Constrained Re-Planning in Spatial Crowdsourcing

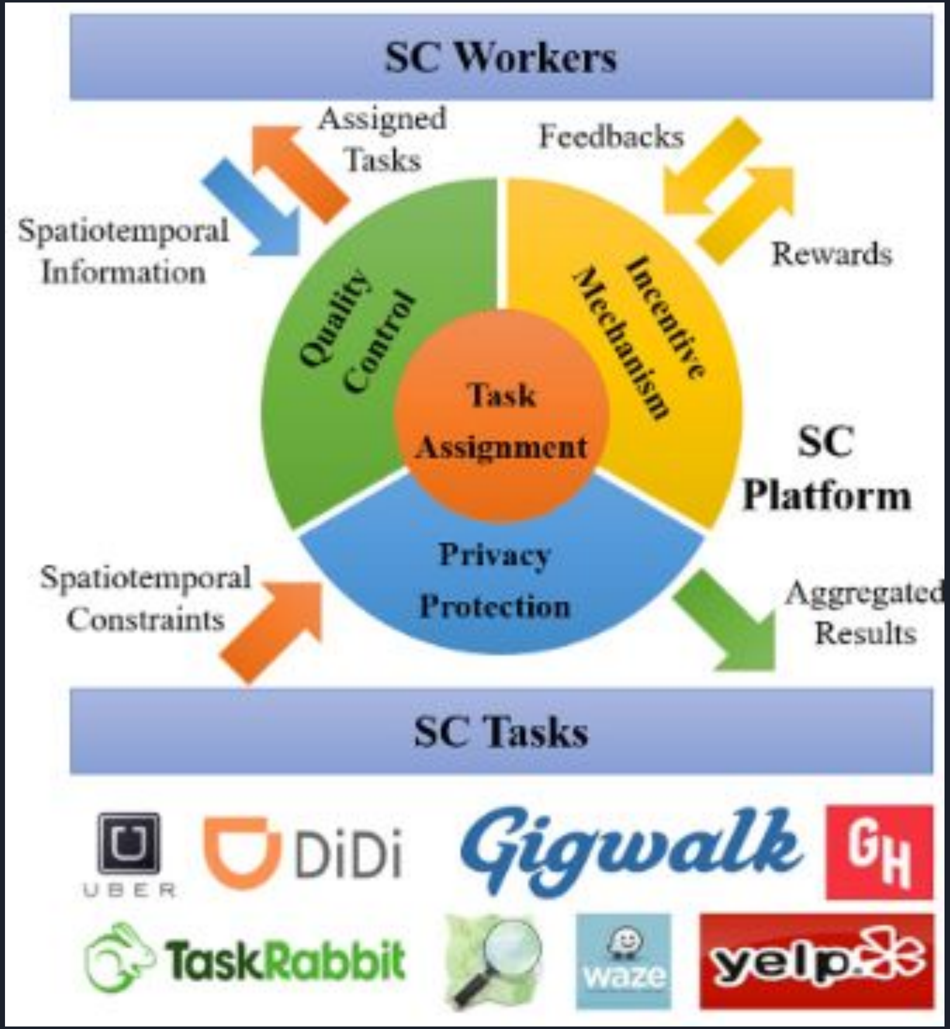
Team 51
Logan Anderson, Nicholas Heger, Steven Sheets,
Jared Weiland, James Volpe

Client and Advisor
Goce Trajcevski

Project Overview

Project Goal:

Create a system that implements a spatial crowdsourcing algorithm to run on a web application that can match workers with jobs from consumers/employers.

