Mystery Shopper Application

by

LEUNG Hiu Lam

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science (Honours)
in Computer Science

Hong Kong Baptist University

April, 2020

Declaration

I hereby declare that all the work done in this Final Year Project is of my independent effort. I also certify that I have never submitted the idea and product of this Final Year Project for academic or employment credits.

_____

LEUNG Hiu Lam

Date: _____

Hong Kong Baptist University
Computer Science Department

We hereby recommend that the Final Year Project submitted by LEUNG Hiu Lam entitled "Mystery Shopper Application" be accepted in partial fulfillment of the requirements for the degree of Bachelor of Science (Honours) in Computer Science.

Dr. CHOY, Martin Man Ting	Prof. LEUNG, Yiu Wing
Supervisor	Observer
Date: _____	Date: _____

Abstract

Mystery shopping is a useful method commonly adopted by companies and organizations to improve their business strategies. It helps to analyze their companies from a customer perspective, which provides ideas that the companies may not discover. However, mystery shopping is manpower consuming and cost-ineffective. To solve this problem, my project is to build a mystery shopper application "WeTell", which applies homonym of "Retail" to demonstrate that it provides mystery shopping service to retail industry and tells the industry opinions in customer perspectives. The details of WeTell will be described in the following paragraphs.

Acknowledgements

I would like to express my great gratitude towards my supervisor, Dr. CHOY, Martin Man Ting, who had given me invaluable advice to this project. Besides, I would also like to thank Prof. LEUNG, Yiu Wing, who is the observer of this project, for his constructive comments during the development of the project.

Table of Contents

Abstract **i**

Acknowledgements **ii**

1 Introduction **1**

1.1 Background 1

1.2 Current situation 1

1.3 Existing system 3

1.4 Objectives 4

2 System Overview **6**

2.1 Overview 6

2.2 System Design 7

2.2.1 Use Case Diagram 7

2.2.2 Activity Diagram 14

2.2.2.1 CreateLogin and Signup 14

2.2.2.2 Edit Profile 15

2.2.2.3 Get Notification 16

2.2.2.4 Reserve Task (Inspector) 17

2.2.2.5 Complete Task (Inspector) 18

2.2.2.6 Redeem Points (Inspector) 19

2.2.2.7 Create Tasks (Enterprise) 20

2.2.2.8 Comment And Rate Result (Enterprise) . . 21

2.2.2.9 View Tasks Result (Enterprise) 22

2.2.2.10 Logout 23

2.2.3 State Diagram 24

2.2.4 Entity Relationship Diagram 25

3 System Implementation **26**
3.1 System Architecture 26
3.1.1 Mobile application 26
3.1.2 Server 27
3.1.3 Database 27
3.2 Features 28
3.2.1 Login/ Sign up 28
3.2.2 View Profile 29
3.2.3 Notification 30
3.2.4 Create Task (Enterprise) 30
3.2.5 Search and Reserve Task (Inspector) 36
3.2.6 Complete Task (Inspector) 41
3.2.7 View Task Results (Enterprise) 45
3.3 Implementation of Maps Searching 47

4 Discussion and Conclusion **50**
4.1 Difficulties and Limitations 50
4.2 Future Development 50
4.2.1 Enhancement on Security Issues 50
4.2.2 Enhancement on User Experience (UX) 51
4.2.3 Enhancement on User Interface (UI) 51
4.3 Conclusion 51

Appendix 1: System Setup **53**
A1.1 Server (Sails.js) 53
A1.2 Flutter (Windows) 53
A1.2.1 Mininum requirements: 53
A1.2.2 Installation: 53
A1.3 Flutter (MacOS) 54
A1.3.1 Installation: 54

References **55**

Chapter 1

Introduction

1.1 Background

Mystery shopping is widely used as a diagnostic tool for service quality evaluations throughout the year. It applies an anonymous observation from a customer perspective, which includes product quality, service delivery procedures, and front-line staff performance. It can be used in all industries, from business-to-consumer(B2C) and business-to-business(B2B), but mostly it is used by B2C industries, such as retail, hotels, restaurants, and transportation. Enterprises can organize mystery shopping programs by themselves or purchase the service from mystery shopping providers.

1.2 Current situation

To complete a mystery shopping program, it involves three types of characters, which are mystery shopper company, mystery shopper, and enterprise. Mystery shopper companies employ and train mystery shoppers to provide mystery shopper service when enterprises purchase mystery shopping services.

The current workflow for organizing a mystery shopping program involves:

- Recruitment and training
  Mystery shopper company first recruits mystery shoppers. When mystery shopper service is purchased, the company will organize the training sessions for shoppers if the shoppers have not experienced any related job yet.

- Discussion with enterprise
  The agent of the mystery shopper company will discuss details of the service with the enterprise, including price, evaluation questions, and requirements. The price of the service will be introduced in packages. For example, the enterprise may purchase a one-year-package, which includes two visits per month.

- Arrangements of manpower
  Once the agent receives a job from a client, he/she will draft details of the task, including date, requirements, and salary. Then he/she will send the task information to shoppers, who satisfy the client's requirement and make an arrangement through email or messages.

- Evaluation
  Mystery shopper observes the environment and staff behaviour during an evaluation. Audio and photos will be recorded if it is required. Shoppers will fill in an evaluation form in excel and send it to the mystery shopper company.

- Data processing and analysis
  After receiving an evaluation from shoppers, clerical staff will be assigned to take charge of the data including revising audio scripts, verifying the data, and preparing a report for the client. The report may include shoppers' experience, related monthly statistics, and ad-hoc enhancement suggestions.

- Reporting to enterprise
  When the report is ready, the clerical staff will send it to the client. The staff will explain the report details to the client in person if it is the first report. In the case of inquiry from the client, the staff would answer by phone or in person.

The process of a mystery shopping program is considered as time, money, manpower consuming, and cost-ineffective. The agent has to search for targeted shoppers, which fit the task requirements and draft task information and details manually. Moreover, to process and analyze the data, the clerical staff members have to verify the data and generate the statistics by hand. The manual process increases the workload of the staff when it is not effective, especially in reviewing the data and analyzing the statistics.

On the other hand, the current evaluation form in excel is not user friendly and inconvenient for shoppers. Shoppers are required to match questions and answers in excel, which increases the chance of making of mistakes. Also, the data are sent to the staff by email. With a large number of emails, there is a possibility to miss the data. The manual process raises the inaccuracy of the data.

Due to the significant costs of a mystery shopping program, the price of the service is not affordable by the small and medium-sized enterprises. Also, enterprises expect to have a short-term or one-time mystery shopping program instead of having it in a package.

WeTell is an alternative way to provide cost-effective services for enterprises. It is a peer-to-peer platform between enterprises and mystery shoppers. Mystery shoppers are allowed to accept tasks when enterprises post the tasks on the app. When shoppers finish a task, the statistics and reports will be generated automatically. Not only does the app reduce the costs of the mystery shopping program, but it also increases the accuracy of the data.

1.3 Existing system

MissionPick is an existing mystery shopper application in Hong Kong, which allows users to earn money by finishing mystery shopping tasks. To initiate a task, enterprises can fill in the form on the mystery shopper company's website or send an email to the company. The staff will contact them and discuss the details of mystery shopping programs with them. They will get real-time results for the task.

Users are allowed to register as a mystery shopper. To complete the registration, it requires approval from the admin team, which takes 2 to 3 days. Before shoppers start their first task, they have to finish training tasks, and this simulates the process of a mystery shopping task. Once they submit a task, the task will be reviewed by the quality control team. Shoppers will receive payment in 7 days if the evaluation is accepted. Photos and insights of the result will be updated and sent to the enterprises.

MissionPick provides a convenient platform for mystery shoppers. However, it contains a lot of manual operations, such as approval of user registration, reviewing the quality of evaluations, and creating tasks. It is manpower consuming and cost-ineffective. Moreover, enterprises have to discuss with the mystery shopper company in-person to create a task. Both enterprises and mystery shopper companies have to spend a largen amount of time to schedule and arrange the task.

Moreover, MissionPick only allows users to search for tasks by location, which is not user-friendly. It is expected to provide filters for users to choose their targeted jobs. Also, to get the detail information of the task, users have to click on the task on the map. It is expected to preview the task before the user input.

1.4 Objectives

The purpose of this project is to develop a mystery shopper application that provides a peer-to-peer platform for enterprises and inspectors on mystery shopping services. The main objectives of the project are:

1. to reduce the manual operations
   WeTell is designed to automate the manual operations of mystery shopping programs. The system authenticates the data through algorithms so that the process of registration, task verification, and data analysis can be smoothed and unobstructed. For example, the system will track user location to ensure shoppers are in the shop area.

2. to provide a user-friendly interface
   The Inspector platform of WeTell shows important information about the tasks as a list, such as a shop type, pay, shop's name, and address. Inspectors are allowed to filter tasks by shop type and pay. They can also reserve or start the task instantly.

3. to enhance the accuracy of the data
   When shoppers submit their evaluations, the system will process the data. Enterprises will receive the evaluation results instantly. When the task is completed, the system will generate a report with statistics and insights into the enterprises.

Chapter 2

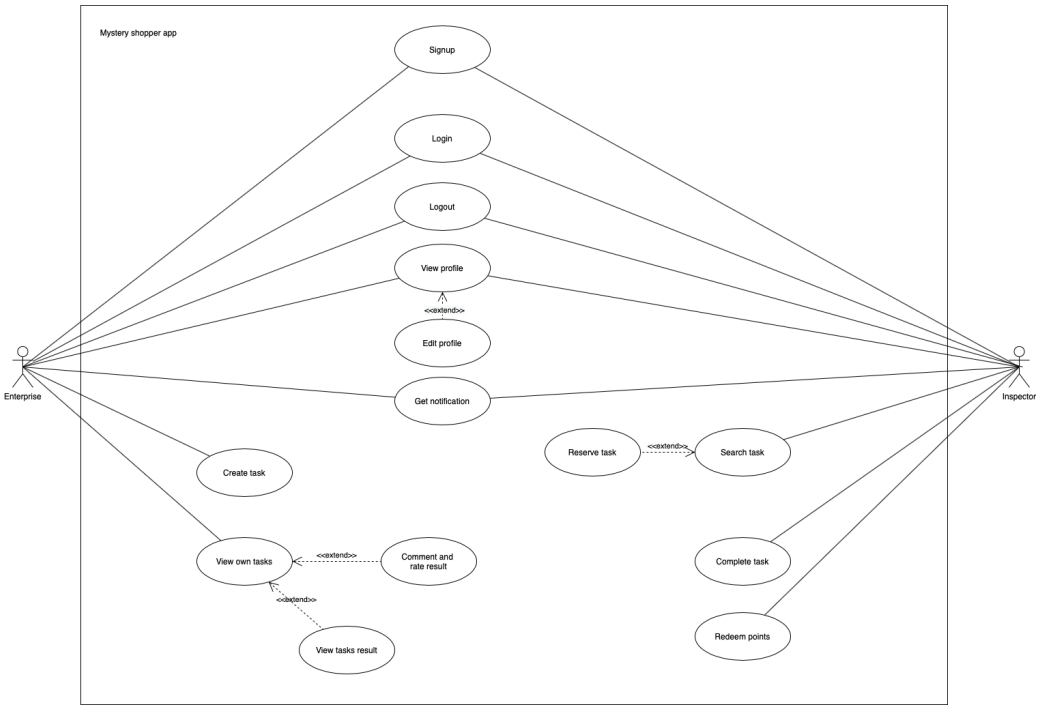
System Overview

2.1 Overview

WeTell provides a peer-to-peer platform between enterprises and inspectors. There are two different systems for inspectors and enterprises, since they have particular user behaviors, which will be described below. Users are allowed to register as an enterprise or an inspector. After registration as an enterprise, the enterprise can create a task. Inspectors can reserve and complete tasks, the evaluation results will then store in tasks record. After an inspector complete a task, task points will be added to his/her balance and a notification will be sent to the enterprise. The enterprise can also view the evaluation records even the whole task is not completed yet.

2.2 System Design

2.2.1 Use Case Diagram



Use Case [number 1]	Sign
Goal in Context	This use case allows user to register as an enterprise or an inspector
Preconditions	- Inspector has attained the age of 18 - Enterprise or inspector has valid email account
Success End Condition	A notification is sent to user and enterprise/ inspector login page is redirected
Failed End Condition	Sign up failure message is shown
Actor	Enterprise, Inspector

Use Case [number 1]	Sign
Description	1. User clicks “Enterprise” or “Inspector” button 2. User clicks “Sign up for new account” button 3. User inputs account information, e.g. email, password, name 4. User clicks “Submit” button
Extensions or Variations	The email has been taken already The length of password is less than 6 charaters A required field is empty

Use Case [number 2]	Login
Goal in Context	The use case allows user to login as an enterpirse or inspector
Preconditions	User has an existing account
Success End Condition	Enterprise or inspector homepage is redirected
Failed End Condition	A login failure message is shown
Actor	Enterprise, inspector
Description	1. User clicks “Enterprise” or “Inspector” button 2. User inputs email and password 3. User clicks “Submit” button
Extensions or Variations	Inputted email does not exist in user records Inputted password does not matched in user record

Use Case [number 3]	Logout
Goal in Context	The use case allows user to logout current account
Preconditions	User has already logged in
Success End Condition	An app start page is redirected
Failed End Condition	-

Use Case [number 3]	Logout
Actor	Enterprise, inspector
Description	1. User clicks “My Profile” tab 2. User clicks “Logout” button 3. An alert message is shown 4. User clicks “Confirm” button
Extensions or Variations	-

Use Case [number 4]	View and edit profile
Goal in Context	The use case allows user to view and edit his/ her information
Preconditions	User has already logged in
Success End Condition	A successful updated message is shown
Failed End Condition	-
Actor	Enterprise/ Inspector
Description	1. User clicks “My Profile” tab 2. User clicks “Edit my profile” 3. User inputs new information 4. User clicks “Confirm” button
Extensions or Variations	-

Use Case [number 5]	Get notifications
Goal in Context	The use case allows user to get his/ her notifications
Preconditions	User has already logged in
Success End Condition	A notification page is shown
Failed End Condition	-
Actor	Enterprise/ Inspector
Description	1. User clicks “Notifications”
Extensions or Variations	-

Use Case [number 6]	Search task and reserve task
Goal in Context	The use case allows user to search and reserve a task
Preconditions	User has logged in as an inspector User has less than 6 current tasks User has score of 3 in ratings User type fits available task
Success End Condition	A reserved successful message is shown
Failed End Condition	No available task for user
Actor	Inspector
Description	1. User clicks “Search Task” tab 2. Users select a location 3. User sorts task by type 4. User clicks a task 5. User clicks “Reserve” button
Extensions or Variations	The task is out of quota

Use Case [number 7]	Complete task
Goal in Context	The use case allows user to complete a task
Preconditions	User has logged in as an inspector User has reserved at least one task User’s current location is in shop area User can take photo by his/ her device
Success End Condition	A complete message is shown
Failed End Condition	An error message is shown
Actor	Inspector
Description	1. User clicks “My Profile” tab 2. User clicks “My tasks” button 3. User clicks a task 4. User clicks “Start” button 5. User submits answers 6. User edit answers on preview section (optional)

Use Case [number 7]	Complete task
Extensions or Variations	If user's current location is out of shop area for 2 minutes Answers are not valid

