Constrained Re-Planning in Spatial Crowdsourcing

FINAL REPORT

Team 51

Client/Advisor: Goce Trajcevski

Team Member	Roles
Logan Anderson	Frontend Engineer, Test Engineer
Nicholas Heger	Frontend Engineer, Progress Manager
Steven Sheets	Backend Engineer, Test Engineer, Report Manager
James Volpe	Frontend Engineer
Jared Weiland	Backend Engineer

Team Email: sdmay21-51@iastate.edu

Team Website: https://sdmay21-51.sd.ece.iastate.edu

Revised: 04/25/2021 – Final Version

Executive Summary

Development Standards & Practices Used

List all standard circuit, hardware, software practices used in this project. List all the Engineering standards that apply to this project that were considered.

- Agile
- Black-box testing
- Object-oriented programming
- Subscriber-publisher model
- Test driven development
- 29119-4-2015 - ISO/IEC/IEEE International Standard - Software and systems engineering--Software testing--Part 4: Test techniques
- 24748-3-2020 - ISO/IEC/IEEE International Standard - Systems and software engineering--Life cycle management--Part 3: Guidelines for the application of ISO/IEC/IEEE 12207 (software life cycle processes)
- 41062-2019 - ISO/IEC/IEEE International Standard - Software engineering - Recommended practice for software acquisition

Summary of Requirements

List all requirements as bullet points in brief.

- Create/research algorithm for task management
- Create a server to host algorithm, i.e., to run the backend on
- Create a database to store data for users and workers
- Develop a web and mobile application to allow utilization of optimized algorithm

Applicable Courses from Iowa State University Curriculum

- COM S 227: Object-oriented Programming
- COM S 228: Introduction to Data Structures
- COM S 309: Software Development Practices
- COM S 311: Introduction to the Design and Analysis of Algorithms
- COM S 363: Introduction to Database Management Systems
- CPR E 310: Theoretical Foundations of Computer Engineering
- S E 309: Software Development Practices
- S E 319: Construction of User Interfaces
- S E 329: Software Project Management
- S E 339: Software Architecture and Design

New Skills/Knowledge acquired that was not taught in courses

List all new skills/knowledge that your team acquired which was not part of your Iowa State curriculum in order to complete this project.

- React
- Mapbox API

- Task sorting/assignment algorithm
- MongoDB
- GraphQL

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

We would like to acknowledge our faculty advisor, Goce Trajcevski, for his advice and guidance throughout this project. Dr. Trajcevski has helped further our understanding of the project's goals and has helped keep us on track and meeting deadlines. We would also like to thank our TA, Rachel Shannon, for being consistently available to answer questions.

1.2 Problem and Project Statement

Spatial crowdsourcing (SC) is an increasingly popular category of crowdsourcing in the era of mobile Internet and sharing economy, where tasks are spatiotemporal (belonging to both space and time or space-time.) and must be completed at a specific location and time. It is a matching problem whereby one has: (1) a set of workers with their skills and geolocations; (2) a set of job-sites with tasks requiring specific skills (and, sometimes, there is a constraint on the sequence of tasks). Spatial crowdsourcing determines workers' assignment to job-sites for a given task, considering travel time. However, frequently there are unexpected time-disturbances – e.g., traffic accidents, prolonged execution of previous tasks, etc., which render an existing assignment no longer optimal (in terms of completed tasks per day).

This project aims to develop algorithms and tools that will re-plan the assignments of workers to new/different jobsites when variables change unexpectedly. This is so that one can still optimize the overall number of completed tasks per day while obeying certain constraints (e.g., minimizing the overtime pay of the re-assigned workers).

1.3 Operational Environment

The operational environment for this project will be web browsers and mobile devices. Since our end products are a web app and mobile app, there will be no physical constraints our project adheres to.

1.4 Requirements

- Functional Requirements
  - Allow employers and workers to be able to create accounts (stored in DB)
  - Take worker inputs of skills and location
  - Take task inputs of skills required and location
  - Optimize a schedule based on worker and task inputs
  - Re-optimize this schedule in the event of new information
  - Alert workers of tasks to complete
  - Web UI for the addition of tasks and visualization of work schedule
- Non-functional Requirements
  - Function with few bugs or issues that impede the users' experience
  - Protect users' personal information from others
  - Optimized applications to run efficiently on mobile devices
  - Be able to be used by a large number of users at one time

1.5 Intended Users and Uses

This project's primary focus is to create an efficient algorithm to solve spatial crowdsourcing problems where tasks need to be assigned to workers. As such, our end products (the web and mobile applications) will be very versatile and could be used by any spatial crowdsourcing service such as Uber or Grubhub. The intended users would then be any current or future users of any app that seeks to use spatial crowdsourcing to accomplish tasks.

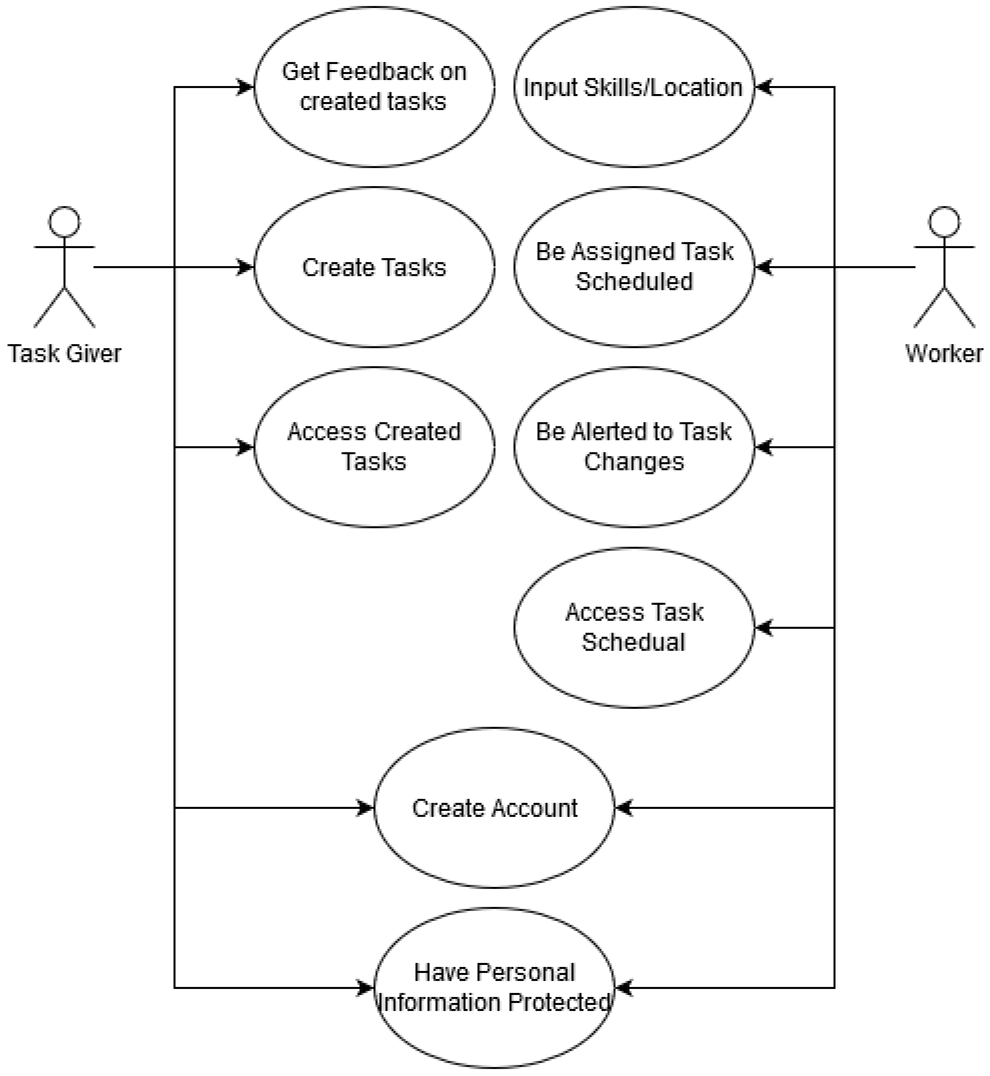

Task Giver
Worker
Get Feedback on created tasks
Input Skills/Location
Create Tasks
Be Assigned Task Scheduled
Access Created Tasks
Be Alerted to Task Changes
Access Task Schedual
Create Account
Have Personal Information Protected