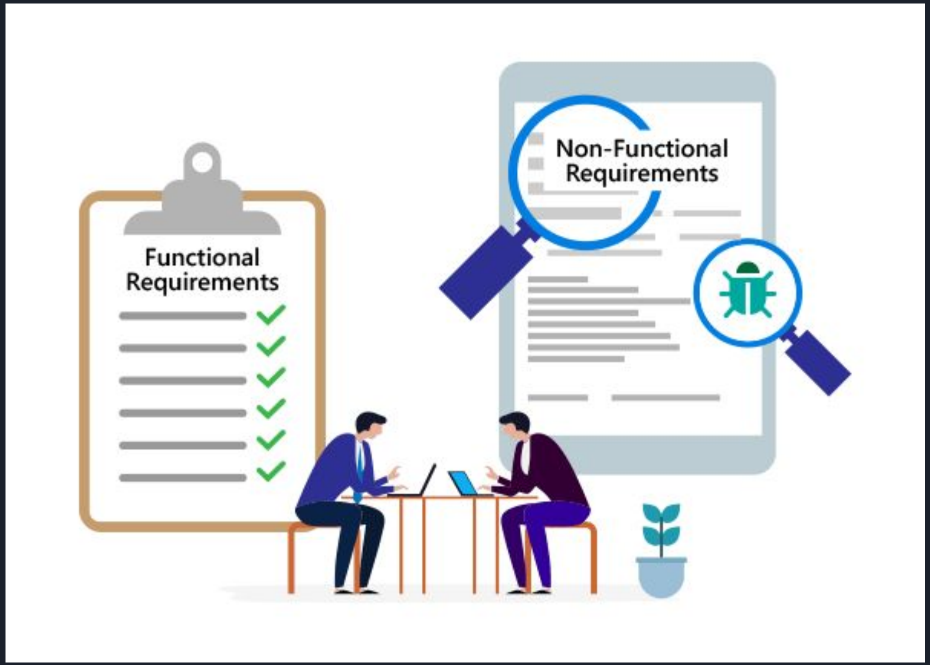


Functional Requirements

- Allow workers and employers to create accounts
- Take input of workers: skills and location
- Take input of tasks: location and skills required
- Optimize a schedule based on task and worker input
- Display tasks assigned to workers
- Re-optimize schedule in the event of new constraints
- User interface for visualization of work schedule (work routes)

Non-Functional Requirements

- Reliability - few bugs or issues that impede user experience
- Performance - algorithm is efficient and app is optimized for web/mobile
- Scalability - able to be used by a large number of users simultaneously
- Maintainability - readable code with documentation
- Usability - intuitive/easy to use



Engineering Constraints

- Must run as a mobile and desktop app
- Server needs to be able to handle algorithm processing
- Application requires internet connection
- Free Mapbox API
  - 50,000 monthly map loads
  - 100,000 monthly direction requests
  - 100,000 monthly geocoding requests
- Project must work without a budget
- Project must be planned and completed within two semesters