Use Case [number 8]	Redeem points
Goal in Context	The use case allows user to redeem points to cash
Preconditions	User logged in as an inspector
Success End Condition	Successful message of payment to user's paypal account is shown
Failed End Condition	A payment failure message is shown
Actor	Inspector
Description	1. User clicks "My Profile" tab 2. User clicks "Redeem Points" button 3. User selects a price 4. User clicks "Confirm" button
Extensions or Variations	User does not have enough points to redeem the price User's email and his/ her account are not matched Paypal has no response

Use Case [number 9]	Create task
Goal in Context	The use case allows user to create a task, then the task can be searched and completed by inspectors
Preconditions	User has logged in as an enterprise
Success End Condition	A successful creation page is shown "End date" of the task must be at least 29 days after "Start date" of the task
Failed End Condition	An error message is shown

Use Case [number 9]	Create task
Actor	Enterprise
Description	1. User clicks “New Task” tab 2. User inputs task information 3. User clicks “Next” button 4. User selects a category 5. User selects a template 6. User edits questions from template (optional) 7. User clicks “Next” button 8. User clicks “Next” button 9. User clicks “Confirm” button 10. User proceeds payment througth Apple Pay/ Android Pay
Extensions or Variations	Payment fails

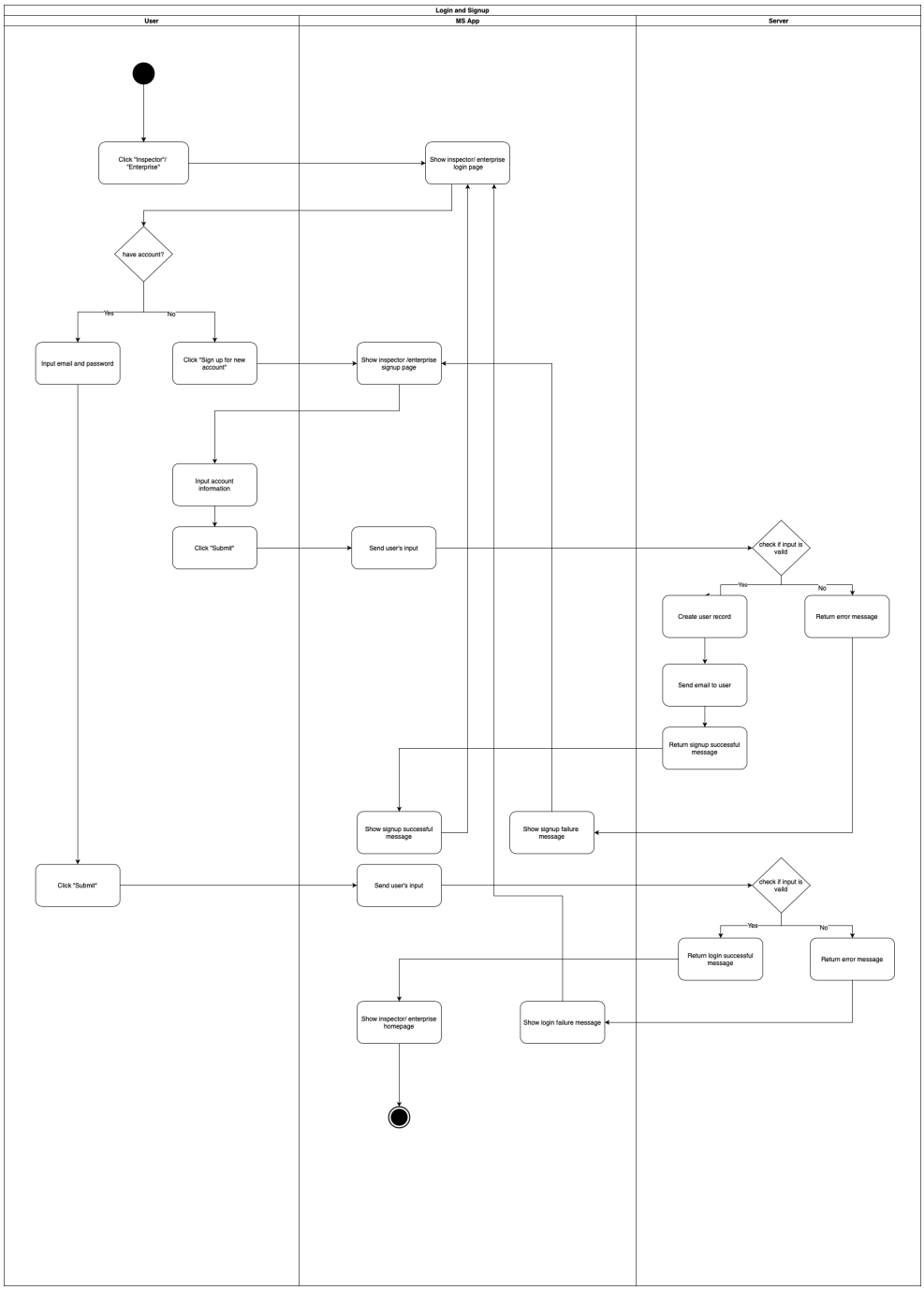
Use Case [number 10]	Comment and Rate results
Goal in Context	The use case allows user to comment and rate the evaluation results which are inputted by inspectors
Preconditions	User logged in as an enterprise User’s task is evaluated by at least one inspector User has not commented and rated the evaluation
Success End Condition	A updated message is shown
Failed End Condition	-
Actor	Enterprise

Use Case [number 10]	Comment and Rate results
Description	1. User clicks “My Tasks” tab 2. User clicks a task 3. User clicks “Evaluation Details” button 4. User clicks an evaluation record 5. User clicks “Next” button 6. User inputs comments and ratings 7. User clicks “Submit” button
Extensions or Variations	-

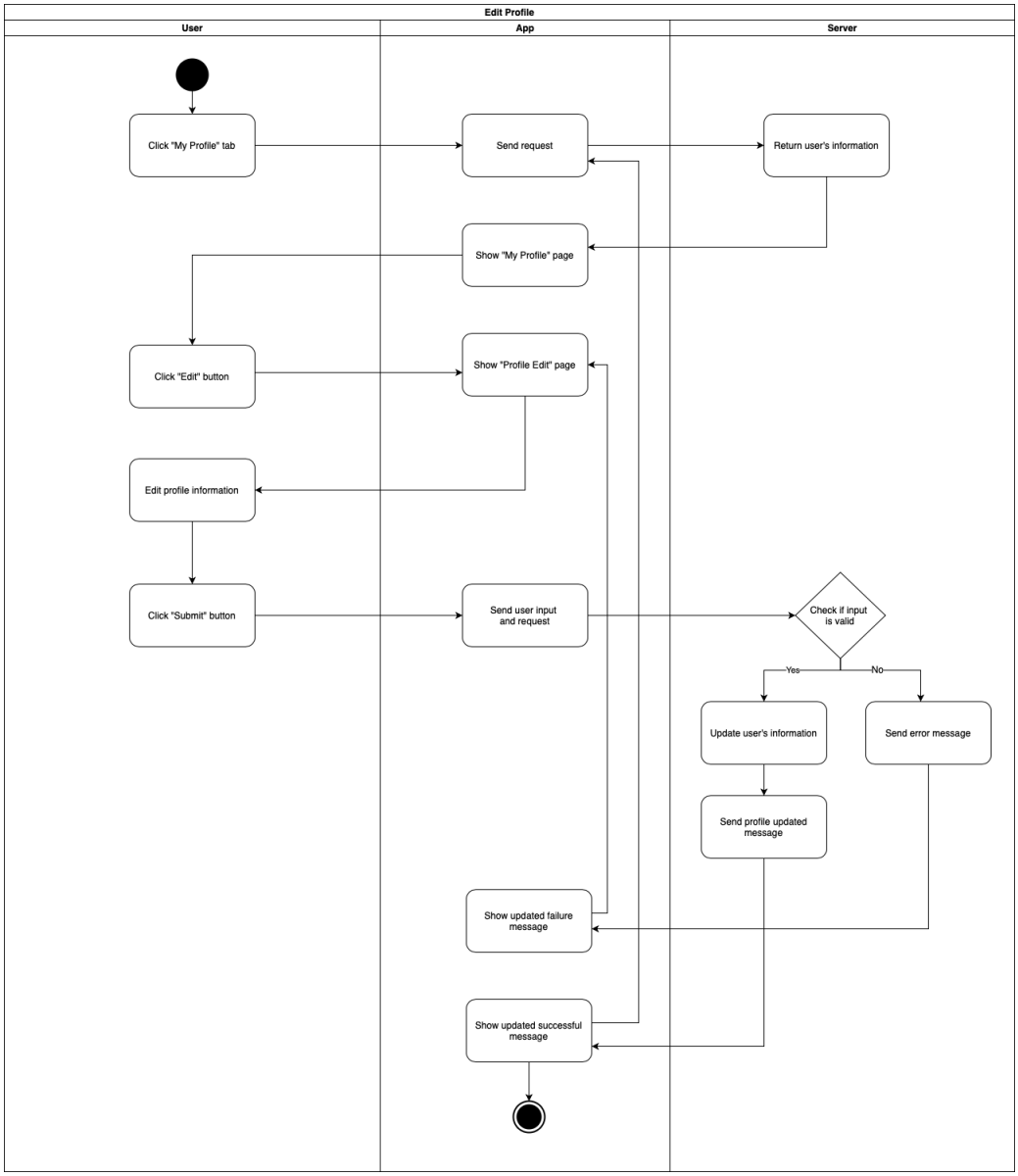
Use Case [number 11]	View task results
Goal in Context	The use case allows user to view on results on his/ her task, which includes statistics, evaluation records, and average results
Preconditions	User logged in as an enterprise User’s task is evaluated by at least one inspector
Success End Condition	Detail results are shown
Failed End Condition	-
Actor	Enterprise
Description	1. User clicks “My Tasks” tab 2. User selects a task 3. User clicks “More results” button 4. Users selects a question category 5. User selects a question
Extensions or Variations	-

