FYP22033

Recommendation-Based Community Helpers Mobile Application

Detailed Project Plan

**Supervisor:**
Dr. Chim T W

**Group Members:**
Lau Chun Leung 3035757066
Wong Ho Yin 3035762580
Wong Sze Lung 3035762190
# Table of Content

1. Project Background 2
   1.1. Motivation 2
   1.2. Related Work 2

2. Project Objective 4
   2.1. Overview 4
       2.1.1. FE-1: Request Helper 4
       2.1.2. FE-2: Browse Requests 4
       2.1.3. FE-3: Helper Matching 5
       2.1.4. FE-4: Helper Search 5
       2.1.5. FE-5: Chat Room 5
       2.1.6. FE-6: Transaction 5
       2.1.7. FE-7: Content Management System (CMS) 5
   2.2. Proposed Brief Flow Diagram 6

3. Methodology 7
   3.1. Proposed System Architecture 7
   3.2. Frontend 7
   3.3. Backend 8
   3.4. Recommender System 9
       3.4.1. Collaborative Filtering 9
       3.4.2. Hybrid Model 10
       3.4.3. Graph Neural Network 10

4. Project Schedule and Milestones 11

5. Reference 12
1. Project Background

Imagine that you are working so hard at midnight. You are so hungry but your refrigerator is empty. Even worse, your home Wi-Fi suddenly mal-function for no reason. What could you possibly do? Usually, you can only eat cup noodles and wait until the other morning for help. What if there is a solution in the near future that you would be able to seek help from your neighborhood?

1.1. Motivation

We notice that people in a neighborhood have many different special skills. They also have a lot of problems that need to be solved everyday or every minute. There will always be some problems that people cannot solve on their own. We believe people would appreciate that there is someone who has the knowledge and is willing to help you out to solve a particular problem. For an alternative perspective, if you have a lot of spare time and you have the knowledge or professional skills shared with your neighborhood. Also, you would be able to earn money as long as you become a helper. It should be a win-win situation for the community. We want to build a platform to make people strongly connect with the neighbors. Let us share neighbors a helping hand.

This project is going to create a multi-platform application that merges your neighbors into your daily life. For example, the service provider can post their profile on the platform and illustrate what kind of personal skills or materials they can give it to you as a service. They might be good at cooking, provide tools that you need or even help you repair furniture and appliances. The helper can receive an amount of earnings in return.

1.2. Related Work

There are several applications with similar ideas in the market. All of them have a common objective which is better utilize some idle resources in our society and use them to serve others. However, those existing platforms have room for improvement. The following would highlight the similarity and improvement of the platforms in the market.

Gaifong

Gaifong is an application that allows users to rent stuff from their neighborhood [1]. You can rent books or even repair equipment from your neighborhood. It provides location service to find your thing that can be borrowed nearby. Users could receive rents for lending their own stuff to others.

The idea of this application is very similar to our idea. But instead of lending stuff to others, we want to provide service to our users. For example, instead of lending electric drills, we want the neighborhood to provide the repairing service. With this alternative perspective of sharing resources, we can provide some services that users are not able to complete or want someone to do it for them.
Toby - Hire Local Service

The business operation of Toby is very similar to the application that we are going to produce. They provide a lot of services on the platform. Users can browse different categories and choose the service they want [2]. Those services came from small and medium sized enterprises on their platform which gives a good chance for those businesses to grow up. The services they provide might be more professional. For example, you must have a business register for a moving company to start providing service on the platform. What we want to achieve is that you do not need to be a professional to help others. As long as you are able to complete the task and your help targets are near you. You can start providing service.