Figure 1: Use-Case Diagram

1.6 Assumptions and Limitations

- Assumptions
  - Privacy is handled through outside sources. like location ghosting for hiding user location
  - There is only one task per assignment
  - Tasks are assigned in sequence
- Limitations
  - Traffic APIs have a processing cap on the number of routes that can be run per period
  - Will need to be able to run on multiple types of mobile devices
  - Will need a connection to the Internet to receive updated information

1.7 Expected End Product and Deliverables

The main deliverables from this project are a desktop and mobile app that will take a set of tasks/workers assignment and the data used for such assignments. Upon notification that some values in the data used for the original assignments have changes (e.g., the average speed or travel-time along a road segment), the app will: (A) calculate the optimal re-assignment; (B) notify the affected workers (and job-sites) who are subject to such re-assignment.

2 Project Plan

2.1 Task Decomposition

Solving the problem at hand helps to decompose it into multiple tasks and subtasks and understand interdependence among tasks.

For our project, the tasks can be decomposed quite simply. Users known as "task generators" will be stored in a database and generate a set of tasks, each task consisting of attributes such as geolocation, necessary skills, and a time requirement for the sequence of jobs. Users known as "workers" will also be stored in our database, each containing attributes such as geolocation, skill-sets, ranking amongst other workers, and pricing (per hour). Finally, our objectives mainly focus on assigning workers to tasks, assuming a single-task assignment, and a sequential assignment of tasks.

The necessary tasks we must complete to complete are detailed in Table 1 on the following page:

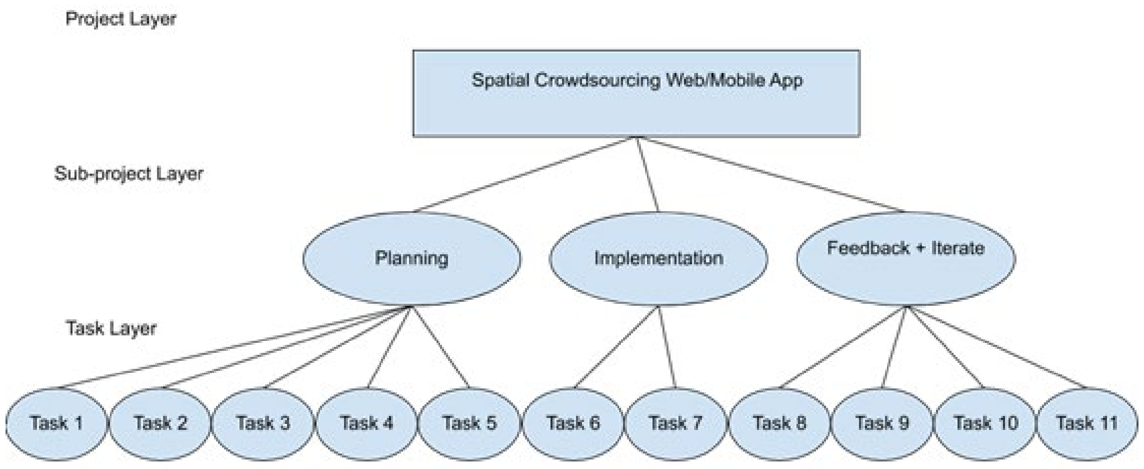


Figure 2: Task Decomposition Diagram

Table 1: Table of Tasks

Task #	Planned Completion Date	Task Description
1	Sep. 10	Complete familiarization with the literature and existing approaches, decide running scenario/use-case.
2	Oct. 10	Finalize the selection of datasets to be used as sources.
3	Oct. 25	Finalize the selection of development platforms and provide architecture design with preliminary UI format.
4	Nov. 10	Finalize the selection of algorithmic solutions; devise use-cases and test-cases; develop test-plans (unit testing; integration testing; etc.); provide basic UI functionality.
5	Nov. 20	Finalize and submit the design document; prepare presentation.
6	Jan. 25	Finalize the role/component assignments and start implementing collaborative modules.
7	Feb. 15	Complete unit testing; begin integration testing.
8	Mar. 5	Provide alpha-version for end-user testing; collect feedback.
9	Mar. 20	Finalize the revisions; release beta-version; run another set of end-user testing of functionalities.
10	Apr. 5	Finalize the user-manual; prepare for public release.
11	Apr. 15	Deploy the final version at GitHub/GitLab; start the final report and presentation preparation.

2.2 Risks And Risk Management/Mitigation

- Task 1 & 2) methodology may not work with our project: 10%
    - We find this unlikely as at this point; we should have enough information to make an educated decision about which to use.
- Task 3) development studio does not work as intended: 50%
    - If a studio does not work as intended and no significant work has been done, it would be in the project's interest to switch to a different studio. If there is a fair amount of work done, then it may be better to stick with it even if it does not work as effectively as it could.
- Task 4) Test cases do not cover all necessary paths: 70%
    - Add more test cases to cover missing paths
    - Testing does not work with studio: 40%
- Task 5) N/A
- Task 6) Team member does not like doing assigned tasks: 40%
    - Team member falls behind on component: 70%
    - Would need to find out why they are falling behind and adjust the schedule as necessary.
- Task 7) Users do not like elements of the UI: 80%
    - Rework UI components
    - Users do not like functionality: 60%
    - Try and make changes, but will not wholly rework
- Task 8) Identical to Task 7
- Task 9) Major issue is found before release: 10% or less
    - Try and hotfix the issues for release before making more lasting repairs. Disable troublesome features if needed.
- Task 10) N/A

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

The proposed milestones of our project have, in essence, a 1-1 correspondence with the tasks described in Section 2.1. Many metrics/evaluation criteria can be used to evaluate our project. Some are as follows:

- **Usability**: Is our code easily understood? Does the UI provide simple usage? Is our documentation comprehensible?
- **Speed**: Is our software slow? Can it be faster? How could it be optimized? Load. Can our software/database deal with large numbers of users? If not, how could the database be improved?
- **Bugs**: Does our software have any bugs? How can they be squashed? Are they negatively impacting user experience?
- **Algorithmic Efficiency**: Does the algorithm make efficient schedules? How efficient should it be? Where do we draw the line between efficiency and practicality?

