3 Design

3.1 Design Context

3.1.1 Broader Context

Our broad context is a situation where we have N job-sites, M workers, who each have their own skill descriptions and respective locations. As we are crowdsourcing our workers will be assigned to these job-sites based on various factors depending upon the work situation. We are targeting individual consumers and service providers by our model. This application will increase competition in our communities and thus provide better and more affordable services.

Below are a few areas and descriptions of how the project would have the ability to impact them.

Area	**Description**	**Examples**
Public health, safety, and welfare	Workers - 1. By increasing economic opportunity for workers in the community, this will also positively impact overall health and welfare, as workers will be able to more comfortably afford more of their expenses. 2. The workers who respond to the tasks may take the risk of responding to malicious users: i.e. users who "pretend" to have an issue, but make a job request for another (possibly malicious) reason. Mitigating the amount of travel workers do is also important, which would decrease the likelihood of a traffic accident or other travel-related issues. Users - 1. Users should see health/safety increase with the use of our solution, as users may request jobs that, if left uncompleted, could impact their physical or mental health and welfare. 2. Similar to workers, users may take the risk of "malicious" workers. These would be workers who use the product and are in some way negligent to the job they are assigned. This would also have the same effect if a user posts a job they need done urgently, but the job is never completed by a worker.	Workers - 1. A worker who previously struggled in being able to find jobs is given more opportunities (which results in increased mental wellbeing for workers) as a direct result of the algorithmic solution implemented. 2. A malicious user uses the app to request a job, but when the worker arrives at the job site, neither the user nor the job is anywhere to be found. Another applicable example would be requesting an Uber driver, but cancelling last-minute, resulting in wasted time and money for the worker. Users - 1. A user posts a job request for a leaky pipe which is quickly completed by a worker. If left unattended, this pipe could've caused harmful effects to the user's health. 2. A user makes a job request to fix a leaky pipe that is assigned (unintentionally) to a malicious or negligent worker that doesn't show up, or does not complete the job. This could result in not only damage to the user's plumbing and infrastructure, but also health risk to the user due to mold if not attended to in a timely manner.
Global, cultural, and social	It will help organize professional services and their execution by professional workers and help with the organization and	Development or operation of the solution would violate a profession's code of ethics, implementation of the solution

	transportation of those with similar social desires.	would require an undesired change in community practices
Environmental	By dynamically optimizing routes of workers the emissions from workers vehicles will be reduced between jobs.	With algorithmically calculated optimized routes for the workers, excess car emission can be controlled helping the environment.
Economic	It enables more competition amongst service providers which in turn can provide more competitive pricing. More competition also promotes better service and enables more flexibility for the customers. The proposed solution would also create more jobs for workers in the community/area in which it's implemented.	If a customer insists on having his job finished in a certain hour and is willing to pay more, the system should enable such assignments.

3.1.2 User Needs

Workers need to . . . :

1. be able to accept tasks to confirm for the users and crowdsourcing algorithm.
2. be able to get the location of the requested task to be able to arrive and accomplish the task.
3. be able to post their location, skills and task status because both clients and the crowdsourcing algorithm require this information.
4. be able to finish accepted tasks in a timely manner to ensure work done via the app is done properly.
5. regularly monitor task schedules for any updates because of the dynamic nature of the problem.
6. be able to file feedback/issues if any so it can be handled in a timely manner.

Client/Requesters need to . . . :

1. be able to post tasks so there are tasks to calculate assignments for, and so the application can serve their needs.
2. be able see the status of assignment/workers in real time so they can stay up to date on the status of their task.
3. be able to confirm the task completion to ensure the application provided them proper services
4. be able to approve funding for the requested task so the algorithm can provide them with a worker in their price range, and so the worker can receive payment.
5. be able to file feedback/issues if any so it can be handled in a timely manner.

Administration needs to . . . :

1. be able to assess new workers based on the given information to ensure proper sign up, and qualified workers.
2. manage customer-worker conflicts and feedback so they can be handled in a timely manner.

3. be able to issue incentives to workers based on their performance so that the application can keep workers motivated, and draw in new users.
4. be able to build a customer appreciation strategy to boost business and user satisfaction.
5. be able to manage privacy protection to ensure all user data remains private.