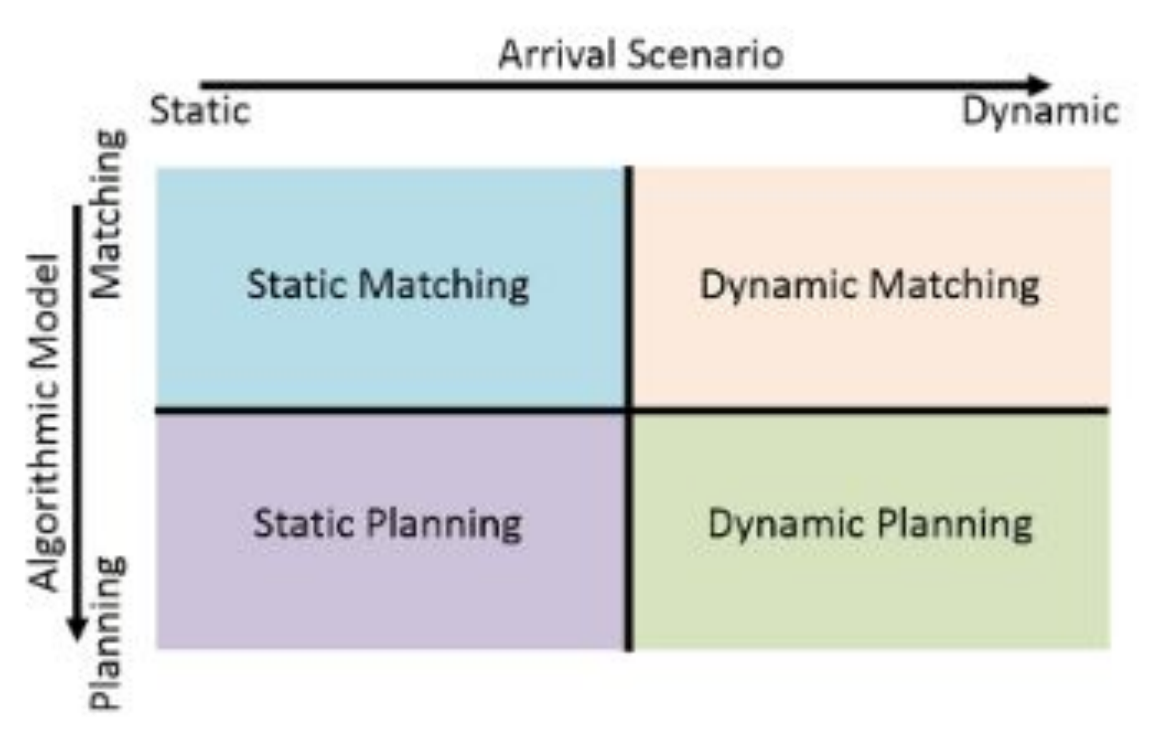
Algorithmic Models/Scenarios

4 types of spatial algorithms

- Static Matching
- Static Planning
- Dynamic Matching
- Dynamic Planning

Current Implementation plan

- Initial: Static Planning
- Final: Dynamic Planning

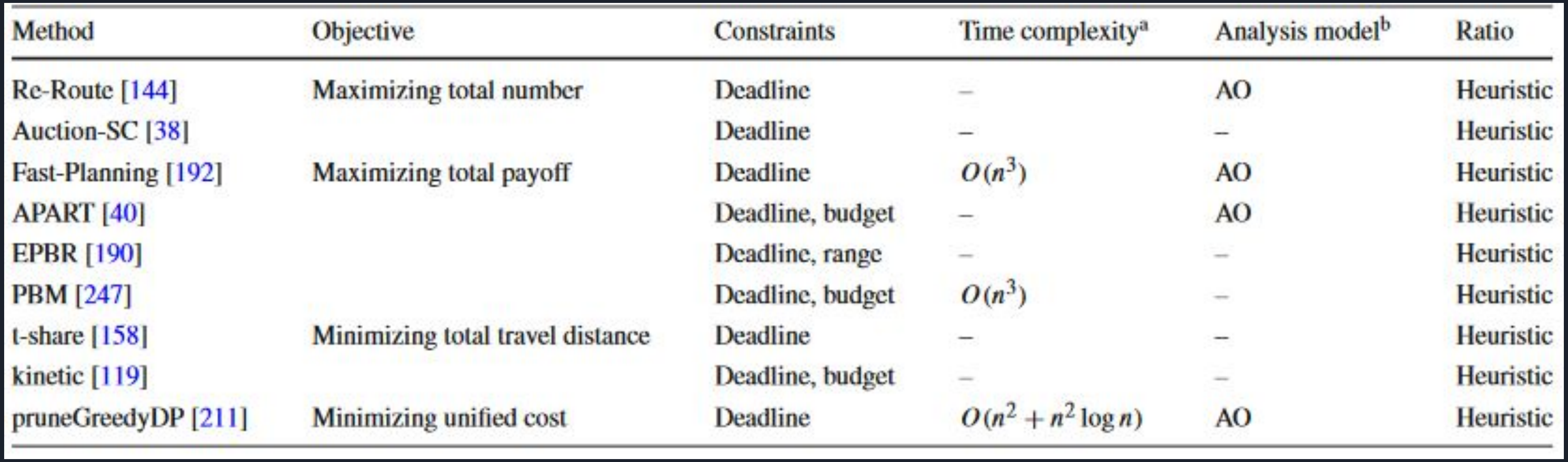


	Static	Dynamic
Matching	Static Matching	Dynamic Matching
Planning	Static Planning	Dynamic Planning