2.4 Project Timeline/Schedule

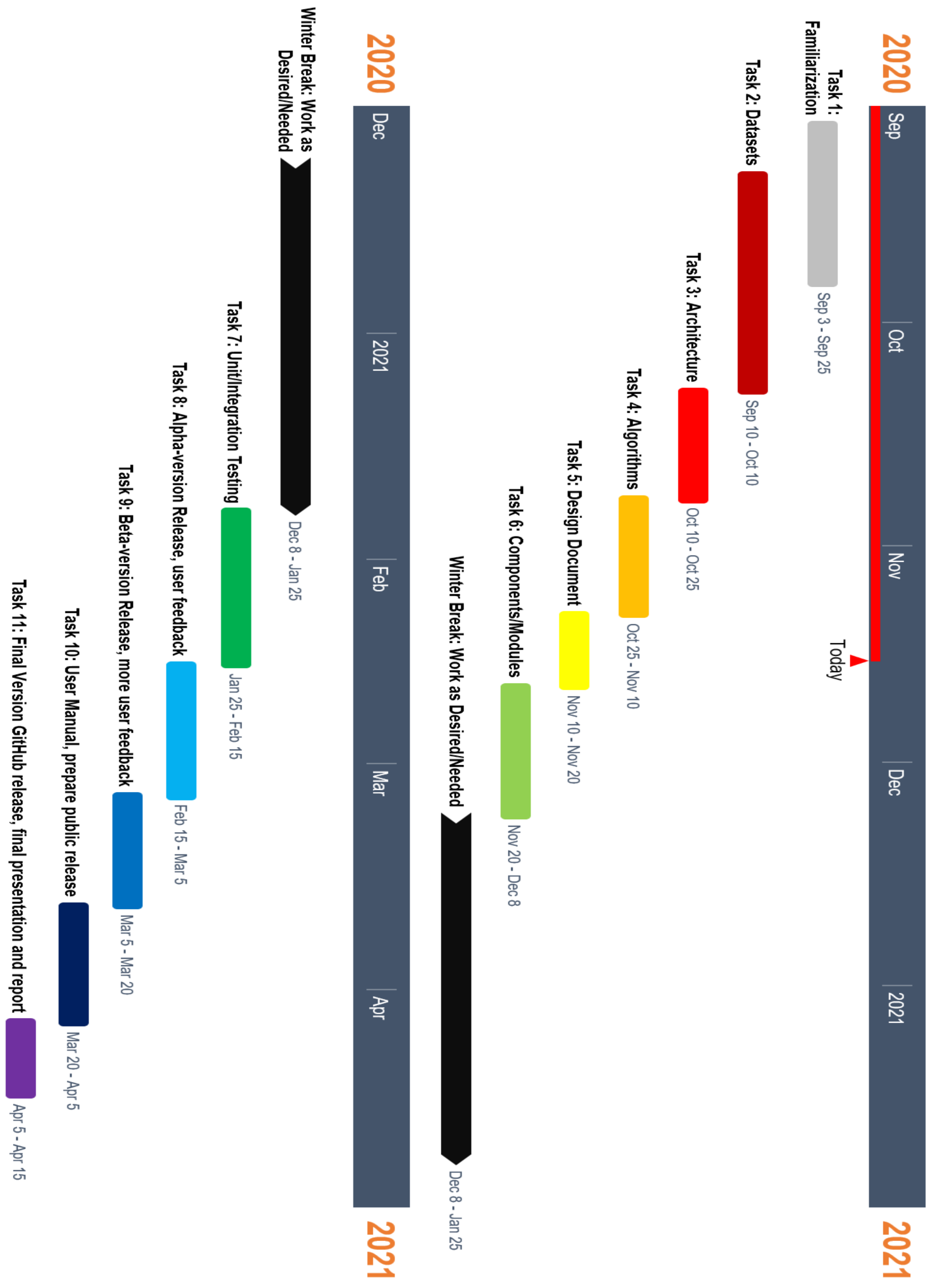


Figure 3: Gantt Chart

2.5 Project Tracking Procedures

Trello will be used to set tasks and track progress. Major tasks are assigned with due dates. Tasks that need to be done are entered into a TODO column. When a team member starts working on a task, it is moved to the "Doing" column and the team member's name is assigned to it. When the task is done, it is moved to the "Done" column and is archived. GitLab will be used for version control of the project. Documents related to the project are kept on Google Drive to keep a single version of the project documentation. General communication is being done through discord for communication history.

2.6 Personnel Effort Requirements

For each of these tasks, we set aside several days less than the time in-between tasks. This is an estimation of the number of days it will take to complete if we spend merely half an hour each day. Keep in mind; this is with the combined effort of 5 workers.

Table 3: Table of Estimated Time

Task #	Estimated Completion Time (in hours)
1	$7\ days, 5\ workers, 0.5\ hours/day$ $7 \cdot 5 \cdot 0.5 = 15.75\ hours$
2	$12\ days, 5\ workers, 0.5\ hours/day$ $12 \cdot 5 \cdot 0.5 = 27\ hours$
3	$8\ days, 5\ workers, 0.5\ hours/day$ $8 \cdot 5 \cdot 0.5 = 18\ hours$
4	$9\ days, 5\ workers, 0.5\ hours/day$ $9 \cdot 5 \cdot 0.5 = 20.25\ hours$
5	$6\ days, 5\ workers, 0.5\ hours/day$ $6 \cdot 5 \cdot 0.5 = 15.75\ hours$
6	$14\ days, 5\ workers, 0.5\ hours/day$ $14 \cdot 5 \cdot 0.5 = 31.5\ hours$
7	$16\ days, 5\ workers, 0.5\ hours/day$ $16 \cdot 5 \cdot 0.5 = 36\ hours$
8	$13\ days, 5\ workers, 0.5\ hours/day$ $13 \cdot 5 \cdot 0.5 = 29.25\ hours$
9	$10\ days, 5\ workers, 0.5\ hours/day$ $10 \cdot 5 \cdot 0.5 = 22.5\ hours$
10	$12\ days, 5\ workers, 0.5\ hours/day$ $12 \cdot 5 \cdot 0.5 = 27\ hours$
11	$8\ days, 5\ workers, 0.5\ hours/day$ $8 \cdot 5 \cdot 0.5 = 18\ hours$

2.7 Other Resource Requirements

Physical devices will be required for testing of the web and mobile app. Web testing may be done through any device with access to the Internet, and mobile testing may be done through a mobile device or an emulator on a laptop or desktop computer. As most people have access to such devices, it is unnecessary to acquire devices specifically for testing. A server is also required. If the school provides the server, then no additional resources will be required.

2.8 Financial Requirements

The only potential financial cost of our project is the Mapbox API. We must be conscious of our Mapbox usage to keep ourselves below the 50,000 monthly request limit. After this limit our Mapbox usage is no longer free, however this limit is substantial for the purposes of our project, so there should be no financial expenses.

3 Design

3.1 Previous Work And Literature

Our problem is one that has been studied for nearly a decade and is continuously being researched. This is not unexpected, as spatial crowdsourcing has a natural, crucial connection with the physical world, and examples of its use are easily demonstrated through services such as GrubHub or Uber.

Our primary reference is a survey Tong, Y., Zhou, Z., Zeng, Y. [1], which focuses on the spatiotemporal factors of spatial crowdsourcing. There are many general surveys [2, 3, 4, 5] or tutorials [6, 7, 8] on traditional Web-based crowdsourcing. There are also some surveys or tutorials that focus on spatial crowdsourcing. E.g., Guo et al. [9] and Tong et al. [10] review task allocation of spatial crowdsourcing. The following table illustrates a timeline of milestone papers concerning spatial crowdsourcing.

Table 3: Timeline of Milestone Papers on Spatial Crowdsourcing

Year	Reference	Influence
2012	[11]	First work of spatial crowdsourcing.
2013	[12]	First work of static task matching in SC.
2013	[13]	First work of quality control in SC.
2014	[14]	First work of privacy protection in SC.
2014	[15]	First work of general SC platform.
2015	[16]	First work of dynamic task planning in SC.
2016	[17]	First work of dynamic task matching in SC.
2016	[18]	First experimental work of dynamic task matching in SC.
2018	[19]	First work of incentive mechanism in SC.
2018	[20]	First work of privacy protection in dynamic scenario.