2.2.2 Activity Diagram

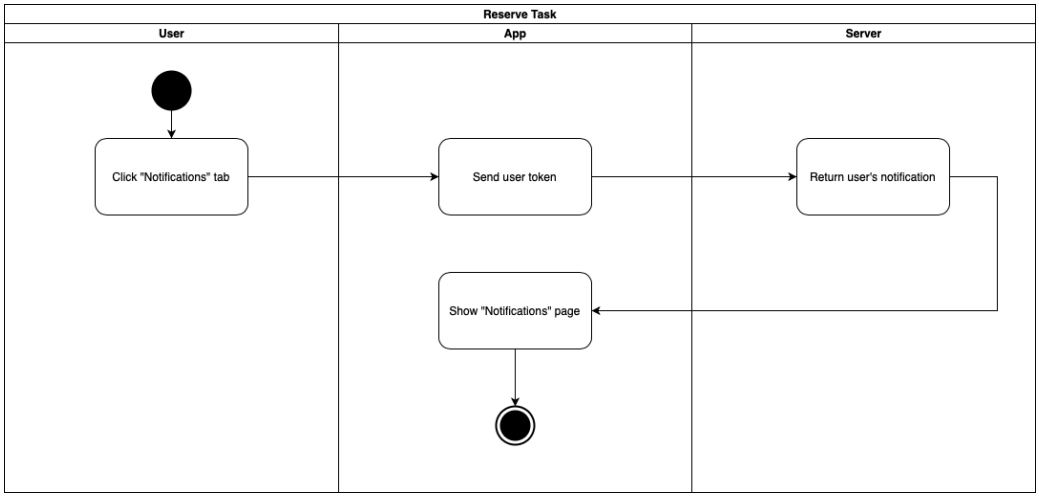
2.2.2.1 CreateLogin and Signup



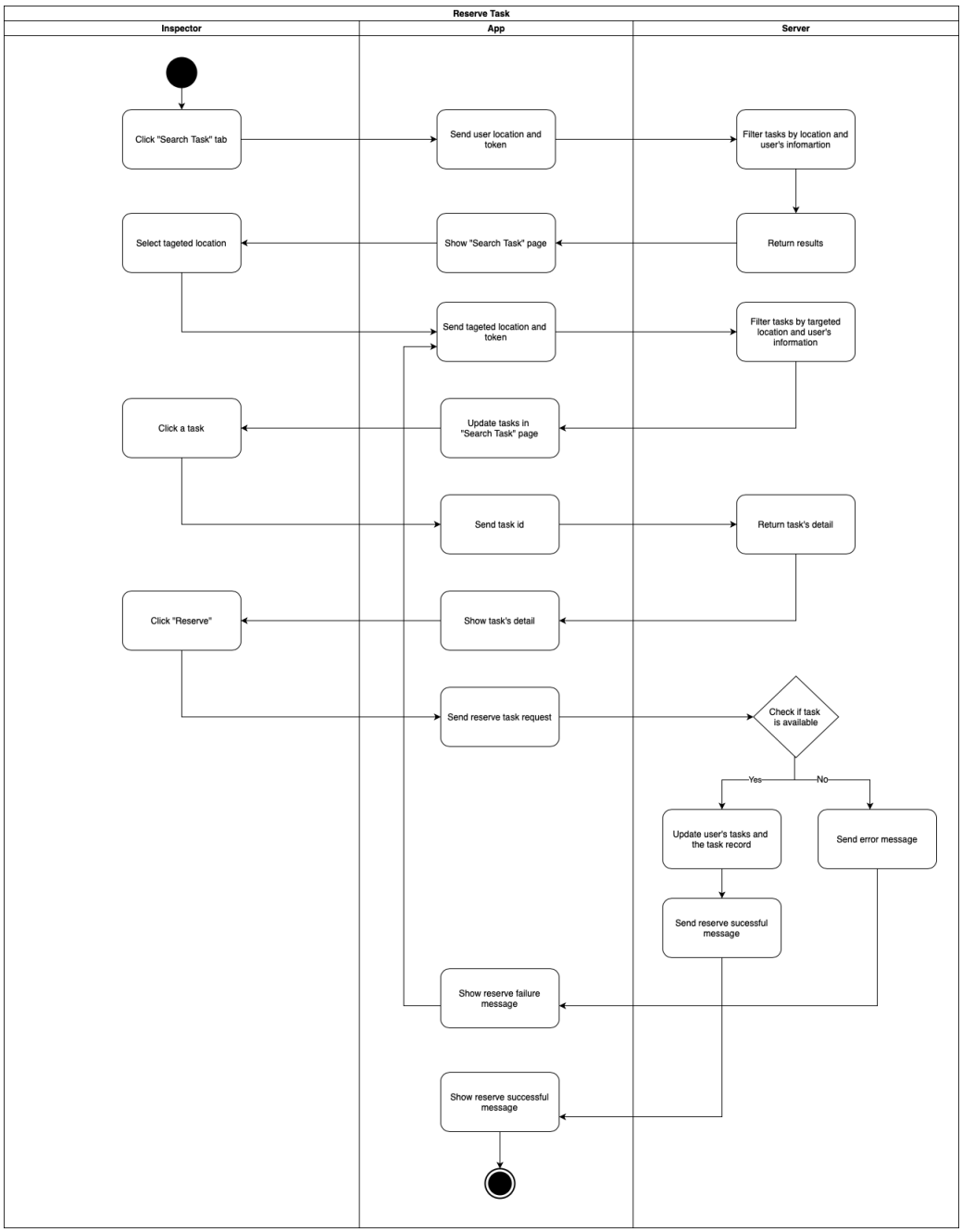
2.2.2.2 Edit Profile



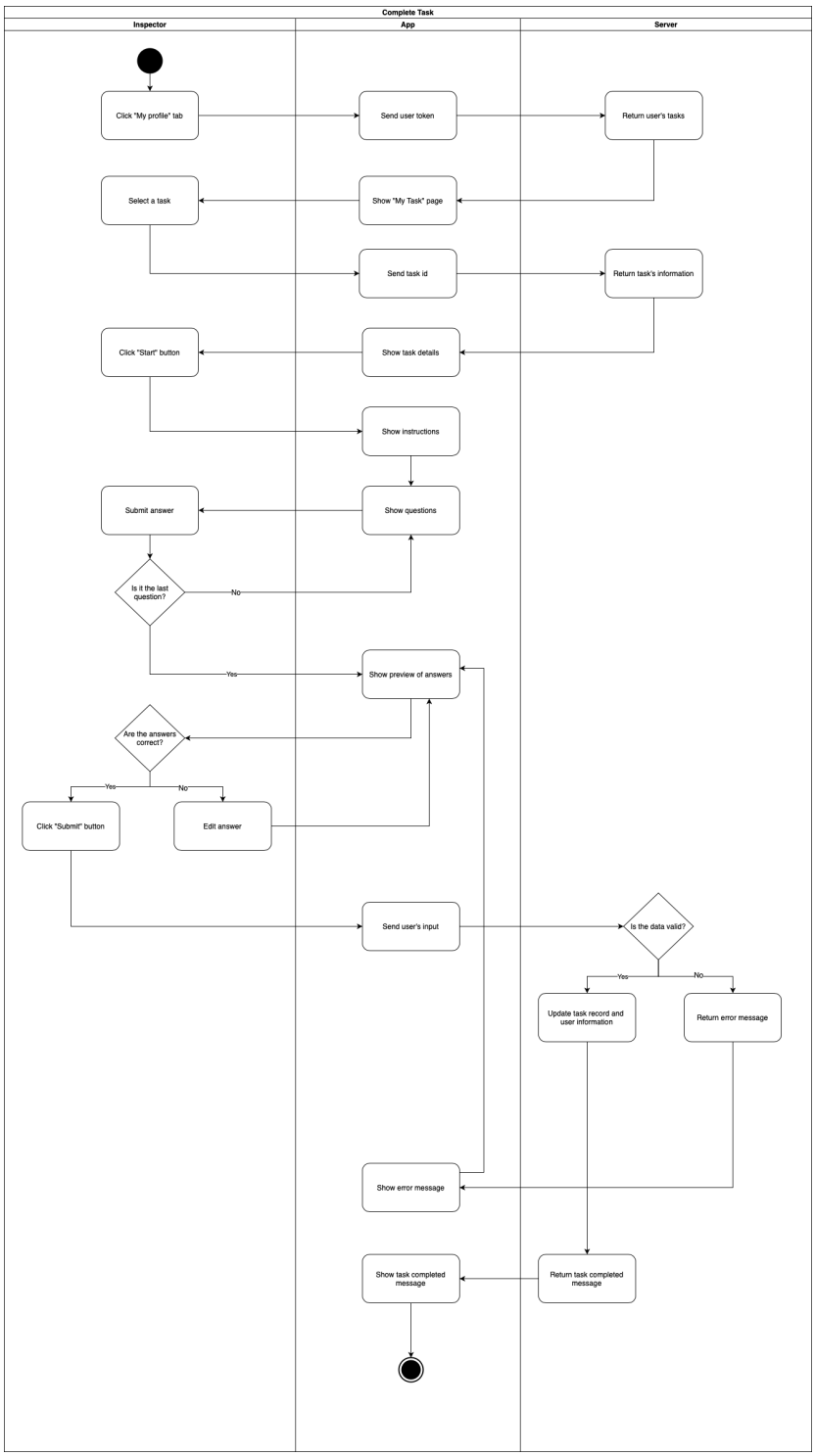
2.2.2.3 Get Notification



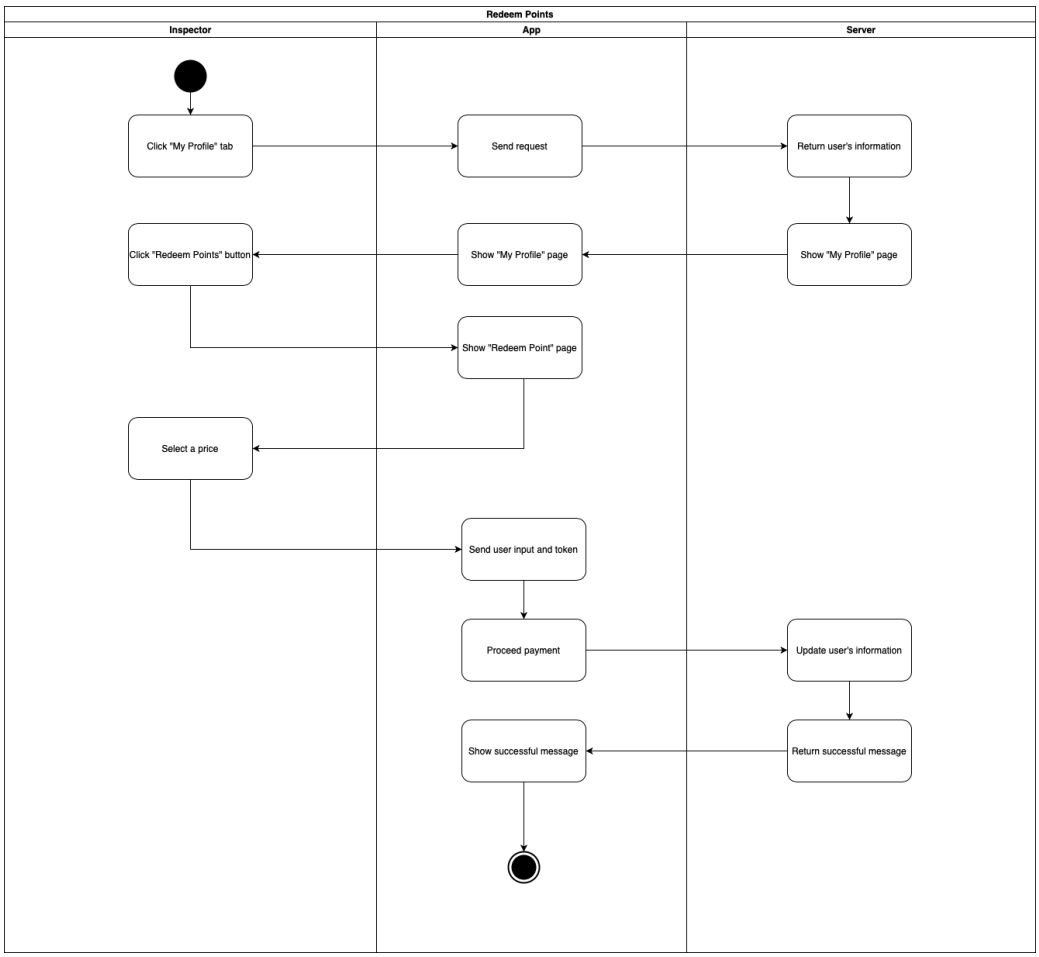
2.2.2.4 Reserve Task (Inspector)



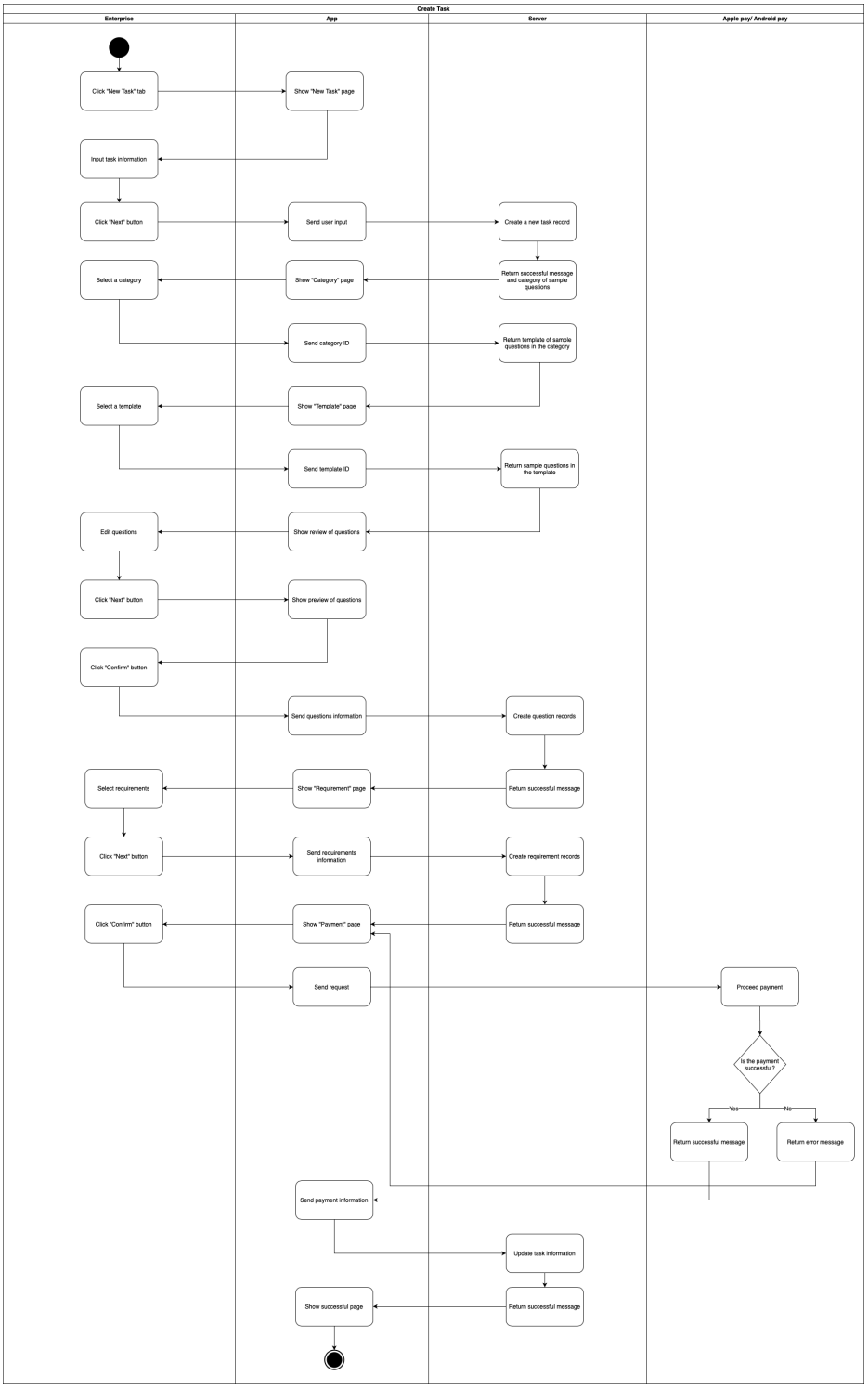
2.2.2.5 Complete Task (Inspector)



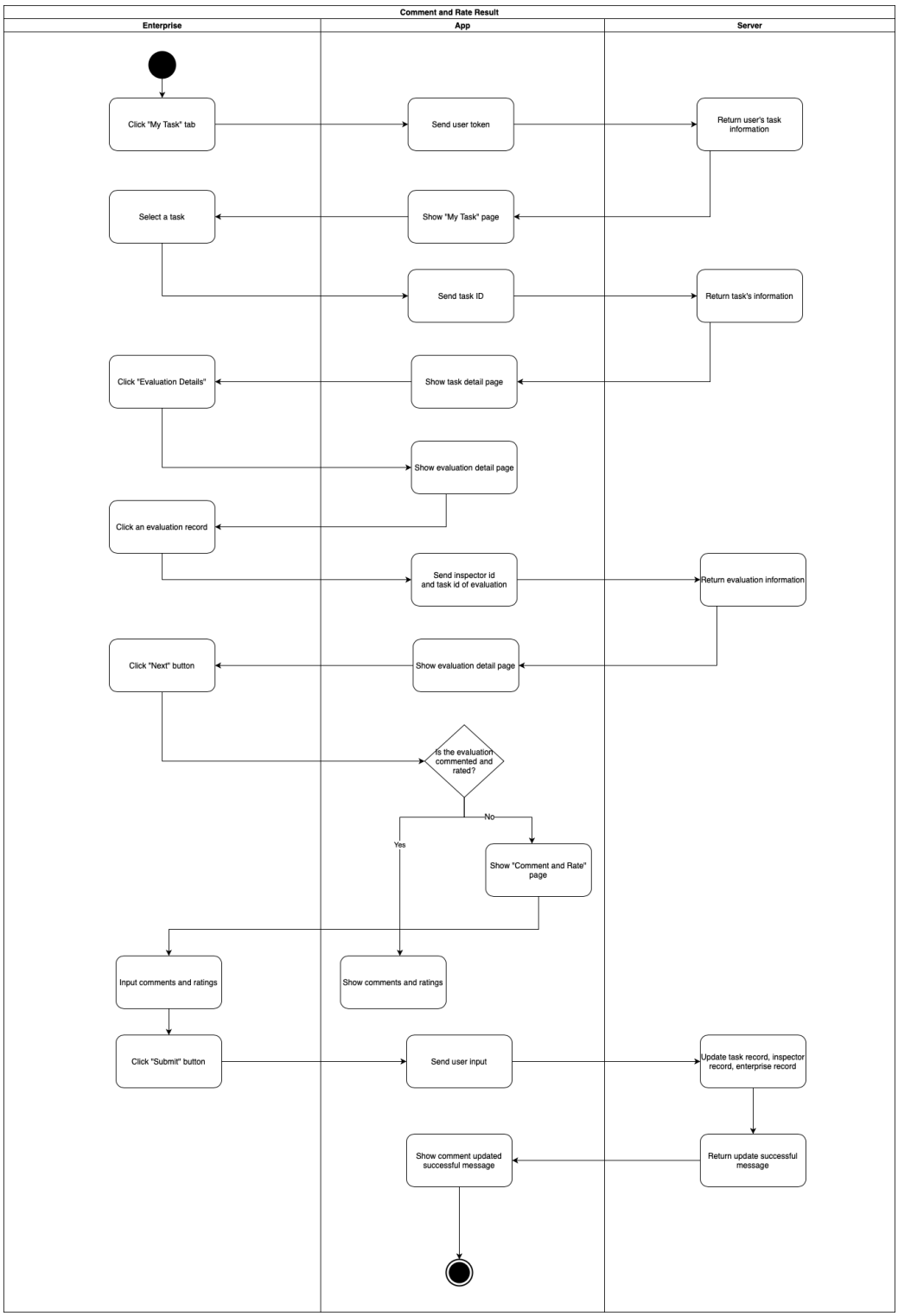
2.2.2.6 Redeem Points (Inspector)