Algorithmic Approaches

Different algorithms have different…

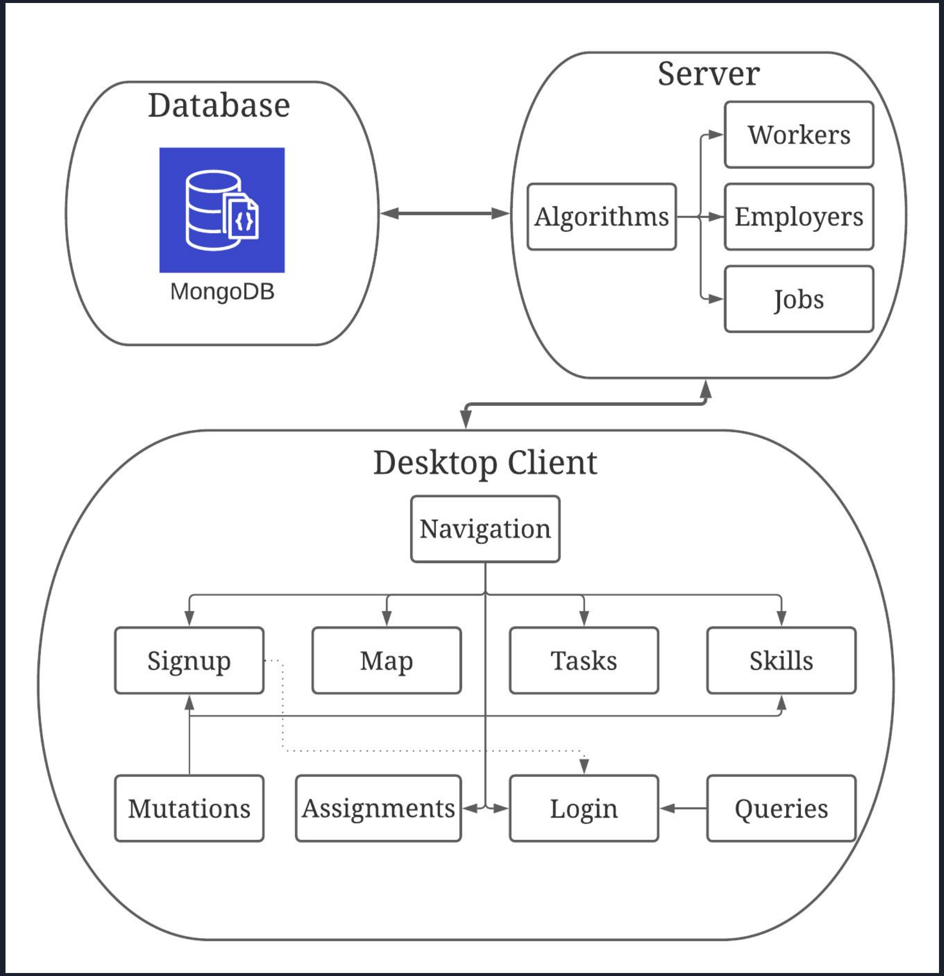
- Objectives
- Constraints
- Complexity



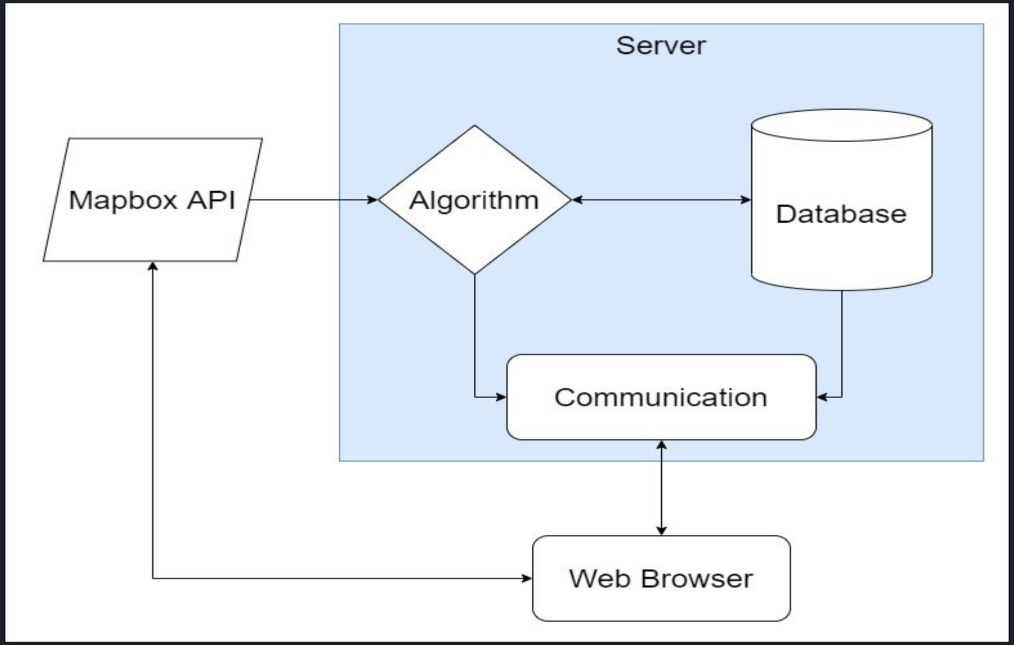
Method	Objective	Constraints	Time complexity[a]	Analysis model[b]	Ratio
Re-Route [144]	Maximizing total number	Deadline	–	AO	Heuristic
Auction-SC [38]		Deadline	–	–	Heuristic
Fast-Planning [192]	Maximizing total payoff	Deadline	$O(n^3)$	AO	Heuristic
APART [40]		Deadline, budget	–	AO	Heuristic
EPBR [190]		Deadline, range	–	–	Heuristic
PBM [247]		Deadline, budget	$O(n^3)$	–	Heuristic
t-share [158]	Minimizing total travel distance	Deadline	–	–	Heuristic
kinetic [119]		Deadline, budget	–	–	Heuristic
pruneGreedyDP [211]	Minimizing unified cost	Deadline	$O(n^2 + n^2 \log n)$	AO	Heuristic

System Design

Block Diagram



Concept Diagram



Technologies/Tools Utilized

Frontend

- ReactJS
- Map API - MapBox
- UI - Bootstrap

Backend

- Server - Spring Boot
- GraphQL - Queries
- Database - MongoDB

Technical Challenges

- Implementing spatial crowdsourcing algorithm
- Frontend-backend communication/familiarizing ourselves with Apollo GraphQL
- Familiarization with Mapbox API
- Working without in-person team interaction for much of the project
- Makes sure there is not memory bloat in the client

Design Changes

- Switched database
    - MySQL to MongoDB
    - + Simple, intuitive setup
    - + More scalable (performs better than MySQL with large data)
    - + Better integration with GraphQl
    - - Less advanced privacy and security
- Switched Map API
    - Google Maps API to MapBox
    - + Volume based vs feature based
    - - Less public adoption and support