3.2 Design Thinking

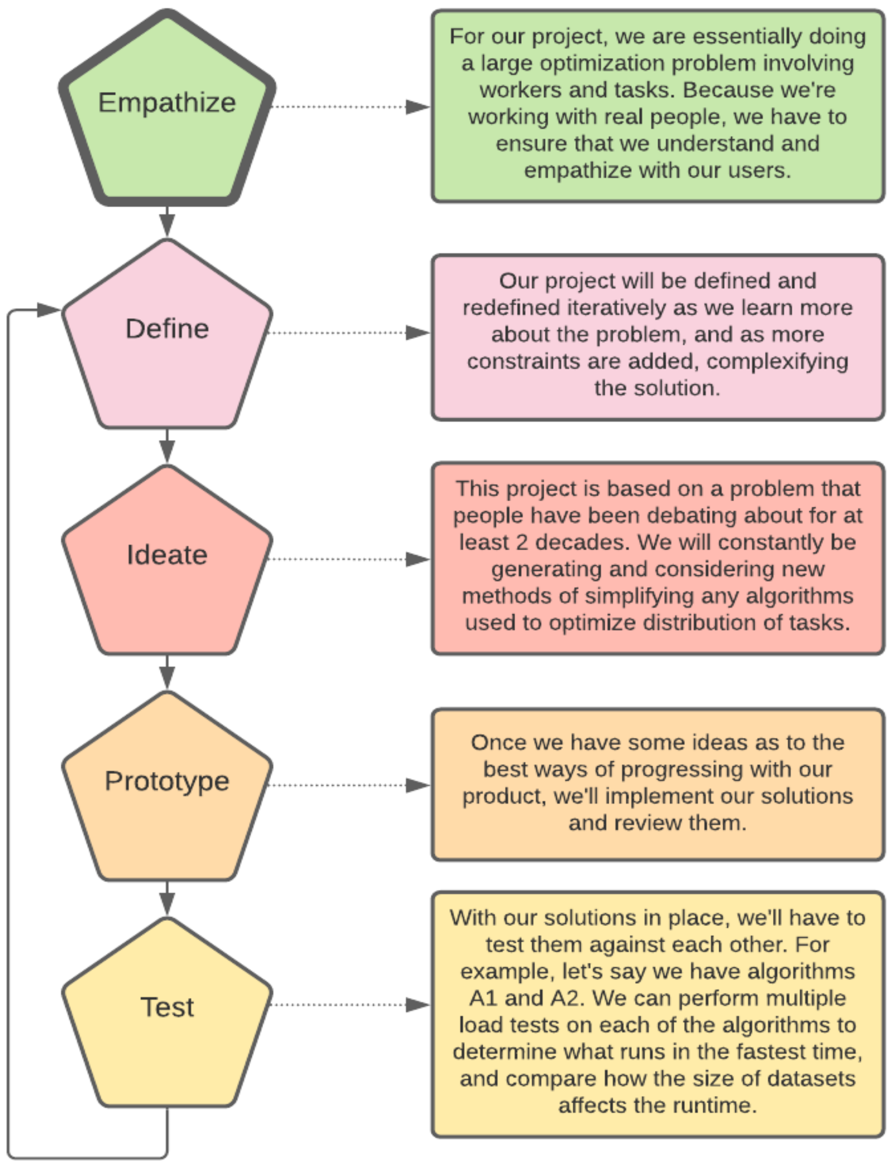

Empathize
For our project, we are essentially doing a large optimization problem involving workers and tasks. Because we're working with real people, we have to ensure that we understand and empathize with our users.
Define
Our project will be defined and redefined iteratively as we learn more about the problem, and as more constraints are added, complexifying the solution.
Ideate
This project is based on a problem that people have been debating about for at least 2 decades. We will constantly be generating and considering new methods of simplifying any algorithms used to optimize distribution of tasks.
Prototype
Once we have some ideas as to the best ways of progressing with our product, we'll implement our solutions and review them.
Test
With our solutions in place, we'll have to test them against each other. For example, let's say we have algorithms A1 and A2. We can perform multiple load tests on each of the algorithms to determine what runs in the fastest time, and compare how the size of datasets affects the runtime.

Figure 4: Design Thinking Diagram

Before we even started planning the project, our team planned how we would go about creating a sound and thorough plan. To do this, we decide to use a design thinking diagram (see Figure 4). The first step in the design thinking process is to empathize with our clients. To reach a better understanding with our client, we have had many meetings with our client and faculty advisor, Dr.

Trajcevski, and listened to his ideas as well as presented our own ideas. During the define phase of our project, the objective was to narrow down the problem we hope to solve. This involved choosing which factors our spatial crowdsourcing algorithm needed to take into consideration and determining what was outside the scope of our goals. In section 1.6 we discuss some of the results of the "define" process, for example making the assumption that privacy is handled by an outside source and will not need to be implemented in our project. We also defined a set of tasks to be completed for our project. This task set and decomposition diagram of these tasks can be found in section 2.1.

During the "ideate" phase we discussed many different approaches to an algorithm that solves this problem. More information regarding this decision process can be found in section 3.7. Figure 5, shown in Section 3.7, shows the system's overall architecture that should be the deliverable, subject to modifications. We have many iterations to the problem, such as the contexts of the use-cases. Another complexity is considering the tradeoffs between different tools, technologies, and frameworks. The following sections explore this in more detail.

In addition to creating an algorithm, our objective is to build a system. We had to consider the different available technologies. The tradeoffs will be addressed in more detail in Section 3.5.

3.3 Design Evolution

We will now break down the diagram from Section 3.2 into its components, explain its technology, and discuss some of the changes that were made to our original design.

The front end consists of a web and mobile application. The algorithm is part of the backend architecture, which is being supported using Spring Boot. The algorithm will take tasks, workers, and locations stored in the MongoDB database along with route information given by the Mapbox API to generate routes for the tasks. It will then take these routes to create an optimal route for tasks. Because of this, the algorithm can recalculate these changes if new tasks are added that affect the already created task schedules. These task schedules will then be assigned to the worker they were generated for. This would then be stored in the database. The worker will then be able to use the frontend, which is a web application created using React, to access the list of tasks assigned to them, view the routes for those tasks, and view and add skills to their profile. Data will be communicated between the frontend and backend using GraphQL. The tasks listed in section 2.1 will allow the completion of the function requirement listed in 1.4.

In place of MongoDB above, we had originally planned to use SQL. We decided against SQL due to how simple it was to setup Mongo, as well as how documents are stored in Mongo vs. SQL. It enabled easy read/write access, whilst SQL can be quite complex.

We also originally planned to use Google Maps API instead of the Mapbox API mentioned above. This switch was because of funding. While we would have preferred to use Google as it has much more support and documentation it has a set of basic features that are available for free and cost to unlock other features. Mapbox allows for the use of all of its features and charges by volume where if you are under it is free. It also does not require linking a payment system and will not auto charge you and will instead stop services.

Additionally, we originally did not plan to use GraphQL for our project. We ended up using it because of it has an incredibly nice GUI, accessible when running at `localhost:8080/graphiql`, that allows you to easily test every single query or mutation you create. If you can get it working on that link, you can get it working on the frontend.

3.4 Technology Considerations

We will discuss the different technologies that we researched and the reasons for choosing the technology that we did.

There are many options for Traffic APIs. We looked at Google Maps, Bing Maps, Mapbox, Foursquare, PositionStack, and MapQuest Developer. We decided upon Mapbox because
 a) It is popular, meaning that the interface is familiar to users
 b) It has clear developer documentation
 c) It is free if its usage is under 50,000 requests per month

For creating the user interface, we primarily researched React and Angular. We decided on React because of its increasing popularity as a JavaScript framework and its ability to be ported over from a web app to a mobile app. Additionally, it can easily be used to create a web app that runs in mobile device browsers, which is what we settled on.