3.1.3 Prior Work/Solutions

Detailed background information on the problem of dynamic spatial crowdsourcing, and programmatic solutions, can be found in the "*Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach*" report. This report can be found in section 3.4.

In the modern day, there are actually many applications which intend to solve the issue of spatial crowdsourcing for a specific subgroup of tasks. A few prominent examples are rideshare services such as Uber or Lyft, and food delivery services such as DoorDash and Eat Street. These companies provide similar tools, however they only handle location data; not worker skills / and differing tasks. Due to this, these popular solutions actually lack many of the criteria shown in figure 1, and cannot truly be considered examples of solutions to our problem.

Summary of Notations

Notation	Definition
s	Spatial task
l_s	Location of spatial task s
e_s	Expiration time of spatial task s
$maxW_s$	Maximum acceptable workers for task s
w	Worker
l_w	Current location of worker w
d_w	Destination of worker w
t_w	Deadline of worker w
$speed_w$	Movement speed of worker w
R	A task sequence
S_w	A task set for w
$VTS(w)$	A valid task set of w
$t(l)$	The arrival time of particular location l
$c(a, b)$	Travel distance from a to b
A	A spatial task assignment

Figure 1: Zhao, Y. et. all (2019, June 12)

We may be able to draw information from these services such as location assignment and UI designs, but we will need to make substantial changes in order to provide a proper solution. With these changes in mind, the examples and information provided in the "*Destination-aware task assignment in spatial crowdsourcing"* report will likely prove to be valuable to our development process. This report includes notes on optimization, algorithms, and more general details that need to be taken into consideration.

This advancement makes us stand out from any existing competition. Our service will take the concept that has caused these apps to take the world by storm, and we will combine it with worker skills and tasks to provide a general user solution to spatial crowdsourcing. Rather than simply having Geographical Information System functionality with a basic simple task, we will have detailed functionality to provide clients a way to get all services they may need in one convenient location. We will also subsequently attract more workers, as we will serve more professions. Workers may also have a better experience on our services than other popular services. This is due to the dynamic spatial crowdsourcing algorithm giving them their optimal job rather than the worker needing to manually find and accept tasks.

3.1.4 Technical Complexity

From a technical side, our project has three main components: the frontend, backend and database. Alongside these it will need to make use of at least one outside API as well, a location API. Our frontend will be composed of two client types, a web browser and mobile, both of these will need to provide the functionality for the user to provide information input to the backend and query the backend data for general profile information, task assignments/updates and location data. Our mobile application will also need to be able to provide location updates for workers to the backend.

1. The backend will need to store all of our data in the selected database. As this data changes it will need to put data through the algorithm and update worker assignments. Part of processing the data through this algorithm will require querying a location API.
2. The frontend will provide functionality to query the backend and provide task assignment updates to workers. On the mobile application side of the front end the application will also need to be able to provide location updates for user reference and also for algorithm usage.
3. Our current target is a more general purpose algorithm. While it is comparable to some existing companies/solutions, like Uber or GrubHub, it is not equivalent, as our algorithm will be taking multiple task types into consideration, as opposed to a single category. This adds an extra layer of complexity to the problem with us having to take more than just spatiotemporal data into account.

Technical aspects that increase the complexity of the project:

- Our project will have three main technical interfaces
    - Backend to database, providing the algorithm with the data that it needs and writing to the database.
    - Frontend to backend, providing user data for the database, location data, and updating users and workers with task info.
    - Backend interfacing with a location api to assist with the spatial dependent portion of assigning tasks.
- The frontend is not one singular component but will need to have components written for both web and mobile.
- ReactJS and Spring Boot are new to some members of the team and will require knowledge acquisition.

Internal complexity

Components and subsystems: Technique identification(processing and database),location services, dynamic route optimization,

Scientific, mathematical, or engineering principles: Location detection,

External complexity

Our project has multiple functional requirements that will match or exceed current state-of-the-art standards, such as: error recognition, dynamic location monitoring, task optimization, input processing, and dynamic contingency handling.

