Designing a Pull-Based Automated System of Mobile Mechanic Service

Majid Abarghooei

Abstract—It is an inevitable fact that handheld devices (smartphones, tablets, etc.) play an important role in our daily lives in these days. These gadgets precipitate the time-consuming tasks. One of these aspects in which handheld devices can play a vital role is companies which present a special service such as mobile mechanic services. In this paper we intend to present a system in which the process of doing these kinds of services is carried out much faster than before by handheld devices supporting 3G and equipped with GPS chipset. For this purpose we will design a mobile app and a server-side application benefiting from JSON as a data interchange standard. We also address the architecture of this system and the challenges which should be overcome. This paper is the result of doing an identical project in one of these companies.

Index Terms—Empirical software engineering, software architecture and design, use cases and scenarios.

I. INTRODUCTION

As it was previously mentioned, our main purpose is designing a system to precipitate the process of service providing. First of all we are going to have an introduction to this process. This process is common in companies which provide mobile mechanic services but it can be applied to the companies with different businesses which follow the identical procedure. Hence, from now on we use mobile mechanic services to illustrate the system.

The whole process of providing service is as follows:
1) A customer contacts the call center and requests a service.
2) The operator finds a free mechanic and assigns a new task to him.
3) The mechanic goes to the address in which the customer is waiting for the service.
4) The mechanic calls the center and informs them of his presence in the address and start of providing the service.
5) When the task is done, the mechanic calls the center again and informs them of the end of the service process.
6) The operator adds the mechanic to the queue of free mechanics for further tasks.

In this paper we try to address a solution to precipitate this process using handheld devices. Some of the aforementioned companies are so busy because of the huge number of telephone calls from customers. On the other hand, each of the mechanics calls the center at least two times per each service which can considerably increase the number of telephone calls of the center. Hence, we are going to find a solution to reduce the mechanics’ calls and expediting service providing process. This means that mechanics can do the whole process independently without calling the center and informing them of the start and end of the service. In what follows, we are going to present a solution and discussing its challenges. From now on, we abbreviate the mobile mechanic service system as “MMS”.

As it was mentioned, the new solution is based on handheld devices. Hence, before discussing the new solution we try to have a brief definition of handheld devices. “Handheld devices are extremely portable, self-contained information management and communication devices” [1]. A candidate must operate without cables (except temporarily), be easily used while in one’s hands (not resting on a table), and allow the addition of applications or support Internet connectivity [1].

II. DISCUSSING THE NEW SOLUTION

As it was previously stated, all of the mechanics are provided with a handheld device which supports 3G SIM card and has been equipped with GPS chipset.

The system we intend to design should consist of two major modules:
• Server-side application
• Mobile application

To illuminate these two modules we should go back to the process which was explained in previous section now supposed to be done by the new solution. In that process when a customer calls the center and requests a service, the operator registers a new service and assigns this task to a free mechanic. In the new solution all of the requests should be stored in the database using the server-side application and after assigning a new task to a free mechanic, the operator does not inform the mechanic using telephone call. Instead of the traditional telephone call method, in this solution, the server-side application sends the new task to the mechanic’s handheld device. When he arrives to the location in which the customer is waiting for the service, he uses his device to inform the center of starting or finishing the service.

Now it is time to restate the functionalities of the two aforementioned modules clearly, according to the previous definition of the new solution. Fig. 1 shows the MMS’s outline.

A. Server-Side Application

This application is incumbent on storing all of the new tasks and their status (waiting for service, start, finish, cancel, etc.). This application also keeps the mechanics’ statuses.
(free, going for service, doing service, etc.) and after assigning a new task to each of them, it sends the new task to the mechanic’s handheld device.

**B. Mobile Application**

The most important functionality of this application is synching with the server and sending the service start and finish time. All of the handheld devices are equipped with GPS chipset which makes the device able to send its geographical information to the server as well.

![Fig. 1. MSS outline.](image)

### III. MMS ARCHITECTURE

First of all we address the architecture of the system. MMS, as it was previously mentioned, consists of two major parts; a server-side application and a mobile application. In this section we address the server-side app and client mobile application and discuss their modules and major functionalities to illustrate the architecture of MMS.

What is a server-side application? Why we prefer to use it? To answer this question we bring up a scenario. Suppose that there is an application which works with a bunch of sensitive data. We have decided to store the data on the device. What if the device is lost or encounters a serious problem which leads to data loss? This is the most important problem of storing data on the handheld device. The server-side application can solve this problem. In this way the client mobile app only manipulates the data or displays it but it is not stored locally (permanently). Actually we are applying a thin client approach to the mobile app [2].