For the backend, we look at using Spring Boot and Django. We decided on Spring Boot because it is designed for Java and would therefore integrate well with our Java/JavaScript based project.

The technology for the database on the backend was between SQL and MongoDB. We decided on MongoDB for the reasons discussed in section 3.3.

To transfer data between the frontend and backend, we are using GraphQL. It has a simple interface for communication between the DB, frontend, and backend, and is all testing with its interactive user interface. The entities are stored in collections in the DB, and they are fundamentally JSON values the frontend can access. It also works very well with Spring Boot – well integrated.

For many of these technologies, we wanted to keep the project in Java or Java derivatives (like JavaScript) to make implementation more manageable and reduce the likelihood of incompatibility between different components. It would also allow developers to switch between components as needed without needing to learn something completely different.

The algorithm we decided to use is a dynamic scheduling algorithm. This is because it allows for workers' schedules to be updated as new tasks are added and for workers to plan more than a single task.

3.5 Design Analysis

We considered all the information to date and have made an analysis of the technologies involved, as seen in section 3.4. This has convinced us that our design is the best way to implement this problem. One reason for this, as mentioned in Section 3.4, is that the decided technologies are based or are compatible with Java/JavaScript. This means a low likelihood of incompatibility between components makes it easier for people to work on different components. Our design is simple and straightforward; the fact that the algorithm is a separate component makes it easier to change or swap out with other options without creating large problems in the system as a whole. This contributes to a microservice architecture with loose coupling between the algorithm and the rest of the system.

As the project evolved, we made decisions to modify several design choices (outlined in Section 3.3). These modifications were all in response to our original choices becoming inadequate or suboptimal after new information was made available to us. Because of this, each of these changes resulted in an improvement over our initial design and ultimately a better end product.

3.6 Development Process

As discussed earlier, we implemented an Agile development methodology to complete our project. Notably, we followed a Kanban model instead of the slightly more popular Scrum model. This decision was made because, as a small team of just 5 members, and this being an academic setting, the Kanban model allows for a less rigid development cycle and more flexibility to account for some weeks being more or less busy in our schedules.

Additionally, we used Trello to create tickets for each task that needs to be completed. Upon starting a task, we moved the ticket into the "In Progress" column on Trello and created a branch in our GitLab repository, if applicable. Once work is completed on the task, a pull request was created in GitLab, which allowed the team to review and request any changes to the implementation of the ticket. Once approved, the branch can be merged into master (which serves as the "on-production" branch), and the pull request closed. This ensures everyone on our team approves of the changes, and we have a history we can go back to in case anything breaks.

3.7 Design Plan

As mentioned earlier in this document, there are many different facets of this problem – several categories and variants. For each of these, we developed proper use-cases. The back-and-forth of this process was in terms of identification of the commonalities and incongruencies of such variants.

Let us investigate the table on the following page from the survey by Tong, Y., Zhou, Z., Zeng, Y. [1] and pull some examples from there. With the "Greedy-GEACC" method, the objective is the total payoff, and the only constraint tied with it is Capacity. This results in a specific time complexity and requires its own unique tests, elaborated on in Section 4, Testing. Compare this method with a completely different one, such as IDA, which has the primary objective of minimizing the total distance travelled. These two methods, while having the overall same end goal, have two very different objectives for optimization. Comparing these two methods in tests gives us even more insight – e.g., IDA's tests need to ensure that the distance travelled is always the minimum across multiple methods. Greedy-GEACC should ensure maximum profit is being made.

To fully elaborate on every unique facet of this problem in this document is, as they say, a tad bit overkill. Fundamentally, we want our database/server to be universal, while the UI and algorithm is flexible based on constraints. For a more thorough dive, please view the references for respective topics.

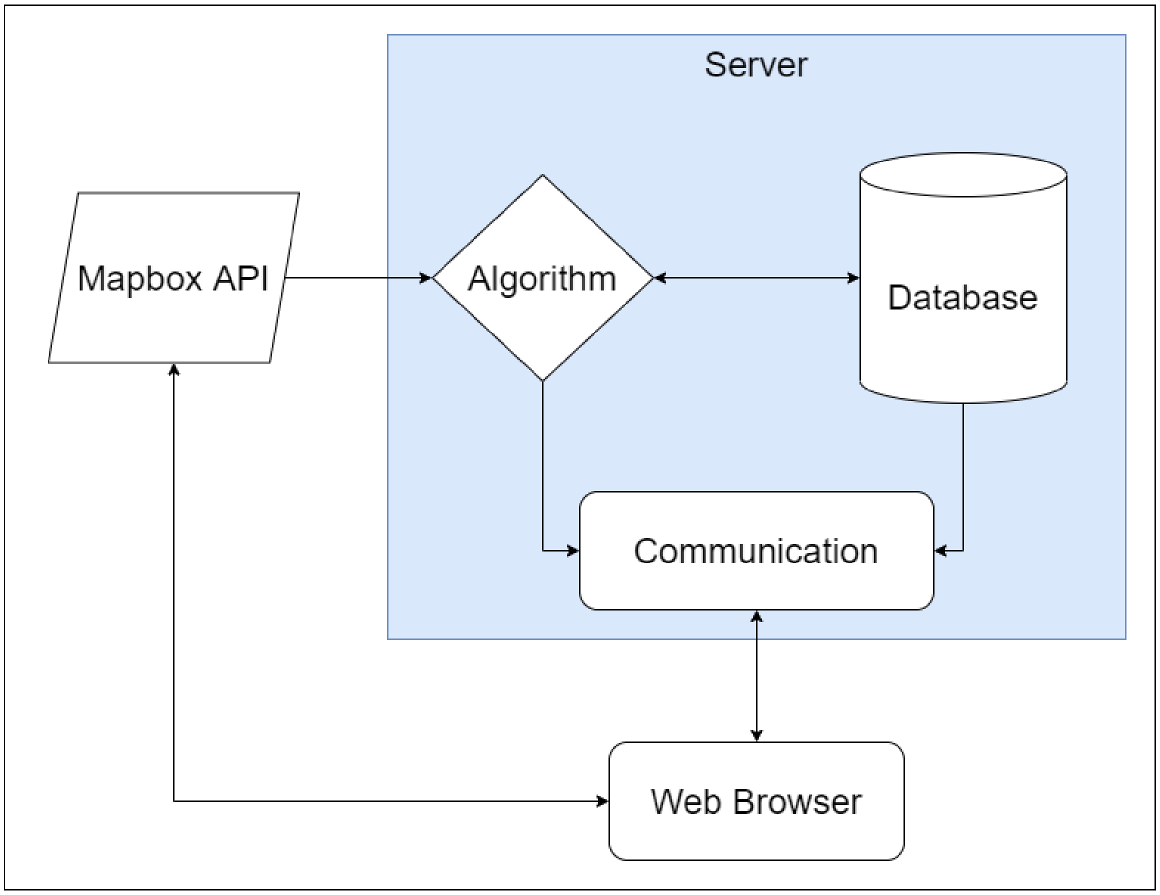


Figure 5: System Diagram

Table 4: Comparison of Existing Solutions to Task Assignment as a Static Matching Problem

Method	Objective	Constraints[a]	Time Complexity[b]	Ratio
GR [11]	Maximizing total number	Deadline, range	-	Optimal
SP-WR-A [21]		Range	-	Heuristic
Temporal [22]		Deadline, range, budget	-	Heuristic
Greedy-GEACC [23]	Maximizing total payoff	Capacity	$O(n^3)$	$\frac{1}{1+C_{max}}$
g-D&C [24]		Deadline, range, skill, budget	-	Heuristic
ADAPTIVE [24]		Deadline, range, skill, budget	-	Heuristic
BASIC [25]		Deadline, range	-	Optimal

IDA [26]	Minimizing total distance	-	-	Optimal
CA [26]		-	-	Heuristic
Allocation [27]		Capacity	$O(n^3)$	2.5
Swap Chain [28]	Minimizing maximum distance	Capacity	$O(R \cdot \|T\|(\|T\| + \|W\|))$	Optimal
Gale-Shapley [29]	Minimizing #blocking pair	Capacity	$O(\|T\|\|W\|)$	Optimal
Closest Pair [30, 31]		Capacity	$O(\|T\|\|W\|^2)$	Optimal
Chain [32]		Capacity	$O((\|T\| + \|W\|) \cdot (\|T\| + \|W\|))$	Optimal

[a] In the constraints column, "–" is used to represent that the method supports no aforementioned constraints.
[b] In the time complexity column, "–" is used to represent the case when time complexity is not given. T and W are used to denote the set of tasks and workers, respectively.