Image 2: Brief look on what service Toby is providing
2. Project Objective

2.1. Overview
In this project, we aim to develop a multi-platform (iOS & Android) application that includes the following main features,
- FE-1: Request Helper
- FE-2: Browse Requests
- FE-3: Helper Matching
- FE-4: Helper Search
- FE-5: Chatroom
- FE-6: Transaction
- FE-7: Content Management System (CMS)

2.1.1. FE-1: Request Helper
Seeking help from the community is one of the main features of this application. In this app, users can create a request to the system to seek help from the helpers. When a user creates a new request, it should be provided some description/detail of the request, such as
- Category
- Short description of the request
- Date & time
- Location
- Illustration image/photo
- etc.
Note that, the user can specify the time period to schedule the requests when creating a request. After the request is created, the system would start to find the most suitable helper for this request. For more detail on helper findings can refer to FE-3: Helper Match section. Once the system found a suitable helper, that helper would receive a notification to invite him/her to that helper request. That notification could be sent via system notification (iOS/Android) or email, etc. From the helper's perspective, he/she could review the request detail to decide whether to accept this request or not. If the helper accepts that request, then a help event is established.

2.1.2. FE-2: Browse Requests
Apart from passively waiting for a helper request, the helper can actively browse over the pending helper request. When the helper finds a helper request that he/she is interested in, the helper can directly accept the request to establish a help event for the requester without waiting for the request notification. The list of pending helper requests should be personalized by each helper. That means each helper would see a different list of pending helper requests based on their feature, such as help history, age, strengths, etc. The personalized request list is expected to be generated by the recommender system in this system.
2.1.3. FE-3: Helper Matching

Helper matching is the key component in this system as most of the features, such as FE-1: Request Helper and FE-2: Browse Requests, would be based on the helper matching result. The matching process would be driven by the recommender system. In this feature, we would develop a system that can find a list of suitable helpers by giving a helper request (refer to FE-1) and find a list of suitable helper requests by giving a helper (refer to FE-2). The criteria for suitable helpers is that the helper is interested in that request and also knowledgeable in it. We expect that the system would consider a basket of factors, including the request nature, helper’s background, etc., to perform the matching.

2.1.4. FE-4: Helper Search

Besides matching a helper with a user, our app also allows users to browse for services that their neighbors provide. A list of services will be shown when users search the keywords by search bar. Instead of requesting help and matching a helper, users can discover a different kind of service themselves. The results might prioritize in helper popularity, displacement, credit level, promotion, or the search history and request type of user in the past.

2.1.5. FE-5: Chat Room

Chat Rooms is a feature that allows users to interact with each other. Users and helpers will negotiate the details of the service. After the system matches a user to a helper, chat room features will be enabled and they can talk more about the details and the deals. They can exchange their information such as the exact meet-up location or the fees. It would be a traditional chat room that allows users to send and receive messages.

2.1.6. FE-6: Transaction

Helping people is one of the main goals of our project. However, we also want our users to increase their income with their special ability in their spare time. We also charge 5% of each deal as our service fees.

If a user raises a new post and would pay HKD$100 as a remuneration, our platform will cost the user a HKD$105 (HKD$100 + 5% of HKD$100).

The platform allows users to pay for the service with credit cards alongside with apple pay and google pay or Faster Payment System (FPS). We would temporarily keep the charge. If the service is done, we would release the remuneration as the post mentioned, to the service provider in a certain period of time (e.g. a weekly automatic direct deposit). This is the workflow of payment and we charge an extra fee to ensure the platform delivers consistently great service.

2.1.7. FE-7: Content Management System (CMS)

In order to make the administration of the app easier, we plan to develop a simple webpage to manage our content inside the application. It provides several features:

- A dashboard for some analytics use:
- what service is popular
- How much we made

- Admin page for modifying database data
  - browsing customer data (profile/service/request)
  - UI for checking anything from the database if there are any complaints for customer or helper

- difficult for admin to technical stuff (SQL/GraphQL/API) to modify data
- make it easier for us to demo the application with better UI/UX (without coding to modify the app data)
- Allow admin to handle settlement

2.2. Proposed Brief Flow Diagram

![Flow Diagram]

Figure 1: Proposed Brief Flow Diagram
3. Methodology

3.1. Proposed System Architecture

3.2. Frontend

As this project would run on both iOS and Android devices, we also want the layout to be rendered in a native environment. The most convenient way is using React Native with Expo for making the JavaScript XML (JSX) code to be compiled to native UI elements for the iOS and Android platforms. Briefly introduced to JSX, JSX allows us to write HTML elements in JavaScript and place them in the DOM. Most people find it helpful and as a visual aid when working UI inside JavaScript code and they always use JSX with React to describe what the UI should look like [3]. So, How does React Native work?

