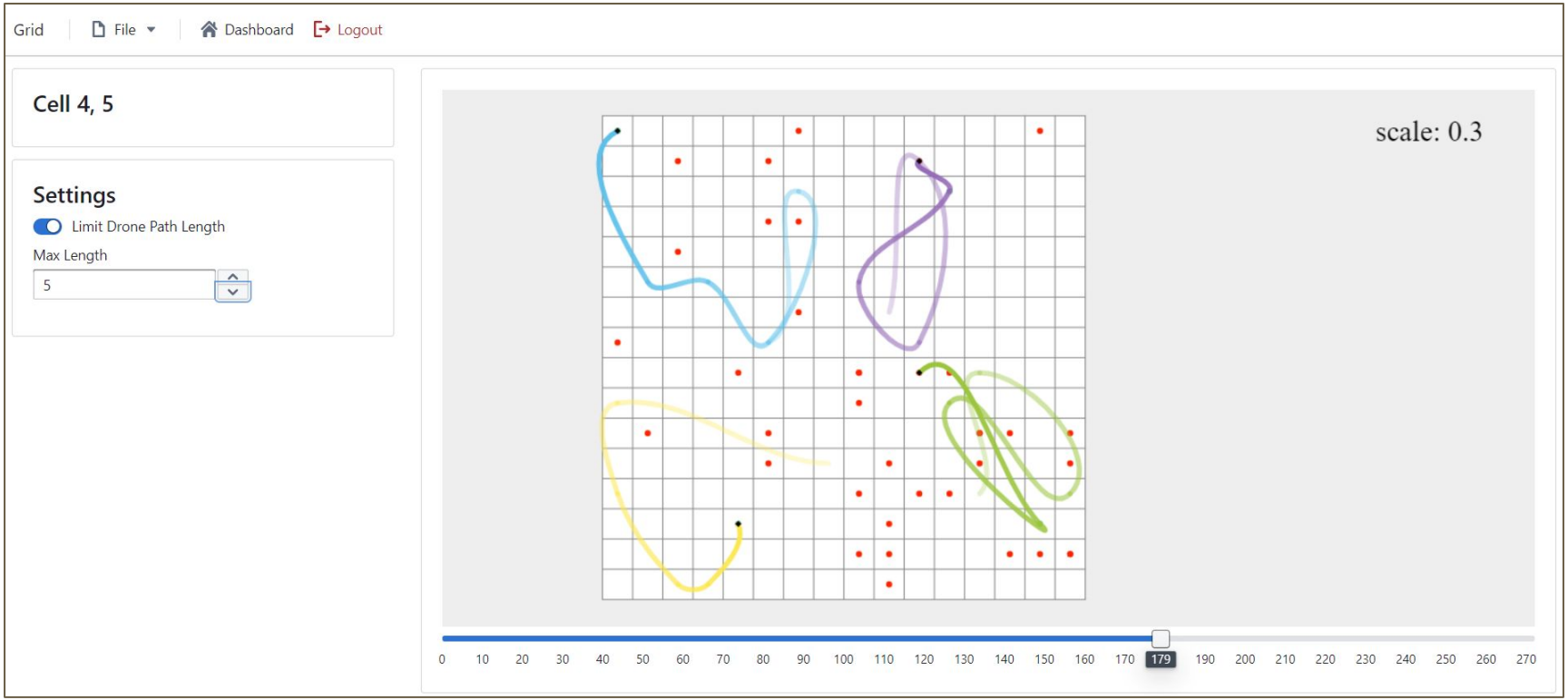
Simulation Result Graph



Result: 2 Result: 1

	x	y	Request Time	Response Time
1	0	0	0	0
2	1	1	2	12

Simulation Visualization



Team Member Contributions

Key Contributions

Rowan Collins
- *Infrastructure:* Database Initialization
- *Spring Boot:* File uploading, File storage, Code Refactoring
- *Testing:* Spring Boot - Algorithm Files, Input File Service

Joe Edeker
- *Frontend*: Vite (Development, Routing), React + Blueprint, Layouts, Navigation, Setup, View, Grid Visualization

Jaden Forde
- *Python*: RabbitMQ integration, drone algorithm file modification, input loading, algorithm execution, database result logging

Key Contributions

Thomas Glass
- *Frontend:* Setup and Dashboard (React + Blueprint)
- Drone Algorithm Research

Jacob Houts:
- *Infrastructure:* File storage
- *Spring Boot:* Initial Setup, File uploading, Run Initialization, Code Refactoring
- *Frontend:* Simulation Comparison Page, Session Storage, Uploading local I/O
- *Testing:* Spring Boot - Run Requests

Marcus Jakubowsky
- *Spring Boot:* Users, Algorithms, Input, Output, Run Requests, Queue, Path, Authentication
- *Python:* Path mapping
- *Frontend:* Signup, Login, Request authorization
- *Infrastructure:* Docker, RabbitMQ, CI/CD, Server
- *Testing:* Spring Boot - Algorithms, Users, Postman