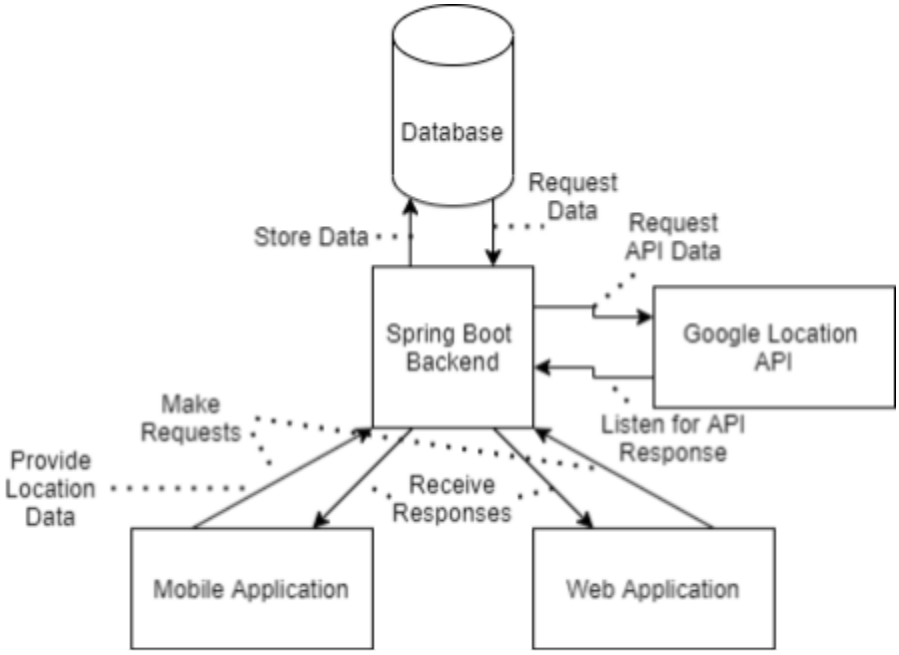

Figure 2: Diagram of Technical Layout

3.2 Design Exploration

3.2.1 Design Decisions

Some key design decisions we have made regarding our proposed solution are as follows. Note these may be subject to change with time or at client discretion:

1. Real-time tracking through google maps api
2. Front-end implementation using ReactJs
3. We will develop our mobile application using React Native
4. For our backend services, we will utilize the spring boot framework.
5. MySQL Db will be used for data storage, using tools such as postman and SQL workbench to run tests.
6. The server provided to our development team will be used to host the application's backend.

3.2.2 Ideation

For the backend implementation, we debated between mongodb(NoSQL) and mysql(relational). We settled on MySQL db because it is a good balance of ease of use, proficiency, and easily accessible documentation. This platform also has an active community and support teams that can be of assistance if appropriate documentation cannot be found.

For the front-end implementation, we decided to go with ReactJs. We looked into other options such as Angular, JavaScript, Vue.js, and Ember.js. Considering our team's experience, ReactJs was chosen as the best choice for our team. Also, ReactJs is easier to understand and execute than the latter options. One of the cons of choosing Angular is the difficult debugging that it comes with, but we prioritized the performance and efficiency of it.

As for general ideation techniques, these decisions were narrowed down using the information provided in prior sections of this document. We balanced our development team's skills with the needs we had documented, and found the best compromises that suited both categories. Once we settled on a list of possible languages and frameworks, tools such as the weighted decision matrix in the following section were used to make the final decision.

3.2.3 Decision-Making and Trade-Off

Weighted Decision Matrix [Scale: 1-10] :

OPTIONS	Personal Preference	Industry Standard	Developer experience	Ease of Use/learn	Ease of Maintenance	Total
Criterion Weight	20%	10%	25%	25%	20%	100%
ReactJs	9	8	9	10	8	8.95
Angular	6	8	7	9	9	7.8
Vue.js	7	7	8	10	8	8.2
Ember.js	1	3	0	3	5	2.25

As it is shown, ReactJs has the highest total point.

3.3 Proposed Design

There has not been implementation yet, project details have not yet been solidified to a point that we have had the option to spend time on that.