4 Testing

In the previous sections, we have explored the Project Plan on Design. For this section, we have developed a carefully examined Testing Plan that elaborates on the reasoning behind "what are we testing" and "why"?

4.1 Unit Testing

We are using Jest and Enzyme to test our React components. Jest allows us to create testing frameworks and generate reports of issues with tests. As each component has their own test file tests that fail in one test will not affect another. Enzyme is used in conjunction with Jest to simplify tests and test in new ways like creating shallow components which are created without children to avoid not from the component currently being tested. These together allow us to create tests which can span from a single component to a system of components.

To test a component the first thing we do is create a new testing file, so the new tests do not affect other running tests. The component is shallow rendered to see if the component will render without outside component interaction. After that, the component is compared to a snapshot to see that they match properly. Once this is done a test for each function within the component is made to check outputs. After, the component is mounted, and tests are created to see if the child components are rendered correctly inside the main component. There were some issues with this system as some components could not be created as shallow without breaking. An example of this is our map function as it cannot render without the Mapbox system.

As discussed in Section 3.7, if we chose the "Greedy-GEACC" method, we would try to make the maximum profit. In contrast, if we decided on the IDA method, we would test that the algorithm produces a solution where the total distance traveled is minimized.

- Optimizing Algorithm

    - Our algorithm will be tested to ensure that it optimizes the worker-task problem's desired aspect, as discussed above. It is imperative that we test how workers' availability affects the final result.
- Alert System
    - Our alert system will be tested to ensure that it notifies the correct workers of the proper tasks and updates their schedules as changes to the initial dataset are made.

4.2 Interface Testing

The different interfaces that we needed to test were between the optimizing algorithm unit and the alert system and the database and the optimization algorithm.

- Optimized algorithm results to the Alert System
    - We needed to verify that the results from our algorithm are passed correctly to the alert system. This is done by checking the algorithm's results with the data in the alert system once it is received.
- Database information into the Optimizing Algorithm
    - We needed to make sure that data is being brought into the algorithm correctly from the database.

4.3 Results

We can query information from our DB and use that information to verify the output of various algorithms. We are also enabled to mutate collections as needed. Of the algorithms, we have tested and confirmed functionality for prioritizing the minimum time, as well as prioritizing minimum distance. We have pseudo-tested the implementation for finding workers for jobs based on desired skills but that is still in progress.

5 Implementation

As described in Section 3.3, we created a web and mobile application that implements the algorithm we designed. This application requires both a database to store the task generators and workers, and a server to allow user devices to communicate with the backend. In Section 2.1, a list of some major tasks are outlined. For most of the tasks there, as well as many tasks making up some of those tasks, we created tickets on our Trello board. Some examples of these such tickets have been tickets to set up our computers to develop the frontend or backend, to create a specific page of the web app, or to connect the frontend and backend together. These tickets all contained a title and description, and often contained a checklist of objectives to complete for the ticket as well. This helped keep our team organized and made it clear who was working on what, and the current status of our work. Because of this Trello board, our team was able to function smoothly and did not step on each other's' toes during development. We also made several unit tests over the work done for each ticket on the Trello board. In addition to unit tests, we manually tested every change to each different component of our app thoroughly by running the application during development. In section 4 we have provided more details on our approaches to testing.

The application was implemented using a client/server architecture style, as it proved to be the best method of storing data as well as offloading the potential load of the match algorithm as well as support many clients, both workers and employers. We also used a layered architecture by splitting the application into two parts and assigning a team to each component, allowing each team member to specialize in a particular system.

As our design changed, we continuously modified our plan, and this is reflected in our final implementation. Our application now uses the technologies mentioned in section 3.4, and more detailed design decisions were made throughout the implementation of the app, such as UI decisions, code style decisions, and code structural decisions.

6 Closing Material

6.1 Conclusion

We created an application that implements our goals for this semester as well as being written in a way that makes it easy for further implementation and increase in scope. We learned a lot about the design process and gained practical experience in working on a product.

6.2 References

1. Tong, Y., Zhou, Z., Zeng, Y. et al. Spatial crowdsourcing: a survey. The VLDB Journal 29, 217–250 (2020). https://doi.org/10.1007/s00778-019-00568-7
2. Amsterdamer, Y., Milo, T.: Foundations of crowd data sourcing. SIGMOD Record 43(4), 5–14 (2014)
3. Chittilappilly, A.I., Chen, L., Amer-Yahia, S.: A survey of general purpose crowdsourcing techniques. IEEE Trans. Knowl. Data Eng. 28(9), 2246–2266 (2016)
4. Garcia-Molina, H., Joglekar, M., Marcus, A., Parameswaran, A.G., Verroios, V.: Challenges in data crowdsourcing. IEEE Trans. Knowl. Data Eng. 28(4), 901–911 (2016)
5. Li, G., Wang, J., Zheng, Y., Franklin, M.J.: Crowdsourced data management: a survey. IEEE Trans. Knowl. Data Eng. 28(9), 2296–2319 (2016)
6. Chen, L., Lee, D., Zhang, M.: Crowdsourcing in information and knowledge management. In: Proceedings of the 23rd ACM International Conference on Information and Knowledge Management (2014)
7. Chen, L., Lee, D., Milo, T.: Data-driven crowdsourcing: Management, mining, and applications. In: 31st IEEE International Conference on Data Engineering, pp. 1527–1529 (2015)
8. Li, G., Zheng, Y., Fan, J., Wang, J., Cheng, R.: Crowdsourced data management: Overview and challenges. In: Proceedings of the 2017 ACM International Conference on Management of Data, pp. 1711–1716 (2017)
9. Guo, B., Liu, Y., Wang, L., Li, V.O.K., Lam, J.C.K., Yu, Z.: Task allocation in spatial crowdsourcing: Current state and future directions. IEEE Internet Things J. 5(3), 1749–1764 (2018)
10. Tong, Y., Zhou, Z.: Dynamic task assignment in spatial crowdsourcing. In: Proceedings of the 26rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, vol. 10, no. 2, pp. 18–25 (2018)