Before explaining the workflow or React Native, let us dive into React.js first. React.js is a JavaScript library which is independent from React Native and it is used for building user interfaces. So, React.js is typically used for web development. But, if working with React.js, there is actually another library called ReactDOM that adds actual web support. Since React.js library itself without ReactDOM is platform-agnostic. In order to build a web application, React.js need to be in conjunction with ReactDOM. It means React library does not care about the underlying platform and just provides tools for managing state or building virtual component trees. Then, an extra library like ReactDOM for translating the result that React.js produced to an actual platform like browser.
Obviously, what React Native basically does, it is an alternative to ReactDOM for building mobile applications. React Native gives a collection of special React components, which are built-in components, that can be used in JSX code. Also, those components are compiled to native UI elements for iOS and Android platforms. Therefore, React Native will handle the compilation step.

In addition, React Native also exposes native platform APIs, developers can call it and use the feature such as using the device camera in JavaScript code. Overall, React Native is like ReactDOM, it does not connect React to the web platform but iOS and Android platforms instead.

It is clear that the JSX elements will be compiled to native code of mobile applications. But the JavaScript code outside of JSX like the statement, function code, or state management. These logic codes will not be compiled. Basically, it is running on a JavaScript thread, hosted by React Native in the native app that was built. So, React Native spins up a JavaScript process as part of the native app and manages the process for us.

Expo is a collection of tools and services developed around React Native [4]. It has numerous features, the most important of which is that it can get you writing a React Native app in minutes. It has a managed app development workflow. It means creating a project or tapping into native device functionalities is easy. Expo is a crucial tool for running the React Native app on testing devices and simulators as well as building the app to upload to the app store on iOS or Google play on Android.

3.3. Backend

Comparing different languages and frameworks, python (fastapi), php (laravel), JavaScript (Node.js), we choose our tools based on the degree of convenience and the performance of building scale applications. Node.js fulfills both of our criteria and it would be the tool for building our backend.

Node.js is a JavaScript runtime environment that allows us to run JavaScript on server-side [5]. The biggest advantage of using JavaScript on our backend side is that we could align the choice of programming language. We are not required to spend extra time learning another language such as PHP/python in order to start our project. So that all team members could work on different parts of the application without learning multiple languages. It is easier and efficient for us to debug the program if we are all on the same page. Node.js also provides asynchronous features which allow servers to process multiple requests at the same time [5]. It could provide better performance compared to PHP which uses blocking I/O. When a non-blocking API is called with an asynchronous function in Node.js, the server is still free to process other API. These features largely increase our server performance.

There are also several frameworks to choose for building web applications. We have examined nest.js and express.js. After comparing what features each framework can provide, we are considering using Nest.js in our project.

Nest.js is Node.js framework that helps building server-side applications. In our project, Nest.js is used to build API for frontend and communicate with databases. It is built on top of
express.js which simplifies the idea of using express.js but also allows us to use other third-party modules made for express [6]. Nest.js is an opinionated framework which has a set of rules for use to follow for avoiding errors. For example, Nest.js uses a model-view-controller design pattern (while express.js is not) which provides proper structure to design our application [7]. It also supports typescript which is JavaScript that supports static type. Using typescript in our application would be more reliable because there would be less error on type and make our development more smooth and efficient.

Nest.js also comes with a built-in command-line tool that would largely increase our productivity [7]. For example, it can use commands to generate file templates for controllers and providers. We are not required to remember lengthy code. Once we use a certain command, we could start work on the design and the logic of the application. It is much better than spending time on writing template code multiple times. It also supports various databases such as mongoDB, postgreSQL. In our project, we plan to use GraphQL for querying APIs since it gets back all the data we need in a single request. The supports from Nest.js is one of the deciding factors that we choose as our web framework.