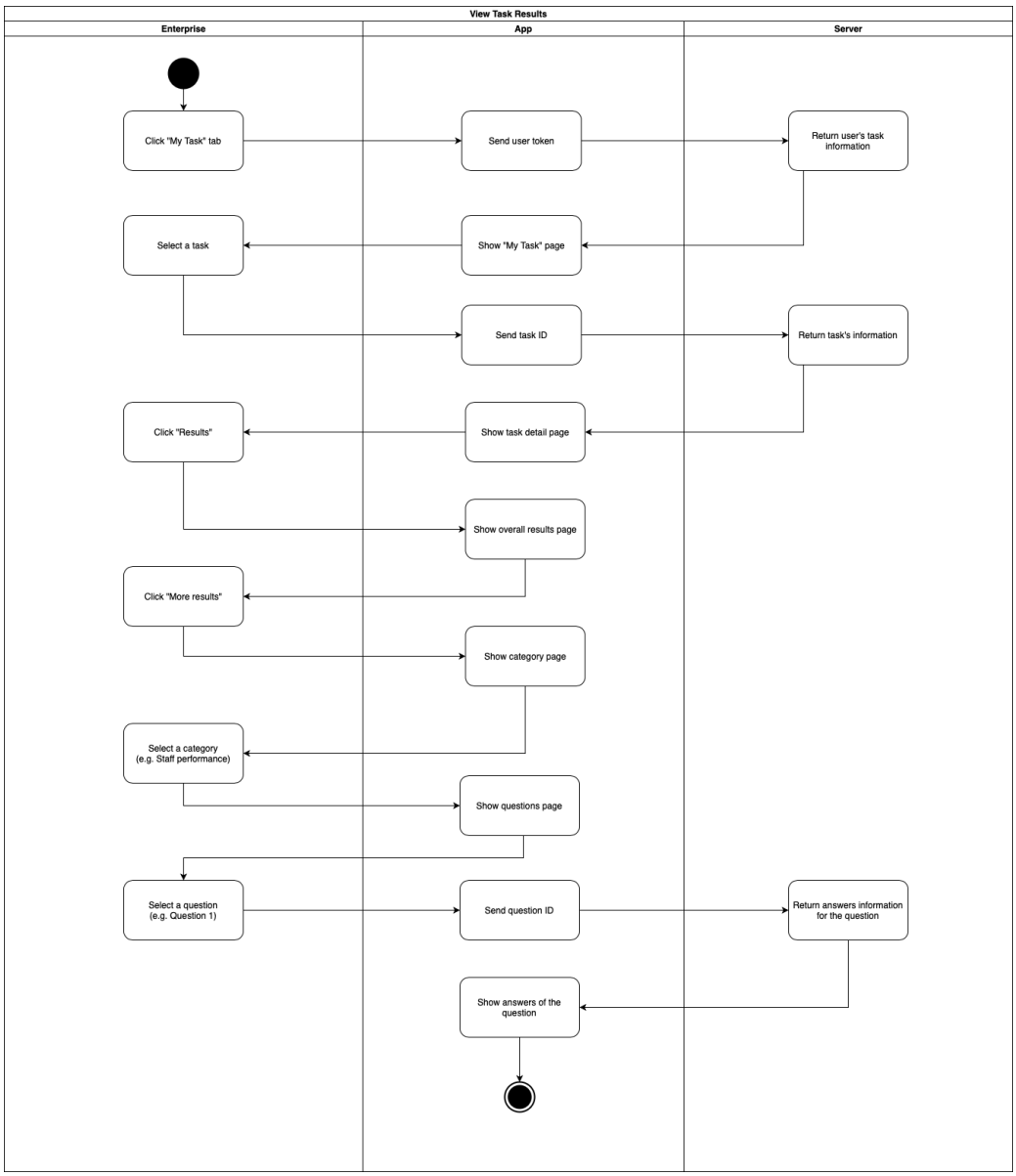
2.2.2.7 Create Tasks (Enterprise)



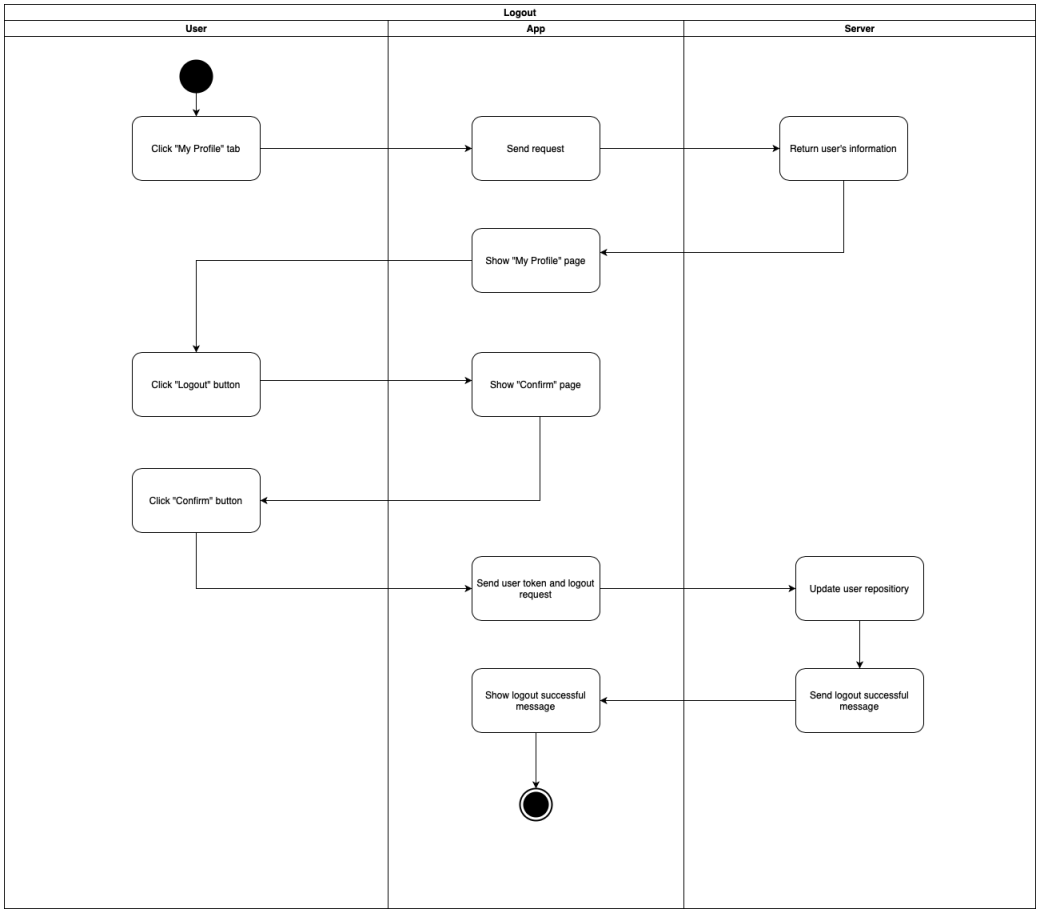
2.2.2.8 Comment And Rate Result (Enterprise)



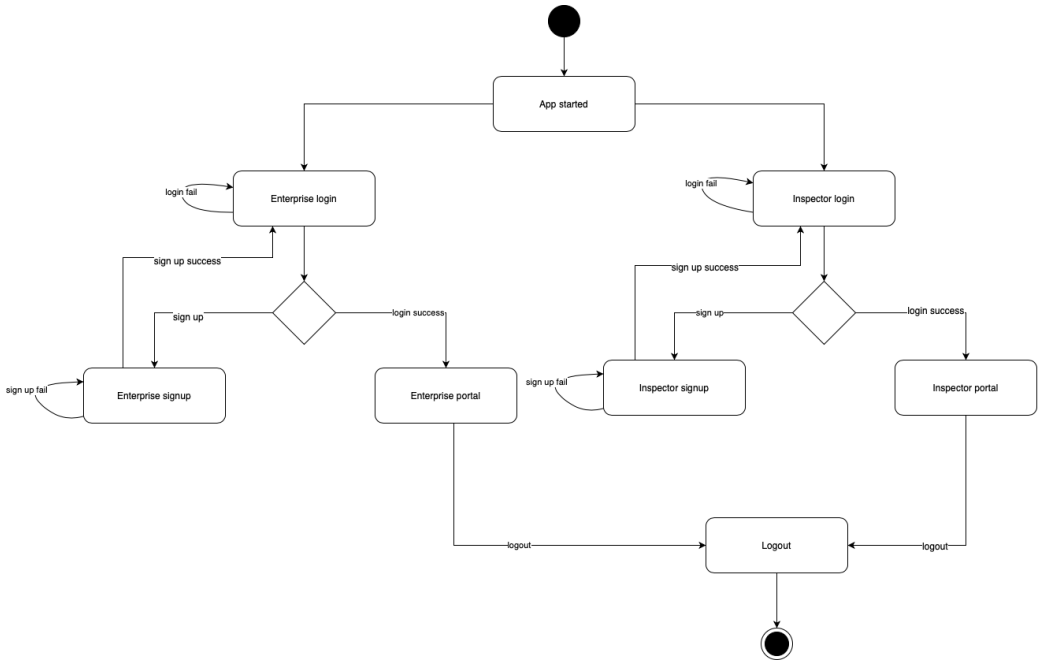
2.2.2.9 View Tasks Result (Enterprise)



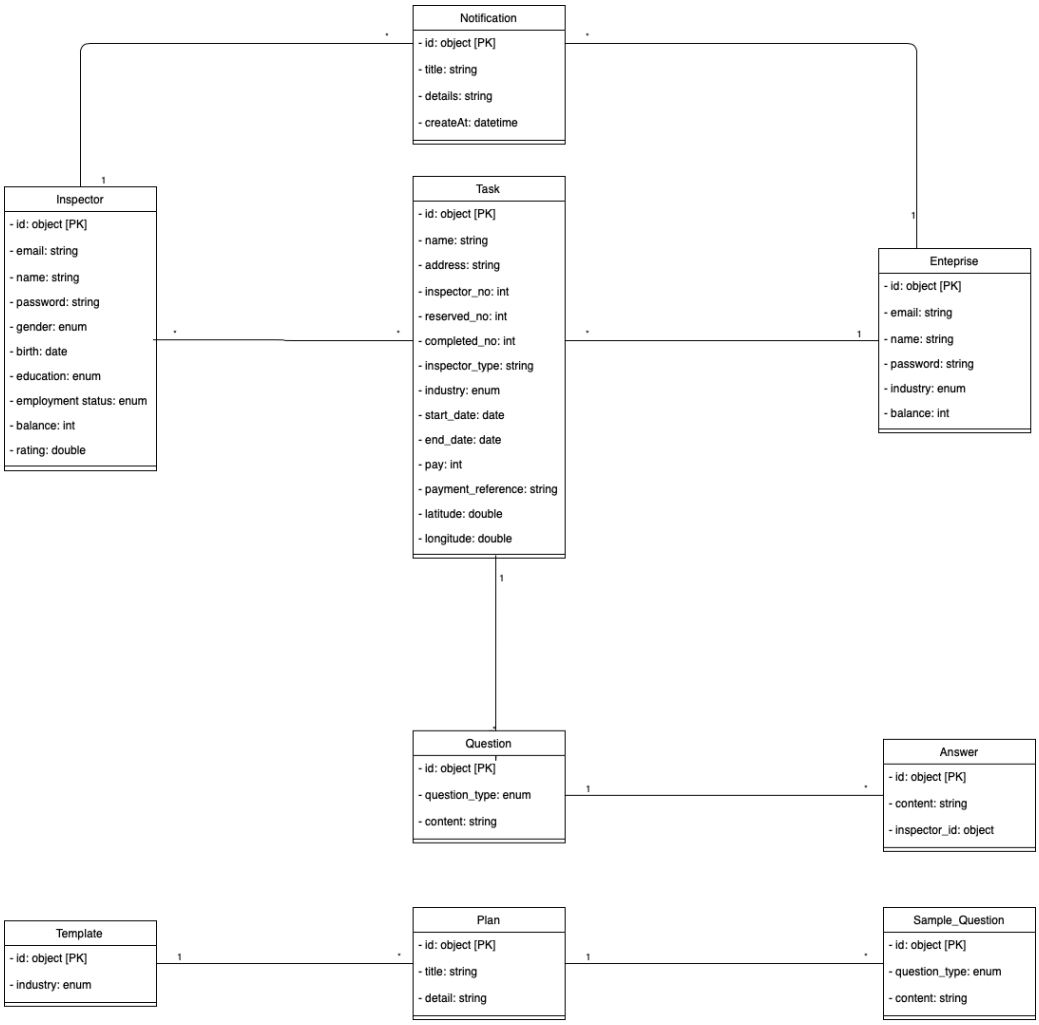
2.2.2.10 Logout



2.2.3 State Diagram



2.2.4 Entity Relationship Diagram



Chapter 3

System Implementation

3.1 System Architecture

WeTell is formed by three components, which are the application for users to access the system, server for handling requests and database for storing data records.

3.1.1 Mobile application

Flutter is an open-source and cross-platform mobile SDK created by Google. It can be used to build both iOS and Android apps from the same code base[1]. Flutter apps are built using dart, which is an object-oriented programming language, therefore it is dynamic and extensible. Also, Flutter eliminates JavaScript bridge and is complied to native machine code, so that the performance Flutter apps . mdeIt can be used to build both iOS and

Android apps from the same code base[1]. Flutter apps are built using dart, which is an object-oriented programming language, therefore it is dynamic and extensible.

3.1.2 Server

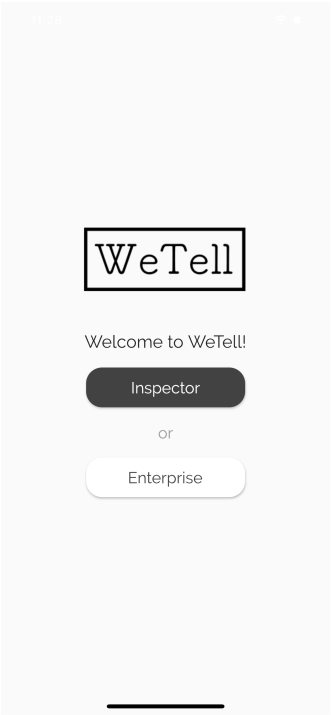
Sails.js is the most trendy MVC framework for Node.js, which helps build custom and enterprise-grade apps. It is built on Node.js, uses Express for handling HTTP requests[2]. Sails bundles a powerful Object-Relational-Mapper(ORM), waterline, which provides a simple data access layer to any database[2].

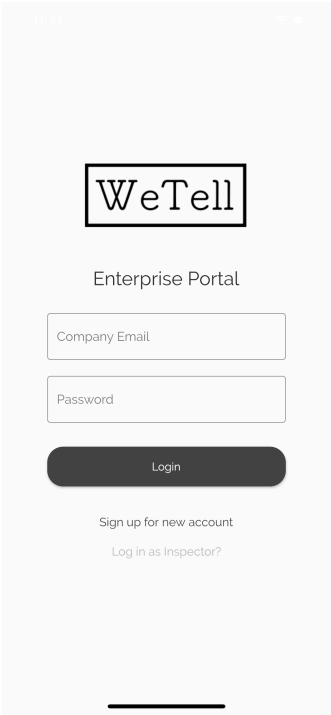
3.1.3 Database

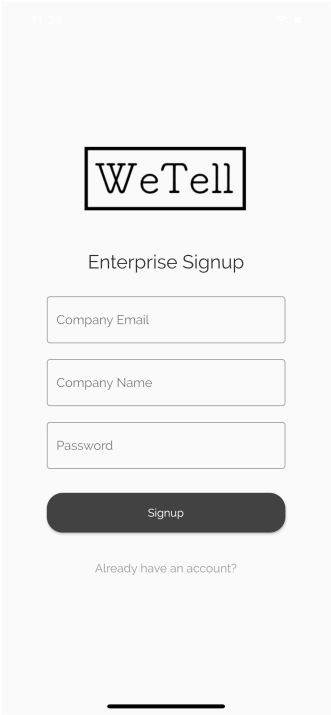
MongoDB is a document database with the scalability and flexibility, which is a non-relational database engine. MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time[3].