Evaluation Criteria

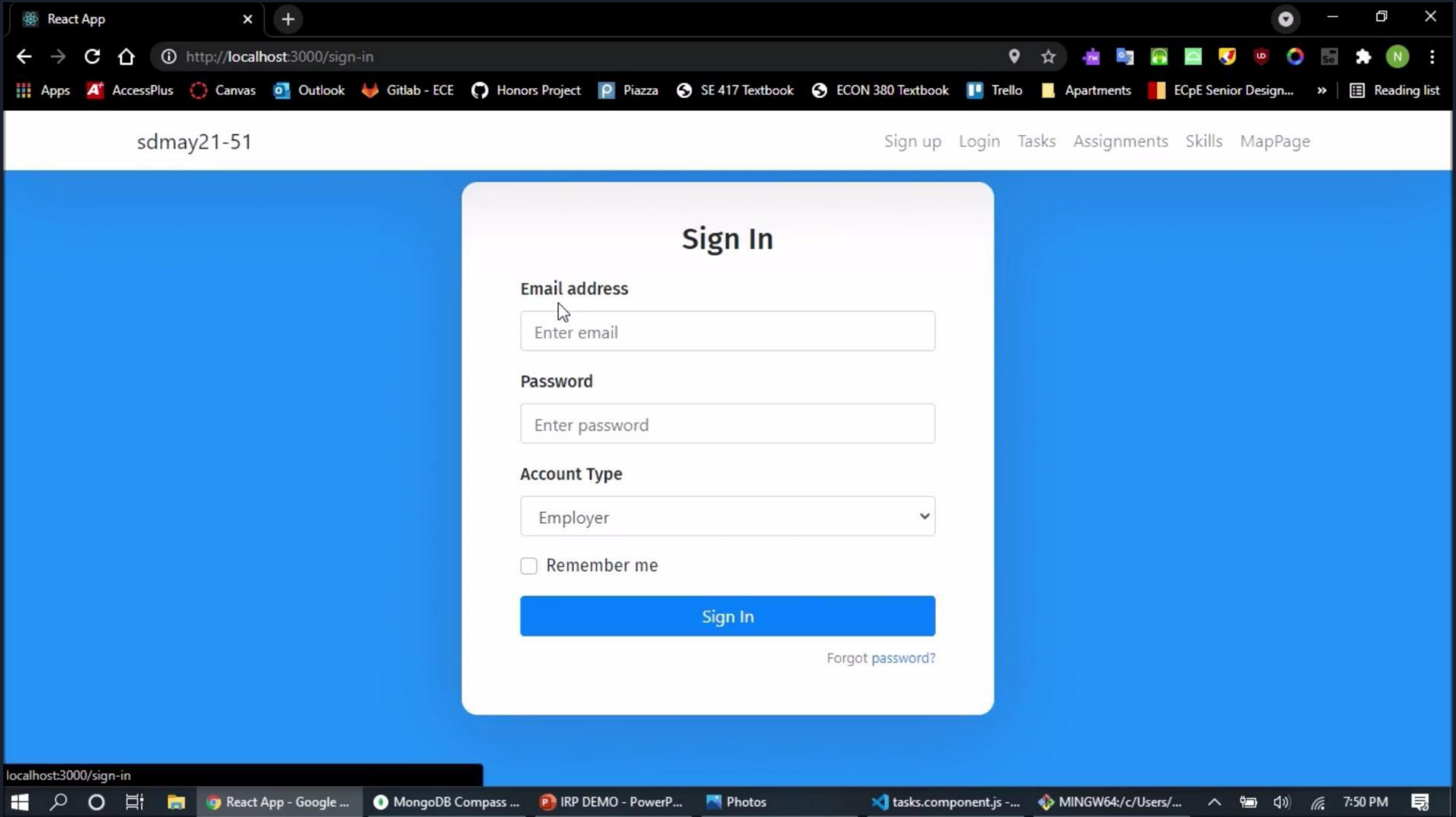
- Usability

- Speed

- Bugs

- Algorithmic Efficiency

Demo



Sign In

Email address

Enter email

Password

Enter password

Account Type

Employer

□ Remember me

Sign In

Forgot password?

Testing Overview

Unit Testing

- Jest testing Framework
- Optimizing Algorithm
    - Correct assignments
    - Optimized runtime

Interface Testing

- Verify data from database are passed to algorithm
- Verify algorithm results are passed to frontend and task assignments are displayed to workers

Acceptance

- Beta Testing
- Different algorithms being used
- Varying sizes of datasets
- Multiple workers processing
- Varying skill-sets

Engineering Standards

- IEEE/ISO/IEC 29119-2-2013 - ISO/IEC/IEEE International Standard - Software and systems engineering —Software testing —Part 2:Test processes

- IEEE/ISO/IEC 29119-3-2013 - ISO/IEC/IEEE International Standard - Software and systems engineering — Software testing —Part 3: Test documentation

- 29119-4-2015 - ISO/IEC/IEEE International Standard - Software and system Engineering -- Software testing --Part 4: Test techniques

Conclusion

- Created a functioning proof-of-concept application
- Learned a lot about design process
- Gained experience in the frontend and backend technologies used
- Application is modular and easy to expand upon

Future Expansion

- Allow workers to decline a task assigned to them
- Employer feedback decides worker skill level
- Allow algorithm to re-run on change of conditions
- Factor in security concerns

Questions/Comments?