Challenges & Solutions

Challenges and Solutions

Frontend:

- Page routing
  - Vite-SSR-Plugin
- UI Components
  - Blueprint
- Drone path visuals
  - Custom Polynomial Curve Calculations
- API Request Authentication
  - CORS implementation in Spring

Challenges and Solutions

Backend:

- Running Python algorithms
  - Created Python backend service to handle
- Python algorithm files not standardized (hardcoded)
  - Refactor initial given algorithms & create compliance standard
- Queuing requests
  - RabbitMQ
- Sharing file system across services (docker containers)
  - Use mounted volumes to share file locations
- Tracking repeat simulation setups
  - Hash input combination and store for future reference

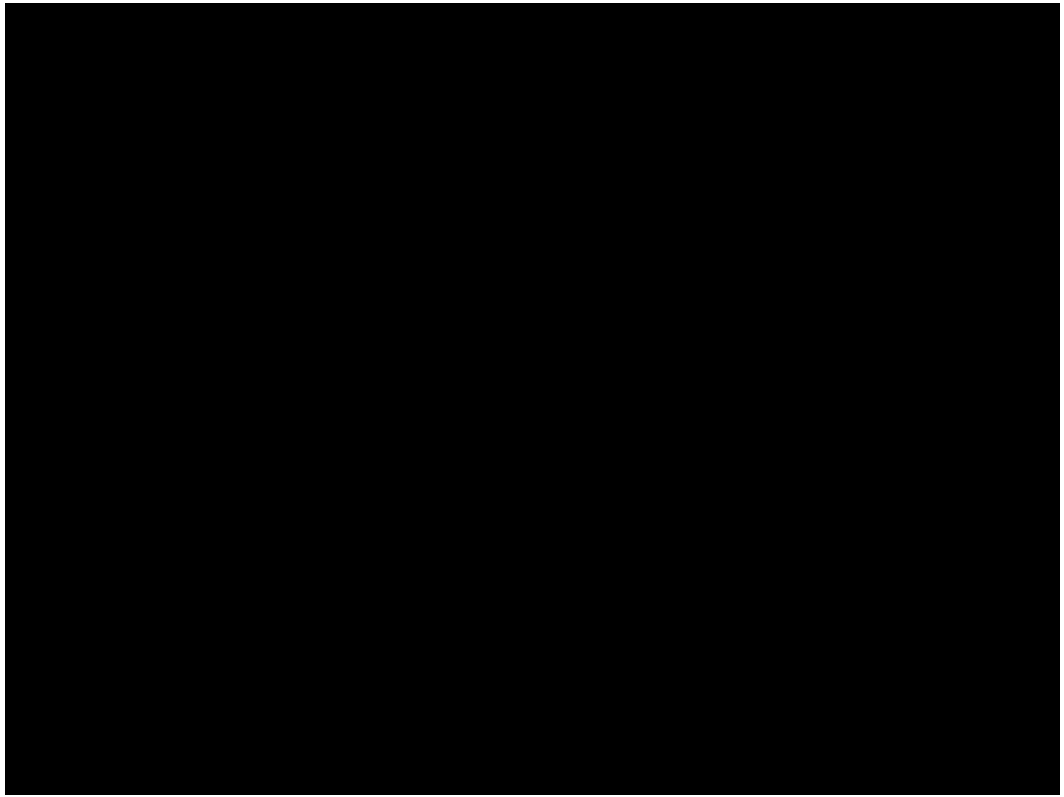
Testing

- Unit Testing
  - JUnit and Mockito
  - Account Creation
  - Run Requests
- Interface Testing
  - Manual Testing
  - SQL Injection Testing
- Python Backend Testing
  - Manual testing with known inputs
- Acceptance Testing
- CI/CD

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 20, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.566 s
[INFO] Finished at: 2023-04-29T20:11:11-05:00
[INFO] ------------------------------------------------------------------------
```

Demo

Short Demo Video



Future Work

Future Work

- Expand setup parameters
  - Partitions (overlap, sizing)
  - Drone start positions
- Enhance security standards
  - Plaintext login/registration
  - Run simulations in restricted containers
- Additional simulation statistics
- Pathing for CPH algorithm
- Additional/Modular algorithms support
  - Standard format (input/output, runner method)
  - Multiple languages
- Email notifications

Conclusion

Conclusion

Completed Work (Frontend):

- Upload user algorithms & simulation setup
- Output stats comparison
- Path visualization

Completed Work (Backend):

- APIs for user authentication, algorithm upload, run request processing
- Asynchronous queue processing
- Python simulation execution w/response outputs & pathing

Conclusion (Project Objectives)

- Ability to upload input and execute simulations
  - Based on specified parameters
- Algorithms run on backend server
  - Output data to frontend
- Visualize events and drone movements
- Extendable application

What does this mean?

- Display research algorithms in ways which are easy to understand
- Easy to modify for future research algorithms
- Potential for real-world use in drone fleet testing

Thank you!

Questions?