3.2 Features

3.2.1 Login/ Sign up

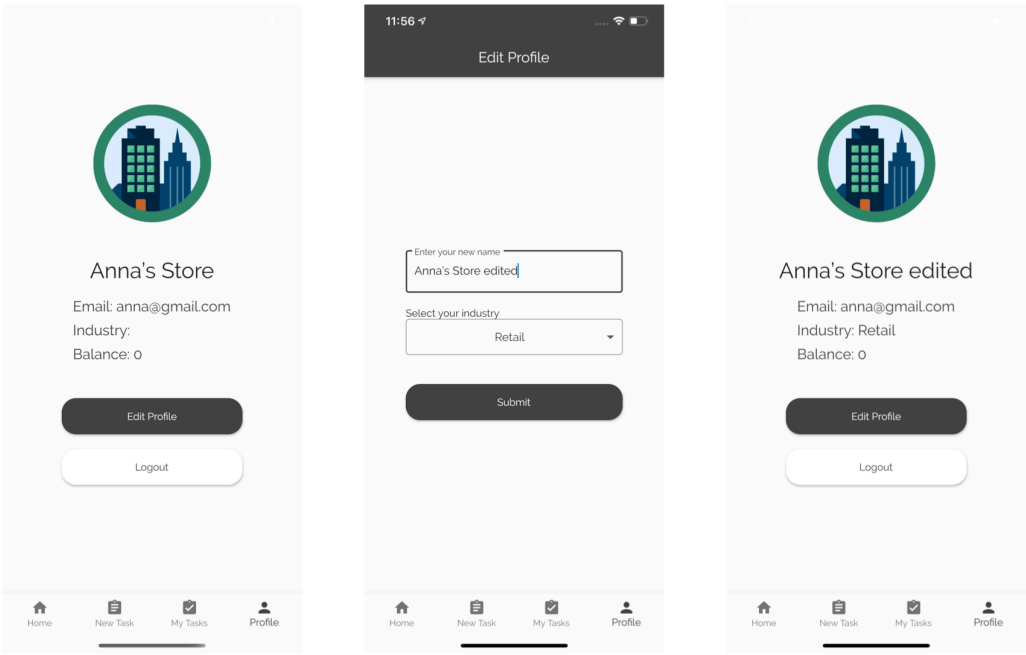






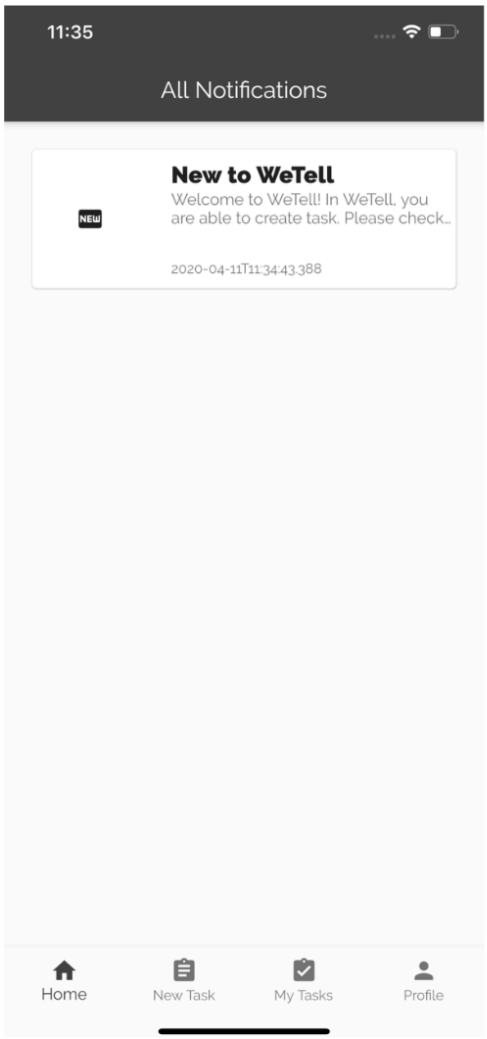
When WeTell is started, a welcome page is shown. Users are allowed to select their roles, either an enterprise or an inspector. When a user clicks the "Enterprise" button, WeTell navigates to the login page of the enterprise portal. The user can input his/her registered email and password for login. If the user has not created an account yet, he/she can tap "Sign up for new account" to move to the "Enterprise Signup" page. To create a new account, the email input must not be taken. Also, the password input must have more than 5 digits. When the account is created, a notification with a welcoming message sends to the enterprise, when the "Thank you for signing up" page is shown. The user interface of login and signup pages in the inspector portal are similar.

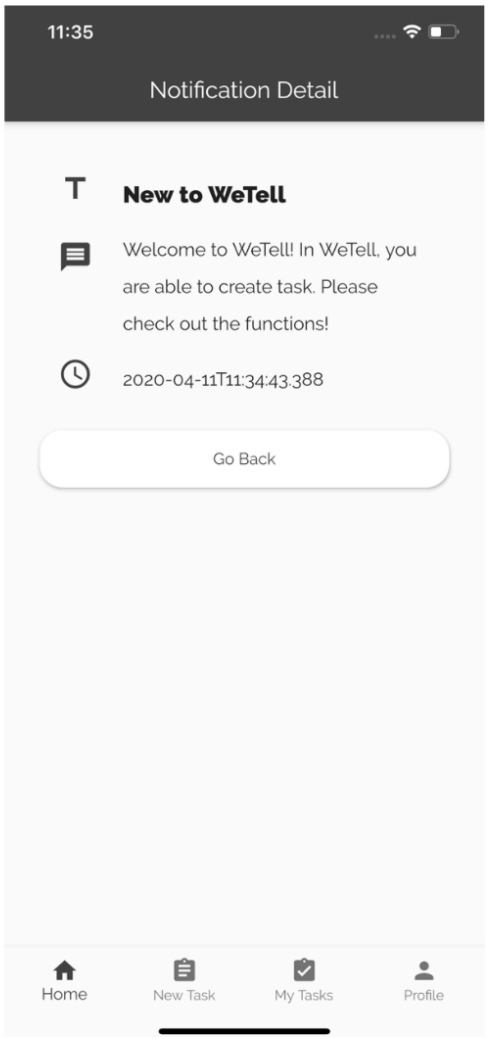
3.2.2 View Profile



By tapping the last tab of the bottom bar, a profile page is presented, which demonstrates the user information. For the enterprise portal, the name, email, industry, and balance are shown. The enterprise can tap the "Edit Profile" button to edit his/her information. When the submit button is tapped, the profile page with updated information is navigated.

3.2.3 Notification

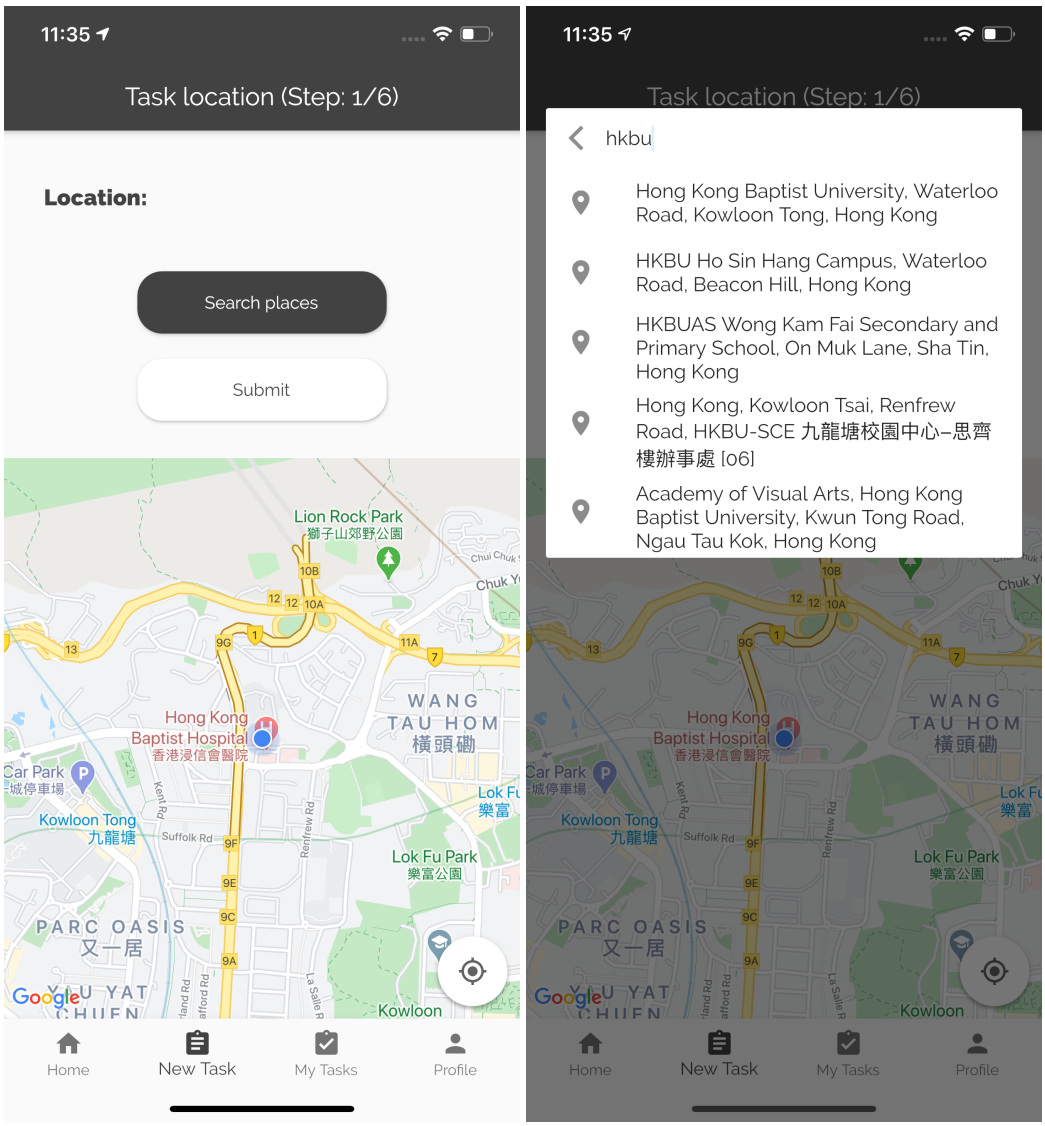




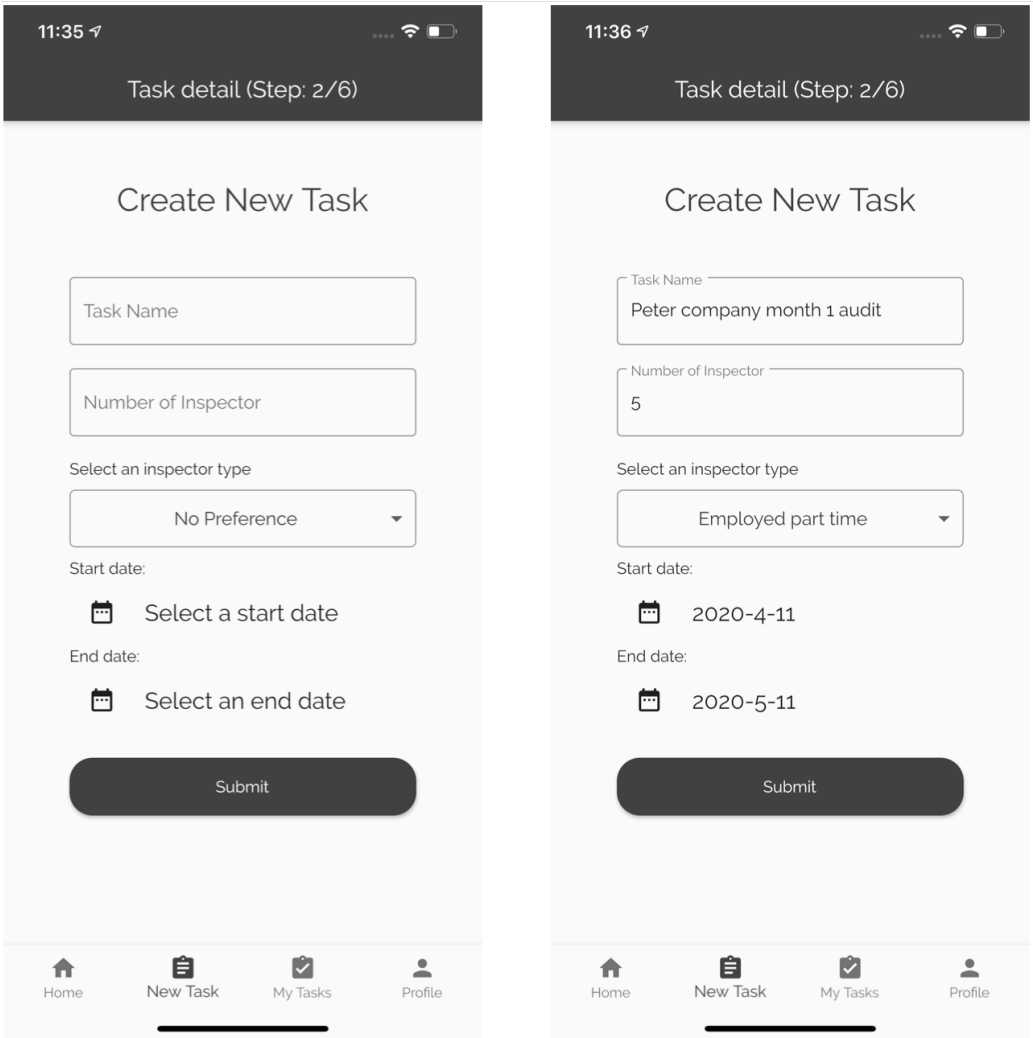
If an enterprise or an inspector login successfully, a notification page is shown, which is the homepage of both portals. Users can also tap the first tab of the bottom bar to enter this page. The user interfaces of the two portals are similar. When the user clicks a notification on the notification list, the details of the notification are introduced.

3.2.4 Create Task (Enterprise)

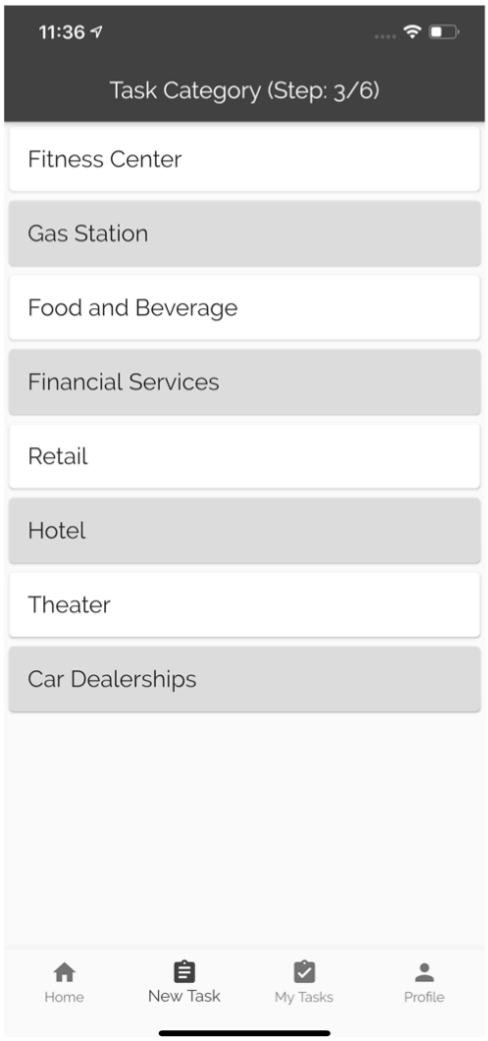
There are 6 steps in total for task creation.