3.4. Recommender System

As a recommendation-based application, users are expected to rely on the system’s recommended offer heavily. Therefore, the performance of the recommender system is critical to this project. Recommender system is a system that provides item suggestions to a user who probably would be interested in that item [8]. In this project, we aim to develop a machine learning based recommender system model to drive the matching system in our application. To achieve that, we would explore different recommender system approaches, such as collaborative filtering, content-based filtering, Graph Neural Network (GNN) based recommendation model, etc.

3.4.1. Collaborative Filtering

Collaborative Filtering (CF) is a very popular approach in recommender systems. The idea behind CF is that similar users should share a similar interest in items [9]. In our setting, “items” in CF would likely be the helper request, and “users” would be our application user. For example, suppose user A has completed a cooking request and user B is similar to user A, the system would consider user B also likely interested in cooking though user B has not completed any cooking requests. One of the popular methods to implement CF is matrix factorization, such as Funk MF, SVD++, etc. Matrix factorization is trying to decompose the user-item interaction matrix to predict the item rating by a user. Nowadays, with deep learning development, He, Xiangnan et. al. proposed Neural Collaborative Filtering (NCF) which is using deep learning techniques, and claims that it can perform better recommendations over the traditional method in 2017 [10]. In this project, NCF could be a potential option model to perform helper recommendations. However, in general, most Collaborative Filtering approaches would suffer from the cold-start problem. The cold-start problem refers to a problem that a new user or a new item does not have any interaction with other users/items [11], and the recommendation quality for those new users/items would become really bad. As for a solution, a hybrid model could be considered.
3.4.2. Hybrid Model

Hybrid Model is a recommender system that combines Collaborative Filtering and Content-based filtering. Content-based filtering is another recommendation approach that would only consider a user’s features/preference to recommend items that share similar features then that user without considering other user data [12]. When a new user does not have enough interaction with the system, using Content-based filtering can alleviate the cold-start problem. Since this project is a new project, we do not have any prior user data. Therefore, the cold-start problem is foreseeable. Wide & Deep model is a hybrid model that was invented by Google in 2016 and it was applied to the Google Play app recommendation. The hybrid design of Wide & Deep model shows that it is able to handle the cold-start problem [13]. Therefore, Wide & Deep model would also be an option model for this project.

3.4.3. Graph Neural Network

Another direction of the recommender system is using graph topology to model user-item interaction and then applying Graph Neural Network technique to perform recommendations. Knowledge Graph is a graph-structured that represents the relationship between different entities [14]. Knowledge Graph is a good way to represent the user-item relationship in this application, as the “item” is a helper request in this project and each helper request could relate to different entities, such as the user, request's category, location, etc. Knowledge Graph Attention Network (KGAT) is a model that is based on Knowledge Graph and attention mechanism to provide item recommendations [15]. So, in this project, KGAT would also be a possible model for the implementation of the recommender system.
4. Project Schedule and Milestones

<table>
<thead>
<tr>
<th>Time Periods</th>
<th>Tasks</th>
<th>Status</th>
</tr>
</thead>
</table>
| 2022 Sep         | ● Detailed project plan  
                  ● Project web page  
                  ● Project background research | Complete |
| 2022 Oct         | ● UI/UX design  
                  ● Database design  
                  ● Recommender system design | In-Progress |
| 2022 Nov - Dec   | ● Basic implementation of frontend  
                  ● Basic implementation of backend  
                  ● Basic implementation of Recommender system | Pending |
| 2023 Jan         | ● First presentation  
                  ● Detailed interim report | Pending |
| 2023 Feb - Mar   | ● Performance improvement  
                  ● Testing | Pending |
| 2023 Apr         | ● Final presentation  
                  ● Finalized tested implementation  
                  ● Final report | Pending |
| 2023 May         | ● Project exhibition | Pending |

Table 1: Project Schedule
5. Reference