3.3.1 Design Visual and Description

Figure 2 will be our reference point for the visual description of the design. Our design only includes software components, as such we have a block diagram of the software components and have no further breakdown of any physical elements.

The backend will be our central component taking input from the frontend and pulling data from the database and location api to provide input to our task assignment algorithm. Once tasks are assigned, clients and workers will be able to see tasks and relevant data with elements like worker location updated in real-time and displayed to appropriate parties. The database will be mySQL written in Spring Boot and the front end will be written in ReactJS. Connections between the different elements are labeled in figure 2 with the functionality they will primarily be used for.

3.3.2 Functionality

Describe how your design is intended to operate in its user and/or real-world context. This description can be supplemented by a visual, such as a timeline, storyboard, or sketch.

How well does the current design satisfy functional and non-functional requirements?

Functional requirements:

- The mobile and web app should be able to display status of assignment and worker in real-time

  ->React allows us to develop both mobile and web apps.

- Only appropriate users should be able to see the location/info of workers

  ->Give permission only to appropriate users.

- Interpret existing worker data to decide optimal routes and assignments

  ->With the GPS API, collect users' location and use the algorithm to make an optimal decision.

- The applications should be scalable for multiple users

  -> Ensure the scalability of the application by choosing the right technology and practices. Using DevOps to manage the project and developing the application.

- Job requests are updated in real time

  -> Workers are able to provide regular updates on the request through the UI.

- Workers must be able to post their availability and location

  -> Workers are prompted to give their location when posting a job, with real-time data on the jobs they're working.

- Application should calculate the optimal assignment of worker to task

  -> Designing an algorithm for assigning the optimal assignment to the worker.

- Users should be able to view status of their requests in real-time

-> Updates made by the worker should be visible at the customer's UI.

- Proper authentication for user accounts

  -> Email and Phone Number Verification for all new user Accounts.

- Data storage and tracking of completed tasks

  -> All user and assignment data is synchronously stored in the database.

- Optimized for fast response time

  -> Ensure industry standard coding practices while using multi-threading for faster response time.

Non-functional requirements:

- Accessible from mobile devices and PC

  -> Using React will allow us to develop for both mobile and desktop.

- Design should be extensible.

  -> Practice clean code and proper documentation of each task assigned.

- Data should be stored safely and accurately, and its integrity shall be maintained across API hits

  -> Store data in the database and use a queue pattern in order not to lose the data.

- Clean and best practice code to improve future maintainability

  -> Evaluate team members' performance every other week to ensure all the best practices are being followed.

- Application should be functioning reliably

  -> Protects the integrity, availability, and confidentiality of the application and it's users. Also, prepare an emergency plan in case there's a breach.

- Privacy and security of workers

  -> Constant testing of our application/code will allow us to find and avoid any data leaks. Use of 2 factor authentication at user login.

- Intuitive and "clean" web and mobile UI

  -> Make sure we consult our group with every UI change to prevent clusters in the UI.

3.3.3 Areas of Concern and Development

Based on your current design, what are your primary concerns for delivering a product/system that addresses requirements and meets user and client needs?

What are your immediate plans for developing the solution to address those concerns? What questions do you have for clients, TAs, and faculty advisers?

Concerns:

- Making the application scalable, i.e. adding multiple users and workers over time.
- Making sure that the application is easy to navigate for new users.
- Making the real time location of workers as precise as possible.
- Smooth integration of all the components of the application.
- Making sure the app is optimized for fast response times.
- Making sure the user's information is secure.

Addressing the concerns:

Team will follow a clean coding principle for a smooth integration and merge their code from time to time. As none of the team members has major experience with developing real time applications, we will consult the faculty on ways to mitigate delay within the application. In order to optimize our app for fast response times, we plan to only implement the necessary functions first in order to prevent overcrowding of unnecessary functionalities. Will try to get as much user feedback as possible, on the UI and new user navigation.

3.4 Resources Referenced

Zhao, Y. et. all (2019, June 12). *Destination-aware task assignment in spatial crowdsourcing: A worker decomposition approach*. IEEE Xplore. Retrieved September 22, 2021, accessed from https://ieeexplore.ieee.org/document/8735884.