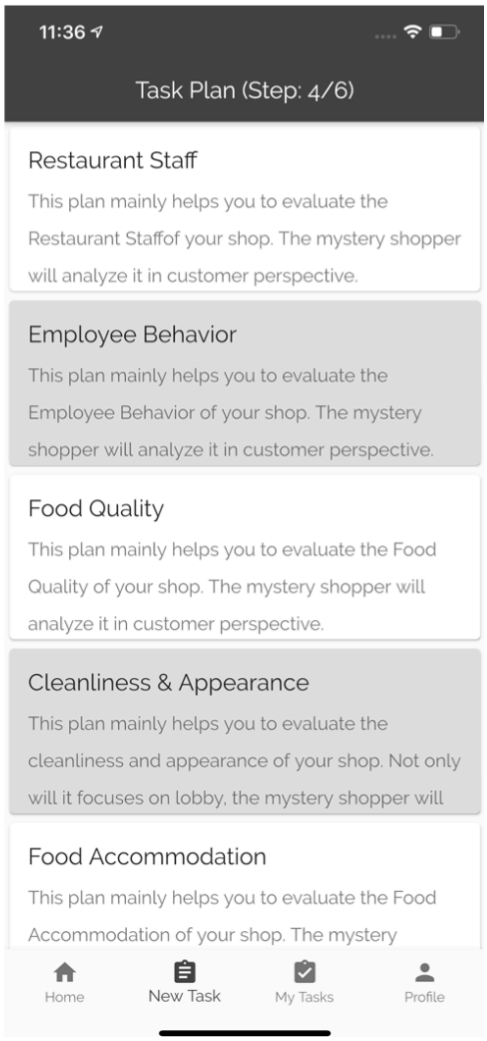


When the enterprise tap "New Task" tab on the bottom bar, the "Address" page is shown. Using Google Maps plugins, WeTell applies to autocomplete function for enterprises to search their target address. When enterprise types some characters, it provides some choices for the user to select. In the "Address" page, if enterprises have not inputted the address yet, when they click the "Submit" button, a snack bar with an error message is presented. After submitting the address of tasks, enterprises can proceed to the second stage.

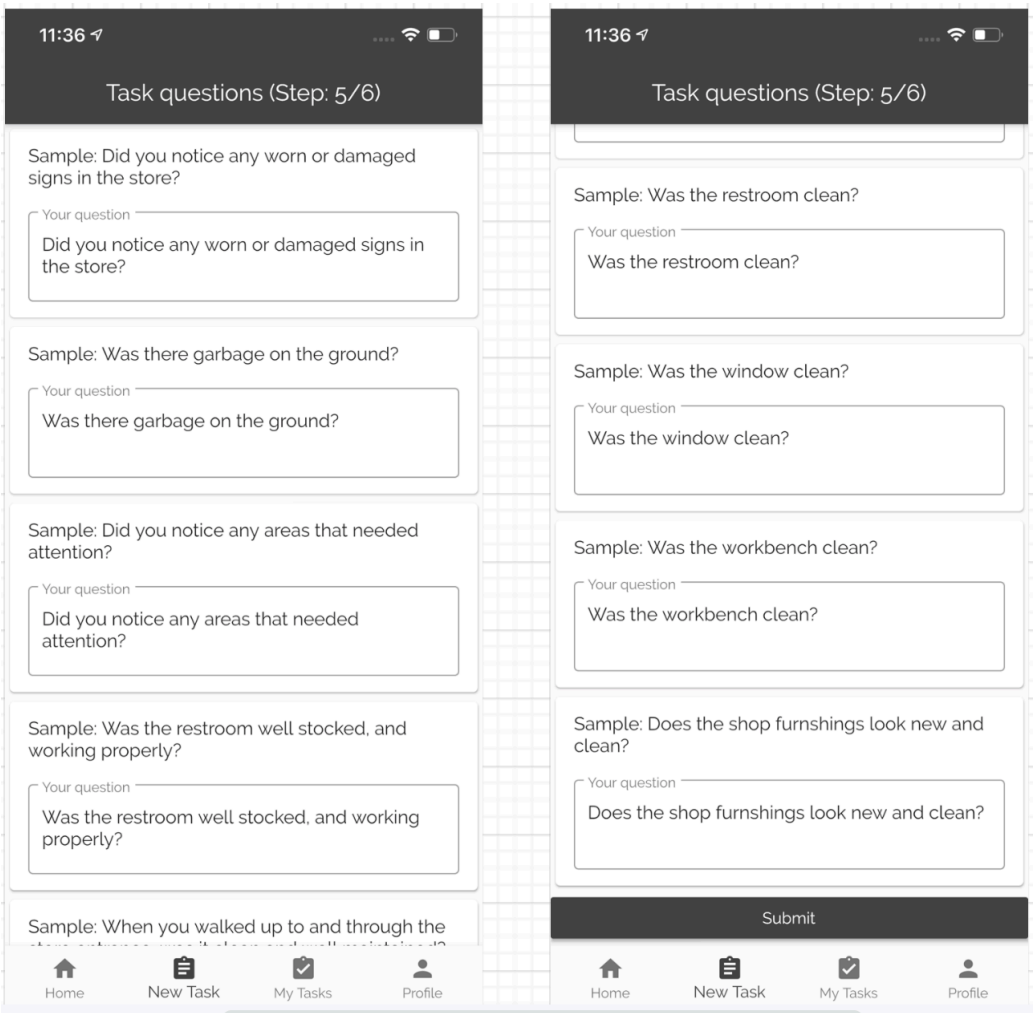


Enterprises are asked to fill in task information, which is task name, number of inspectors, inspector type, start date, and end date before they proceed to the third stage. The range of start date is from the current date to a year after the current date. The range of end date is from the start date to a year after the current date. The end date must be at least 30 days after the start date. If there is no error, enterprises can click the "Submit" button to proceed to the third stage.

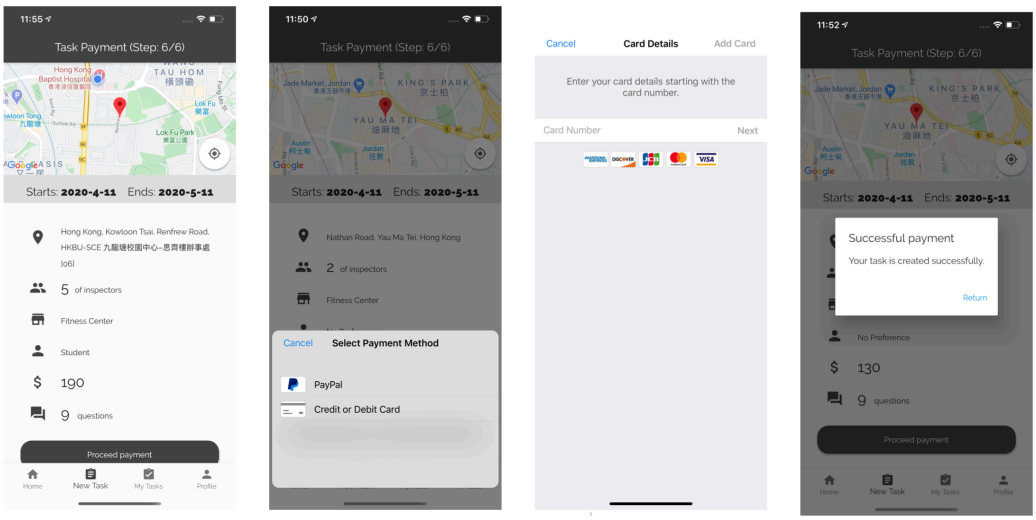




There are some templates provided for enterprises, therefore WeTell does not require too much user input, which is efficient. Clicking an item in the "Category" page, WeTell routes to the plan page. The corresponding plans in the category are illustrated. There are some brief descriptions for each plan, therefore enterprises will understand the plan before their selections. Click an item in the "Plan" page, enterprises proceed to the fifth stage.

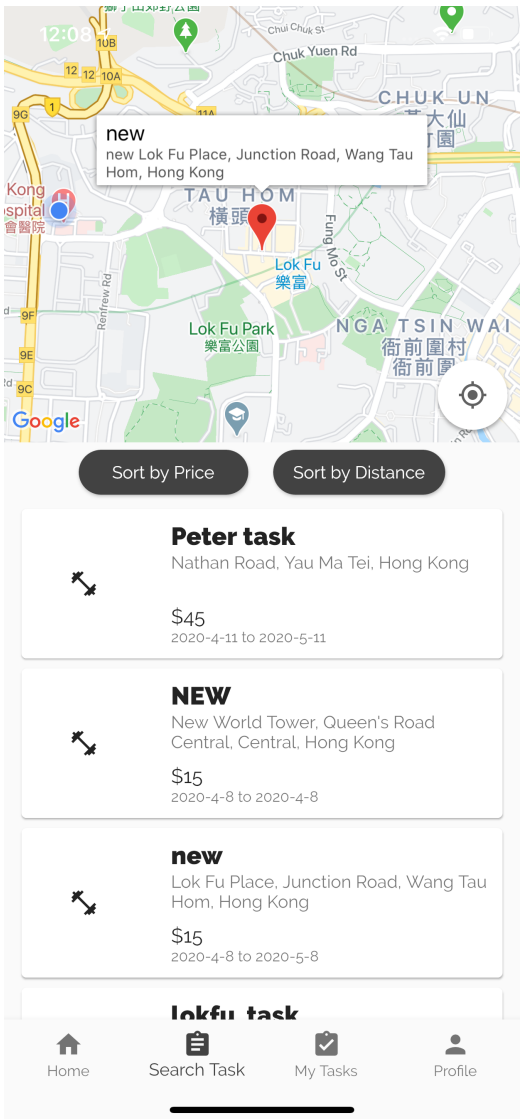


In the "Questions" page, the sample questions of the plan are presented in the card view. It contains a text view and text field for each card. The text field is filled with sample questions in default, therefore enterprises can simply edit partial content to create tailor-made questions for their tasks. After creating questions, enterprises can click the "Submit" button at the bottom of the page to proceed to the sixth stage.



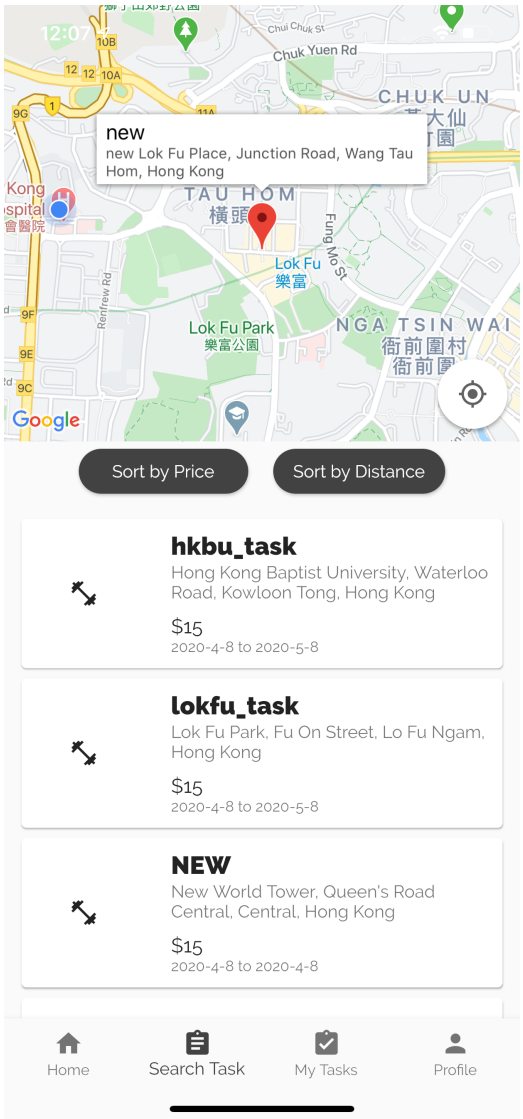
In the "Payment" page, the task details are presented for enterprises to review if the content is correct. The price of a task depends on the number of inspectors and the number of questions. The current price is the number of inspectors multiply by 20 and the number of questions multiplies by 5. Enterprises can press "Proceed payment" at the end of the page. There is two payment method which is Paypal and credit/debit card provided by Braintree. If the payment procedure is successful, a pop-up window is shown. The task creation is completed. The task can be searched and reserved by inspectors.

3.2.5 Search and Reserve Task (Inspector)

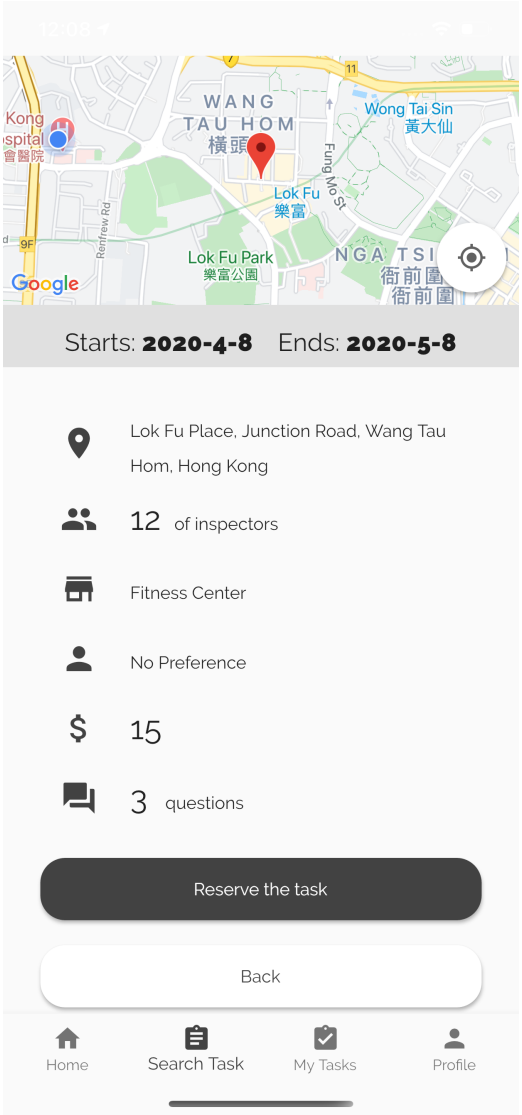


When inspectors tap on the "Search Task" tab, a list of filtered tasks is shown. If the inspector has reserved 6 or more tasks, no search results are shown. WeTell will show a message that the number of reserved tasks reached the quota, and remind the inspector to complete tasks. Also, the system matches the type of inspector and tasks' requirements. When enterprises create tasks, they can select a type of inspector. Therefore, if the inspectors do not satisfy the requirement, the task will not be sent from the server. When inspectors click the "Sort by Price" button, the list of tasks is sorted from high price to

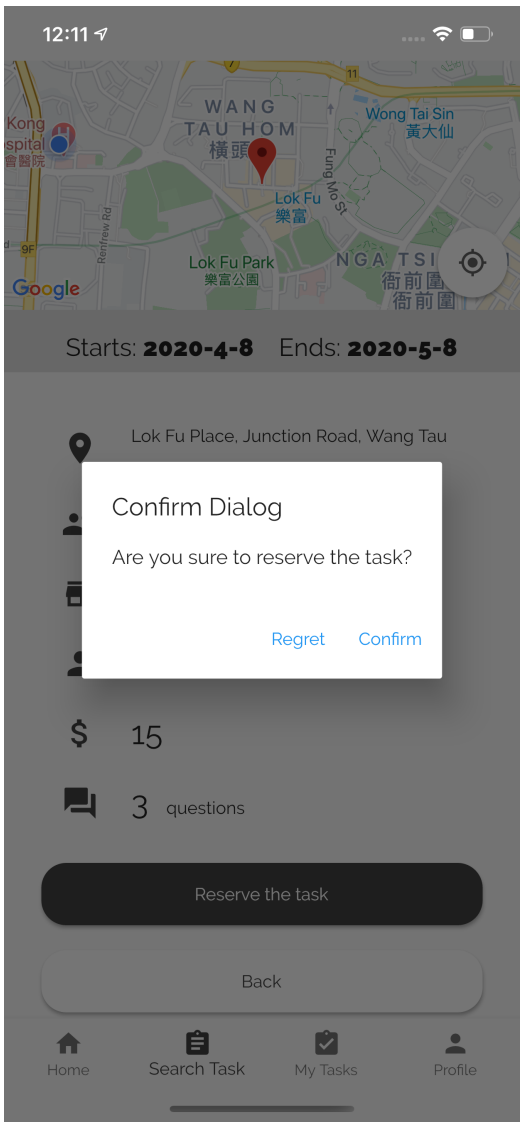
low price. In contrast, when inspectors click the “Sort by Distance” button, the list of tasks is sorted from low distance to high distance.