11. Kazemi, L., Shahabi, C.: Geocrowd: enabling query answering with spatial crowdsourcing. In: Proceedings of the 20th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 189–198 (2012)
12. Deng, D., Shahabi, C., Demiryurek, U.: Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 314–323 (2013)
13. Kazemi, L., Shahabi, C., Chen, L.: Geotrucrowd: trustworthy query answering with spatial crowdsourcing. In: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 304–313 (2013)
14. To, H., Ghinita, G., Shahabi, C.: A framework for protecting worker location privacy in spatial crowdsourcing. PVLDB 7(10), 919–930 (2014)
15. Chen, Z., Fu, R., Zhao, Z., Liu, Z., Xia, L., Chen, L., Cheng, P., Cao, C.C., Tong, Y., Zhang, C.J.: gMission: a general spatial crowdsourcing platform. PVLDB 7(13), 1629–1632 (2014)
16. Li, Y., Yiu, M.L., Xu, W.: Oriented online route recommendation for spatial crowdsourcing task workers. In: International Symposium on Spatial and Temporal Databases, pp. 137–156 (2015)
17. Tong, Y., She, J., Ding, B., Wang, L., Chen, L.: Online mobile micro-task allocation in spatial crowdsourcing. In: 32nd IEEE International Conference on Data Engineering, pp. 49–60 (2016)
18. Tong, Y., She, J., Ding, B., Chen, L., Wo, T., Xu, K.: Online minimum matching in real-time spatial data: experiments and analysis. PVLDB 9(12), 1053–1064 (2016)
19. Tong, Y., Wang, L., Zhou, Z., Chen, L., Du, B., Ye, J.: Dynamic pricing in spatial crowdsourcing: a matching-based approach. In: Proceedings of the 2018 ACM International Conference on Management of Data, pp. 773–788 (2018)
20. To, H., Shahabi, C., Xiong, L.: Privacy-preserving online task assignment in spatial crowdsourcing with untrusted server. In: 34th IEEE International Conference on Data Engineering, pp. 833–844 (2018)
21. GAIA Open Dataset (2019). https://outreach.didichuxing.com/ research/opendata. Accessed May 26 2019
22. To, H., Fan, L., Tran, L., Shahabi, C.: Real-time task assignment in hyperlocal spatial crowdsourcing under budget constraints. In: IEEE International Conference on Pervasive Computing and Communications, pp. 1–8 (2016)
23. She, J., Tong, Y., Chen, L., Cao, C.C.: Conflict-aware eventparticipant arrangement. In: IEEE 31st International Conference on Data Engineering, pp. 735–746 (2015)
24. Cheng, P., Lian, X., Chen, L., Han, J., Zhao, J.: Task assignment on multi-skill oriented spatial crowdsourcing. IEEE Trans. Knowl. Data Eng. 28(8), 2201–2215 (2016)
25. To, H., Shahabi, C., Kazemi, L.: A server-assigned spatial crowdsourcing framework. ACM Trans. Spat. Algorithms Syst. 1(1), 2 (2015)
26. U, L.H., Yiu, M.L., Mouratidis, K., Mamoulis, N.: Capacity constrained assignment in spatial databases. In: Proceedings of the 2008 ACM International Conference on Management of Data, pp. 15–28 (2008)

27. Bei, X., Zhang, S.: Algorithms for trip-vehicle assignment in ride-sharing. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, pp. 3–9 (2018)

28. Long, C., Wong, R.C., Yu, P.S., Jiang, M.: On optimal worstcase matching. In: Proceedings of the 2013 ACM International Conference on Management of Data, pp. 845–856 (2013)

29. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. Am. Math. Mon. 69(1), 9–15 (1962)

30. Corral, A., Manolopoulos, Y., Theodoridis, Y., Vassilakopoulos, M.: Closest pair queries in spatial databases. In: Proceedings of the 2000 ACM International Conference on Management of Data, pp. 189–200 (2000)

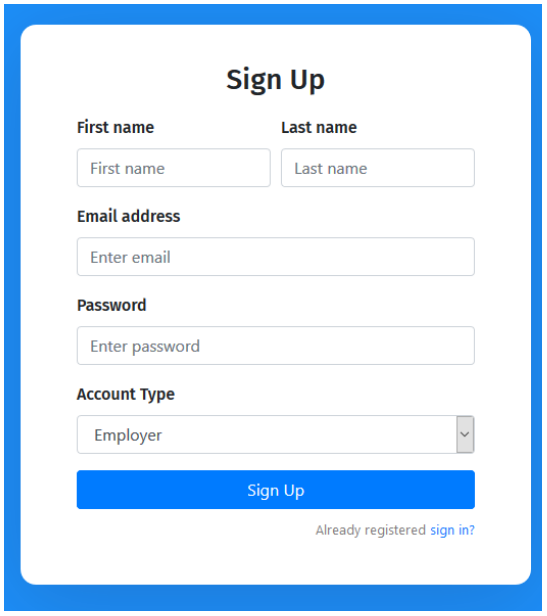
31. Yang, C., Lin, K.: An index structure for improving closest pairs and related join queries in spatial databases. In: International Database Engineering & Applications Symposium, pp. 140–149 (2002)

32. Wong, R.C., Tao, Y., Fu, A.W., Xiao, X.: On efficient spatial matching. In: Proceedings of the 33rd International Conference on Very Large Data Bases, pp. 579–590 (2007)

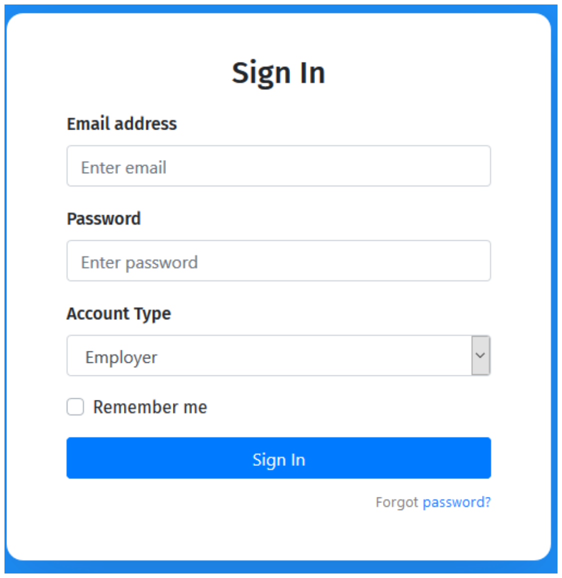
6.3 Appendices

Appendix I - Operation Manual

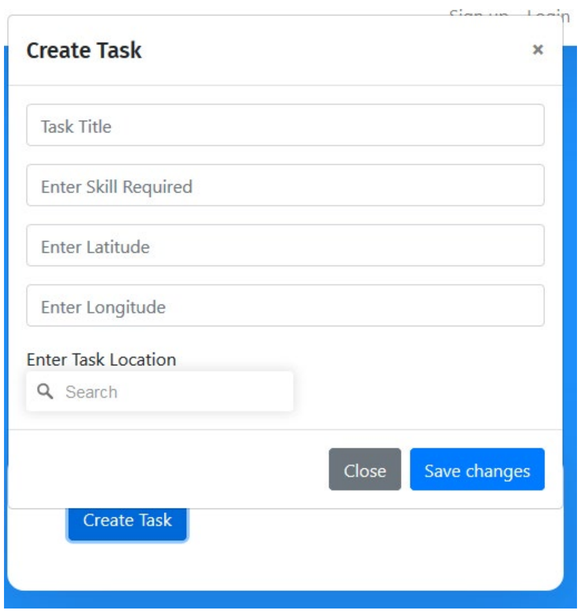
- Instructions for Creating an Account or Signing into an Existing Account:
  - Upon opening the application on a computer or mobile device, you will be greeted with the Create Account page. If you are new to the application, then fill in your first and last name, as well as your email address and a password you will remember. Be sure to select the kind of account you wish to create from the account type dropdown. If you wish to create tasks for others to complete, select “Employer,” and if you wish to complete tasks other people have made then select “Worker.” The below figure depicts the Create Account page.



- If you have used our application before and already have an account, then click the "sign in" link in the lower right. This will take you to the Sign In page. Enter the email address you used to create your account and your password, then ensure the account type dropdown has the correct type of account selected. Below is an image of the Sign In page.