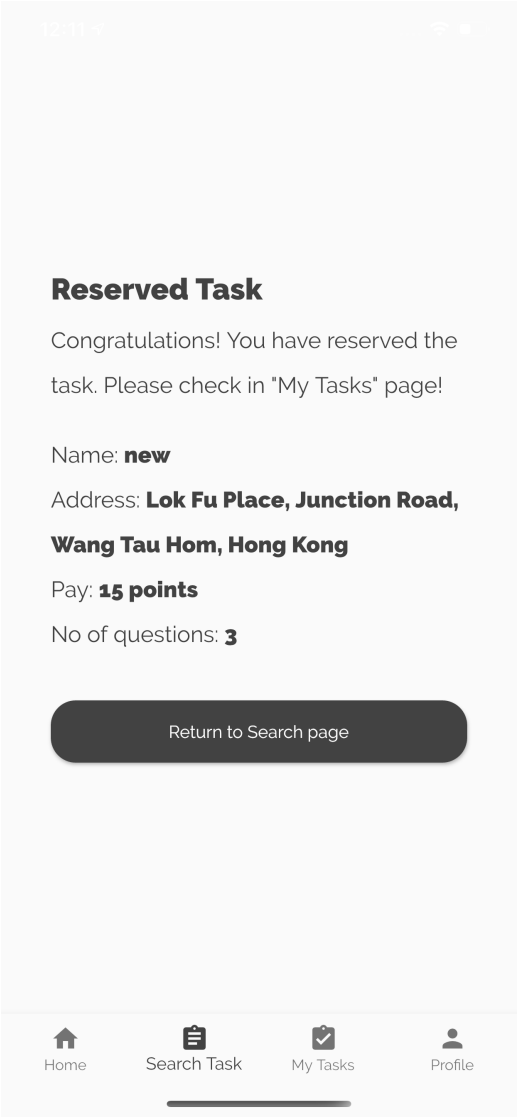
Moreover, WeTell applies Google Maps plugins to present the locations of tasks, which are located by the red markers. When an inspector clicks a marker, an information window with the task’s title and address is shown over the marker. The camera position will then focus on the task’s location. If the user clicks the information window, the detail of the task is shown.



In the "Task Detail" page, not only can users get detailed information, but users can also click the "Reserve the task" button to reserve the task. If inspectors click the "Back" button, WeTell navigates back to the "Search Task" page.

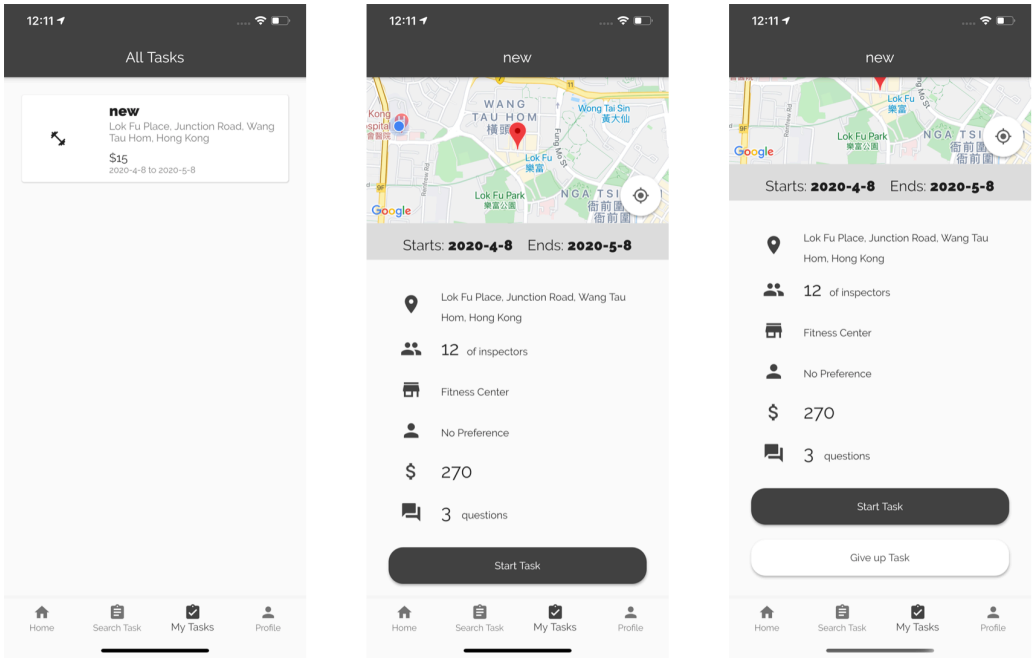


When an inspector clicks the "Reserve the task" button, a pop-up window is shown. If the inspector clicks the "Regret" button, WeTell navigates back to the "Task Detail" page. If the inspector clicks the "Confirm" button, the server will update the user record and task record. Completion of the reservation message will be shown on the page.

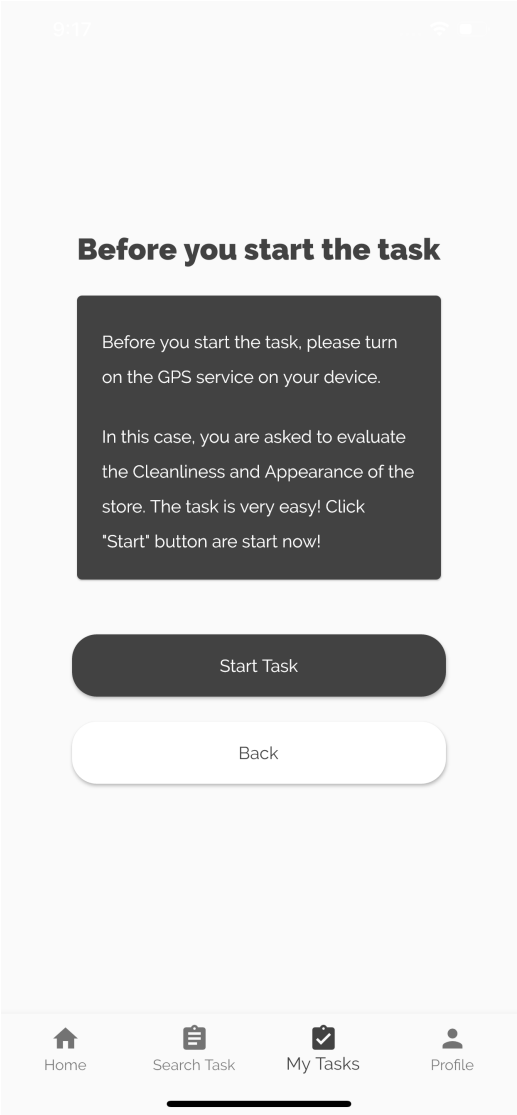


Tapping the “Return to Search Page” button, WeTell routes to the “Search Page”.

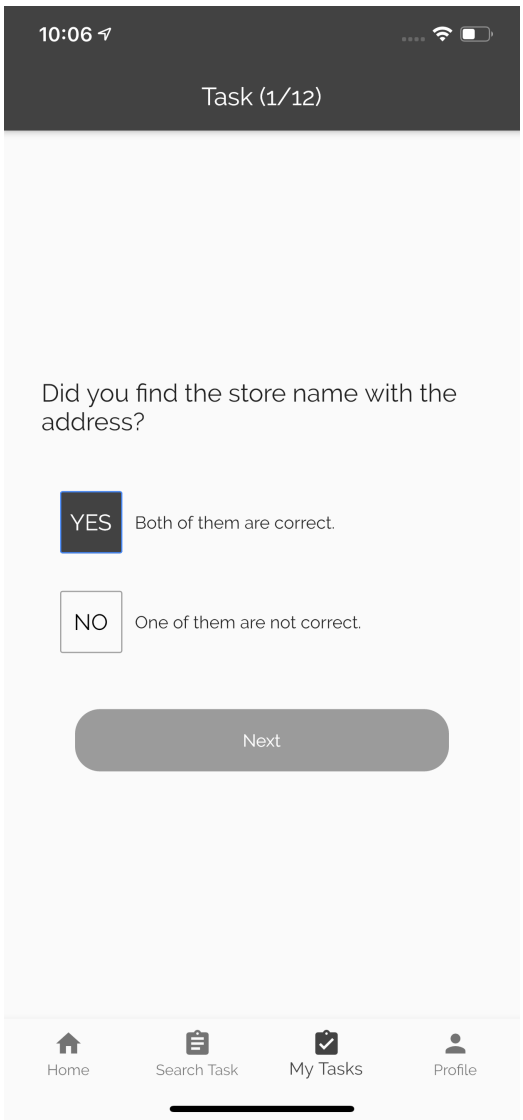
3.2.6 Complete Task (Inspector)



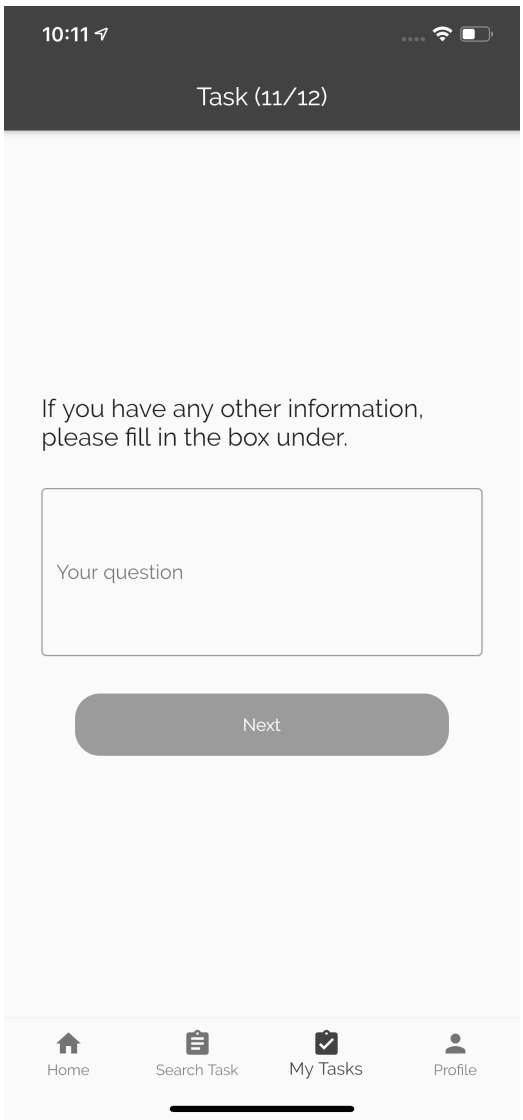
When inspectors click the "My Tasks" tab, a list of reserved tasks of the inspectors is presented. Brief information on each reserved task is described in a list view. The thumbnail on the left side represents the industry of each task. If the inspectors click one of the list items, a page with the task information is demonstrated. Scrolling down, a "Give up Task" button is at the bottom of the page. In case the inspectors intend to cancel the task reservation, they can simply click it. Then they can click the "Confirm" button in the pop-up window. As a result, the task will be removed in the list of reserved tasks. WeTell returns to view the list of reserved tasks. On the other hand, inspectors are allowed to start the evaluation on a task by tapping the "Start Task" button.



When an inspector clicks the “Start Task” button, WeTell navigates to the “Start Task” page. In the “Start task” page, the content of the card view is the reminder for inspectors before they start the task. It will be generated dynamically depends on the task information. If the inspector clicks the “Start Task” button, he/she will move on to the next stage. Otherwise, he/she can click the “Back” button to go back.

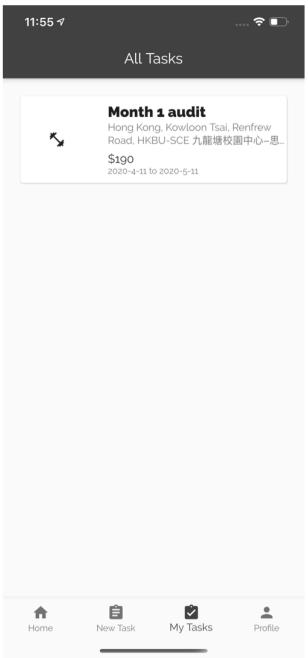


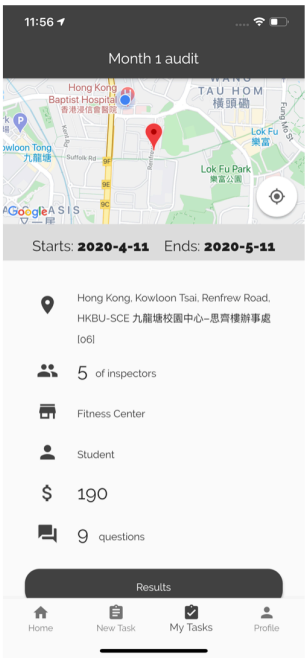
For all tasks, inspectors are asked to confirm the shop name and the address to avoid mistakes. If the answer is "No", WeTell returns a message showing that the task is invalid, then inspectors can click the "Return" button to go back to the "My Tasks" page. At the same time, a notification is sent to the corresponding enterprise to ask for the verification of the address and shop name. If the answer is "Yes", WeTell navigates to the next question.

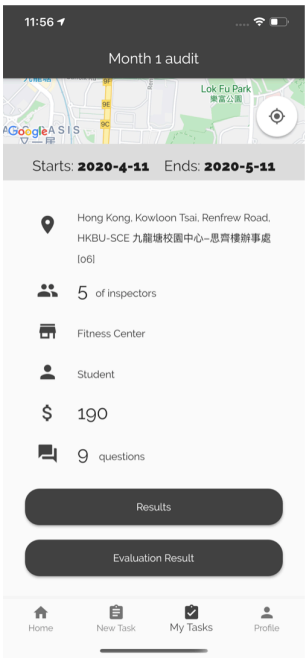


When all evaluation questions are answered, inspectors are asked to fill the text field if they have any extra information related to the task. If they do not have any, the field should be empty. Clicking the "Next" button, a review of answers is shown. Inspectors can edit their answers on the page. To submit the evaluation, inspectors tap the "Submit" button. The task points are added into inspectors' balance, which can be redeemed. The user record and task record are updated.

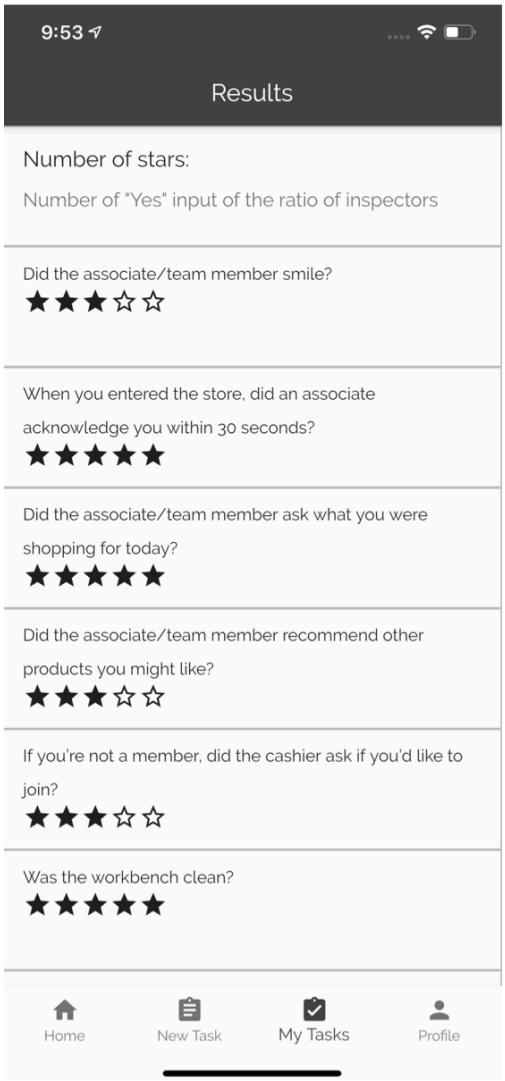
3.2.7 View Task Results (Enterprise)

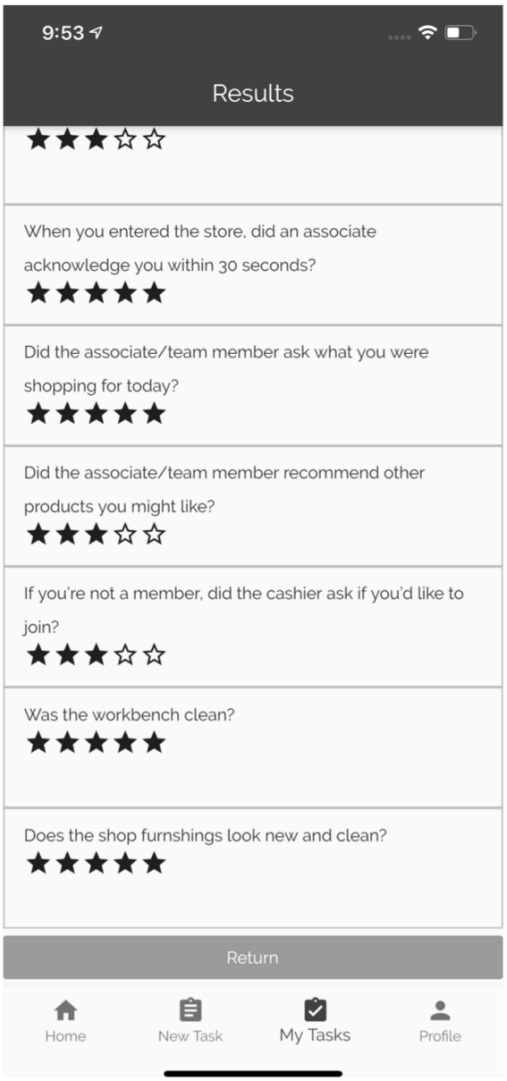




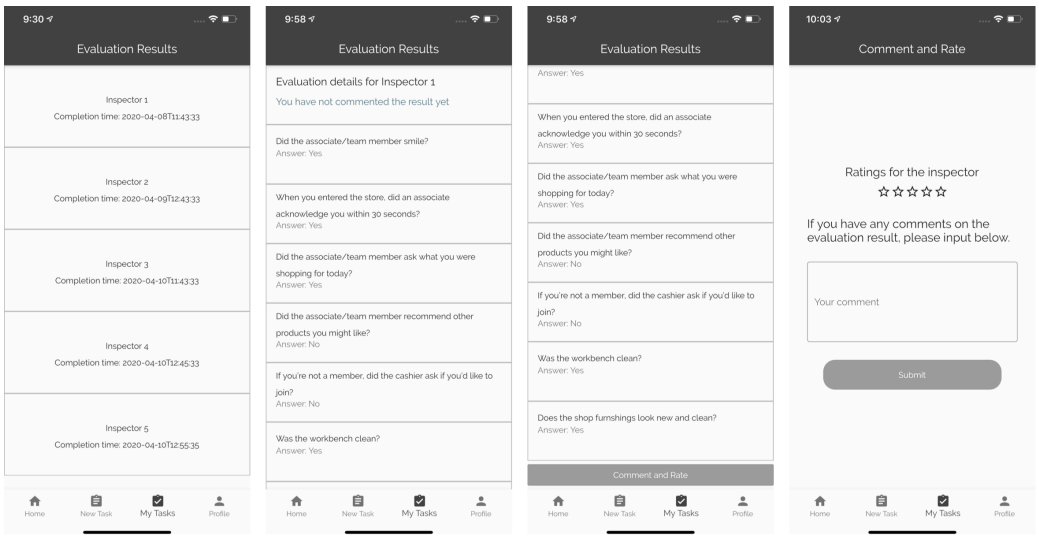


When enterprises click the "My Task" button, the tasks created by enterprises are presented. The format of the list view is similar to the "My Task" page in the inspector portal. Clicking one of the tasks, the details of the tasks are shown. Sliding down to the bottom of the page, there are two buttons, which are the "Result" button and the "Evaluation Results" button. The "Result" button is not clickable if the task is not completed yet, in other words, the task has not been completed by the number of inspectors yet. When the current date is after the end date of the task, the "Result" button is clickable but the output of the result is not complete. Moreover, clicking the "Evaluation Results" button, a list of evaluation records is shown. Enterprises are allowed to click the record and view the detail of the evaluation.





When enterprises click the "Results" button, all the handled results are shown. The results are the average of answers represented by star ratings. Scrolling down to the bottom of the page, enterprises can click "Return" and go back to the previous page.



When enterprises click the "Evaluation Results" button, a list of individual evaluation results is shown, which contains the inspector number and the completed time. Clicking one of the items, the evaluation results inputted by the inspector are presented. If the enterprise has not commented and rated the result, a reminder message is shown at the top of the page. Also, the "Comment and Rate" button is at the bottom of the page as well. Otherwise, the message and the button are not presented. Tapping the "Comment and Rate" button, the enterprise can rate and comment on the result.

3.3 Implementation of Maps Searching

In WeTell, when users search their locations, they are allowed to search by the destination. To implement this function, Google Places plugins through Flutter is used.

```
Future<Null> displayPrediction(Prediction p, ScaffoldState scaffold) async {
  if (p != null) {
    // get detail (lat/lng)
    PlacesDetailsResponse detail =
        await _places.getDetailsByPlaceId(p.placeId);
    final lat = detail.result.geometry.location.lat;
    final lng = detail.result.geometry.location.lng;
    final GoogleMapController controller = await _controller.future;
    _destination = p.description;
    initPosition = CameraPosition(
      target: LatLng(lat, lng),
      zoom: 15.0,
    ); // CameraPosition
    _goToThePlace(CameraPosition(
      target: LatLng(lat, lng),
      zoom: 15.0,
    )); // CameraPosition
      scaffold.showSnackBar(
        SnackBar(content: Text("${p.description} - $lat/$lng")),
      );
  }
}
```

Then, when users select a location, the maps view is changed.

```
Future<void> _goToTheCurrentPlace() async {
  final GoogleMapController controller = await _controller.future;
  Position position = await Geolocator()
      .getCurrentPosition(desiredAccuracy: LocationAccuracy.high);
  CameraPosition _cameraPosition = CameraPosition(
    target: LatLng(position.latitude, position.longitude),
    zoom: 15.0,
  ); // CameraPosition
  controller.animateCamera(CameraUpdate.newCameraPosition(_cameraPosition));
}

Future<void> _goToThePlace(CameraPosition cameraPosition) async {
  final GoogleMapController controller = await _controller.future;
  BitmapDescriptor pinLocationIcon = await BitmapDescriptor.fromAssetImage(
      ImageConfiguration(devicePixelRatio: 2.5), 'assets/mark.png');
  setState(() {
    this._markers.add(Marker(
        markerId: MarkerId('temparary'),
        position: cameraPosition.target,
        icon: pinLocationIcon)); // Marker
  });
  controller.animateCamera(CameraUpdate.newCameraPosition(cameraPosition));
}
```

In order to visualize the location of the tasks when inspectors are searching for new task, markers and infowindow are applied.

```
void addMarkers() async {
  if (tasks != null) {
    this.isLoading = false;
    tasks.forEach((f) async {
      BitmapDescriptor onValue = await BitmapDescriptor.fromAssetImage(
          ImageConfiguration(size: Size(48, 48)), 'assets/icon.png');
      Marker marker = Marker(
        markerId: MarkerId(f.id),
        position: LatLng(f.latitude, f.longitude),
        icon: onValue,
        infoWindow: new InfoWindow(
            title: f.name,
            snippet: f.name +" "+f.address,
            onTap: () {
              BlocProvider.of<SearchTaskBloc>(context)
                  .add(SearchTaskDetailButtonPressed(task: f));
              _goToThePlace(LatLng(f.latitude, f.longitude));
            }), // InfoWindow
      ); // Marker
      setState(() {
        markers[MarkerId(f.id)] = marker;
      });
    });
  }
}

Future<CameraPosition> getUserLocation() async {
  Position position = await Geolocator()
      .getCurrentPosition(desiredAccuracy: LocationAccuracy.high);
  this.initPosition = CameraPosition(
    target: LatLng(position.latitude, position.longitude),
    zoom: 15.0,
  ); // CameraPosition
  return initPosition;
}
```

Chapter 4

Discussion and Conclusion

4.1 Difficulties and Limitations

Since Flutter is a new technology compared to other mobile application frameworks, which is released in May 2017 and the current version is 1.12.13 [1]. Therefore, it lacks third-party libraries and packages. For example, for In-App payments in Flutter, it requires third-party libraries. However, the resources are mostly not well-documented, therefore it is difficult to implement the libraries. Comparing to Flutter, it is easy to search modules for the React Native framework on Google. Flutter's official resource of free packages is getting better, however, its list of tools is still improving.

4.2 Future Development

4.2.1 Enhancement on Security Issues

Since WeTell involves the transmission of sensitive data, such as personal data, credentials, and payment information, therefore security issues are important. Encryption is necessary to communicate securely over the internet. Implementing Secure Sockets Layer (SSL) certificates is needed. SSL is a standard security technology for establishing an encrypted link between a

server and a client. [4] All data transmitted across sites will have encryption deployed on the core switch level.

4.2.2 Enhancement on User Experience (UX)

The enterprise portal and inspector portal should be built in two separate apps. Also, for the "New Task" page on the enterprise portal, WeTell should allow users to custom their questions. Since the users may want to design a mystery shopper plan instead of using the template, WeTell should provide a flexible choice. Furthermore, the testing of the experience of the end-user is needed. A test plan and test cases should be applied to WeTell. Finally, the compatibility of the mobile is important. Although Flutter applies widgets to develop the user interface, the view of different devices may be different. For example, comparing iPhone X and iPhone SE, the layout of the "Task Detail" page may not fit iPhone SE, which may affect the user experience.

4.2.3 Enhancement on User Interface (UI)

The user interface in both the enterprise portal and the inspector portal is similar. It is hard to differentiate two portals. Moreover, the UI design of the functions is not clear enough. When the enterprises click the "New Task" button, it should be a clear description so that the user can understand the usage of the "New Task" page. Collecting opinions on WeTell from the end-user, WeTell user interface would be enhanced in user perspectives. Moreover, WeTell should provide graphical statistics for data analysis, so that enterprises can have a clear view of the evaluation.

4.3 Conclusion

WeTell provides a peer-to-peer platform between enterprises and inspectors, so agents are no longer needed. Also, WeTell reduces manual operations. When enterprises create tasks on the platform, the inspectors can actively

reserve and complete tasks. The process does not involve too many manual operations. WeTell can solve the problem of the process of a mystery shopping program, which is time, money, manpower consuming, and cost-ineffective. Moreover, WeTell also combined the advantages of existing applications, such as the user interface. WeTell also learned the weakness points from the existing applications, which as the process of creating an account and getting tasks. Nevertheless, WeTell needs enhancement on security issues, functional issues, and user interface before deployment to the market. I hope that WeTell would be well-developed in the near future.

Appendix 1: System Setup

A1.1 Server (Sails.js)

1. Run “npm install sails -g” in Terminal
2. “cd” into the directory
3. Run “sails lift”

A1.2 Flutter (Windows)

A1.2.1 Mininum requirements:

Operating Systems: Windows 7 SP1 or later (64-bit) Disk Space: 400 MB (does not include disk space for IDE/tools). Tools: Flutter depends on these tools being available in your environment. Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10) Git for Windows 2.x, with the Use Git from the Windows Command Prompt option.

A1.2.2 Installation:

1. Run “git clone https://github.com/flutter/flutter.git -b stable”
2. Download and install Android Studio.
3. Start Android Studio, and go through the ‘Android Studio Setup Wizard’. This installs the latest Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools, which are required by Flut-

ter when developing for Android.

A1.3 Flutter (MacOS)

Mininum requirements: Operating Systems: macOS (64-bit) Disk Space: 2.8 GB (does not include disk space for IDE/tools).

A1.3.1 Installation:

1. Run “git clone https://github.com/flutter/flutter.git -b stable”
2. Install the latest stable version of Xcode

References

[1] Flutter (n.d.). Inside Flutter. Retrieved April 2020, from https://flutter.dev/docs/resources/inside-flutter

[2] Sails.js (n.d.). Realtime MVC framework for Node.js. Retrieved April 2020, from https://sailsjs.com/

[3] MongoDB (n.d.). What Is MongoDB?. Retrieved April 2020, from https://www.mongodb.com/what-is-mongodb

[4] digicert (n.d.). What is an SSL certificate?. Retrieved April 2020, from https://www.digicert.com/ssl/