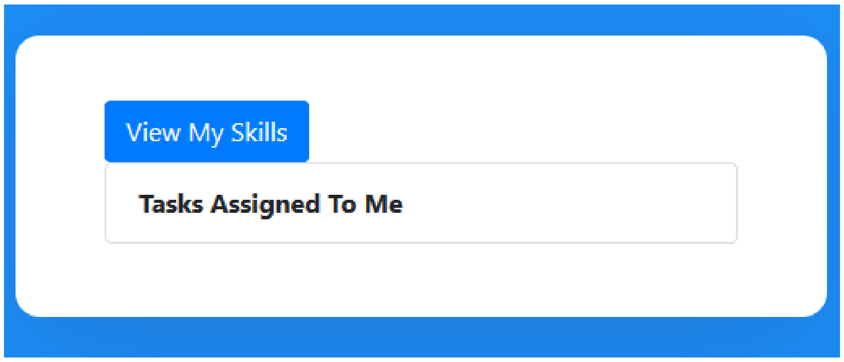


- Instructions for Employers (also known as Task Generators)
  - After creating your account or signing in as an employer, you will be navigated to the tasks page. This page displays a list of all tasks you have created. This list will be empty when you have just created an account. Click the "Create Task" button to open the create task modal. Enter a title or short description for your task, the skills your task requires, and search for a location in which your task will take place. Press the Save Changes button to finish creating your task. Your task will then appear in your list of tasks created, as shown below.

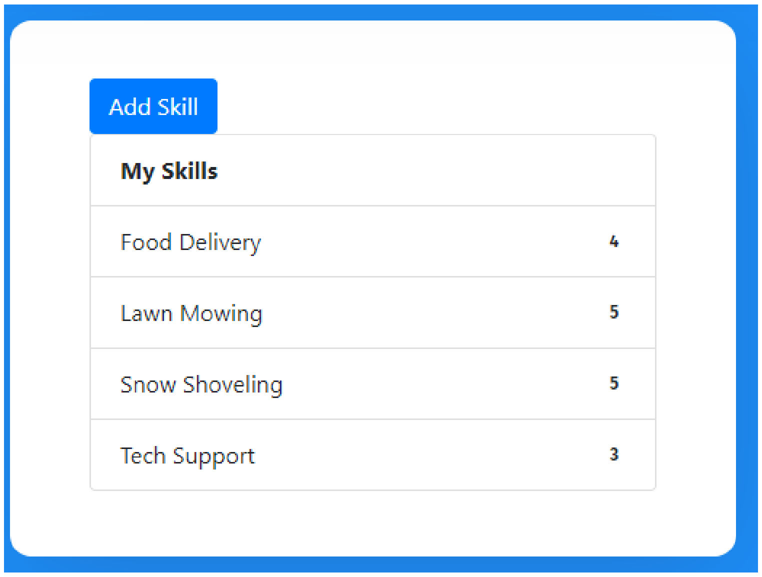


- A worker will then be assigned to your task and will complete it timely, so you can just sit back and relax!

- Instructions for Workers
  - After creating your account or signing in as an employer, you will be navigated to the assignments page. You may be asked to allow this website to use your location. Please allow this so we can automatically update your location to find more relevant tasks to assign you. This page displays the tasks you have currently assigned to you. Just after creating your account, you likely will not have any tasks assigned to you yet. An image of the assignments page is below.

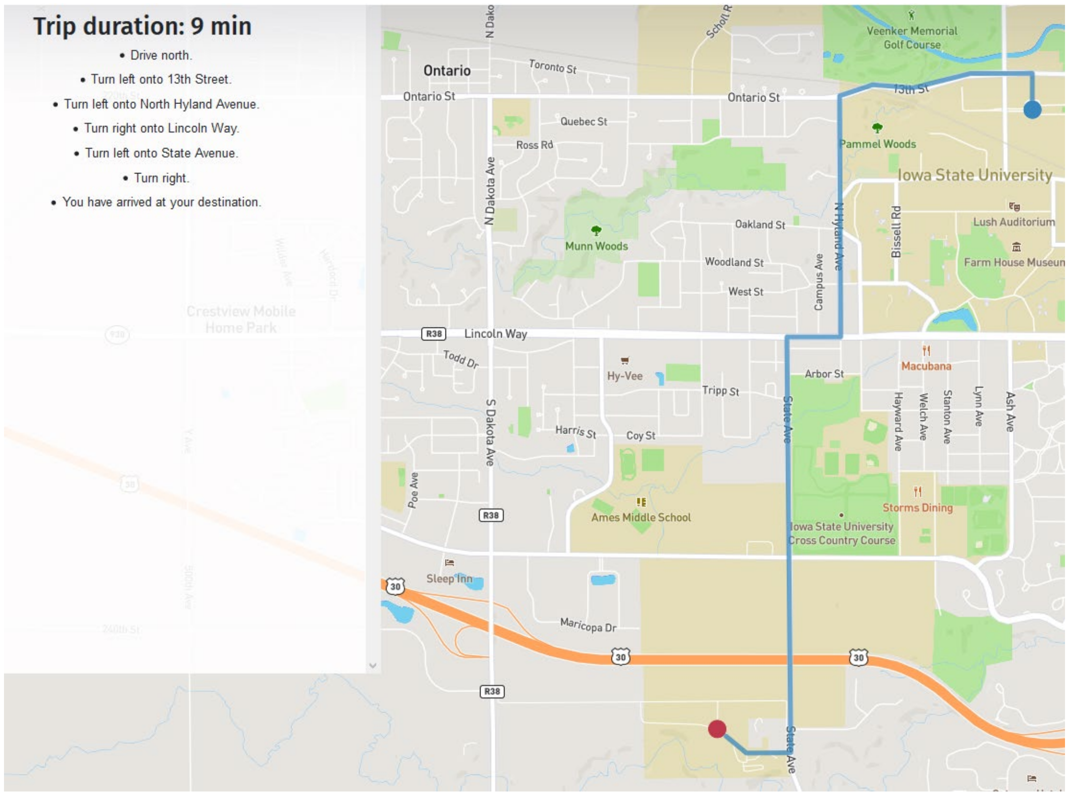


  - The reason you likely will not have any tasks assigned to you is because you have not set any of your skills yet. Click the "View My Skills" button to navigate to the skills page (shown below). This page lists all the skills you have, along with a rating 1-5 of how well you perform each skill. This list will also be initially empty. To add skills to this list and to your worker profile, click the "Add Skill" button. This will bring up a modal that asks you for the name of the skill and a rating of 1-5 of how well you think you perform this skill. After entering these fields, click the "Save Changes" button, and your skill will be added! The more skills you have, the more likely you are to be assigned tasks.



My Skills	
Food Delivery	4
Lawn Mowing	5
Snow Shoveling	5
Tech Support	3

- After adding skills to your profile, you can use the navigation bar at the top of your screen to navigate back to the assignments page or use the back arrow. Once here, you may see a task assigned to you. If not, then you are not the best candidate for a nearby task yet, sit tight! Once you do have a task assigned to you, the title or short description the employer entered for it will appear in the list on the assignments page. Clicking the task will take you to the maps page that will display a route for you to follow to the task's location and step-by-step directions to that task.



Trip duration: 9 min

- Drive north.
- Turn left onto 13th Street.
- Turn left onto North Hyland Avenue.
- Turn right onto Lincoln Way.
- Turn left onto State Avenue.
- Turn right.
- You have arrived at your destination.

- Follow the directions on the screen to arrive at your task. At any time, you may use the navigation bar at the top or the back arrow to go back to the assignments page to view the task again.