According to what was contended, we try to store the sensitive data on the server instead of storing it locally. Now the client mobile application should be synchronized with the server. First of all we need a data interchange format in order to sync the data between mobile app and server-side app; we use JavaScript Object Notation (JSON) for this purpose. Fig. 2 depicts the outline of MMS’s architecture.

![Fig. 2. Outline of MMS’s architecture.](image)

Why we use JSON to sync data? Mostly JSON is more appropriate to data interchange. JSON has many advantages compared to Extensible Markup Language (XML) [4]. In XML every element in the tree has a name and should be enclosed in a matching pair of tags [4]. However, JSON presents trees in a nested array format identical to what is existed in JavaScript [4]. In this way we can transfer same data in a far smaller package with JSON than with XML. Not only is JSON easy to read for humans, but also it is easy to parse for machines. JSON supports both objects and arrays [4]. An object is an unordered name-value pair, while an array is an ordered sequence of values [4]. It also supports strings, numbers, Boolean expressions, and null values. This makes JSON able to describe any resource [4].

Now we go back to the MMS architecture. As it was stated formerly, we use JSON data-interchange format to sync data between mobile application and server-side app. The server-side app has other functionalities, other than storing data, that further we address them in detail. Now we want to discuss the server-side architecture in detail. We consider a 3-layer architecture for the server-side application [5]. According to Fig. 3, which depicts the outline of the server-side application’s architecture, presentation layer contains two main modules:

- **Authentication Module**
- **Sync Module**

### A. Authentication Module

In order to increase the security of the system we provide each handheld device with a unique device ID which has been registered to the server. We also provide each user with a username. Whenever the mobile application tries to be synchronized with server, it should encrypt the device ID with the public key of the server and send it with the data (using JSON format) to the server. The authentication module tries to decrypt the device ID with the private key to be sure that the data is sent from a valid sender. Otherwise the request is rejected by the authentication module.

This module has another responsibility; it should check the username and password when a user (mechanic) logs in to the system. When a user wants to start working with the mobile...
application, they should first login and after authentication process they can use the device.

B. Sync Module

This module is responsible for the synchronization process. When the mobile application sends the data to the server this module receives the data, which is prepared in JSON format, then extracts the data and makes the object of the data and pass it to the service layer. When the server is sending the data to the handheld device this module converts the data object to JSON and sends it to the device.

![Fig. 3. The server-side App's architecture outline.](image)

As Fig. 4 shows, the mobile application contains four main modules:

- **Data Access Module:** This module manages saving the data to the local storage and accessing to the stored data in order to sync to the server.
- **Service Module:** All of the services which the handheld device is incumbent on, are done by this module.
- **Sync Module:** This module is responsible for syncing the device to the server. It also handles the timing of the sync process.
- **Presentation Module:** This module is responsible to interact with User Interface (UI) and display the data.

There are many architectures that can be applied to mobile applications according to their functionalities. They can be designed and implemented as a native application, a cross-platform application or a hybrid application. All of the previous architectures have some advantages and disadvantages. Although implementing native mobile application needs investing more time and effort, according to MMS’s requirements we prefer to implement a native app because accessing to device features is far easier than other methods. Handheld devices which are used in MMS are exactly identical and using cross-platform methods only limits the features which we can use.

![Fig. 4. The client mobile application’s modules.](image)

IV. MMS DESIGN AND CHALLENGES

Despite the straightforward architecture of MMS, there are some important challenges in its design. One of the important functionalities of the server is sending new tasks to related devices. However, how the server should send the new task to the device? What is the IP address of the device? This is exactly the problem. Unfortunately, none of the devices which are equipped with 3G SIM cards have static IP address. On the other hand, we cannot hold a connection between the device and the server open because it is possible that a mechanic is not assigned with a new task for hours after his latest task and this can make many idle connections. This is the most important challenge which we are going to overcome.

According to the previous statements, we have only one solution and it is applying a pull-based method. As it was contended formerly, the server-side application cannot push the new task to the device. Handheld devices are not provided with static IP address because the number of mechanics and their devices is not fixed and can be increased. Also providing a bunch of static IP addresses from the service provider company is not practical. Suppose that there are 200 mechanics in a mobile mechanic service company; this means that we need 200 static IP addresses. Hence, we cannot found our system on a method which is highly depended to a third party or an impractical method.

In order to design MMS with pull-based method, mobile application which is running on the handheld device should pull the new task. Mobile application tries to connect (sync) to the server periodically and claim its readiness to do a task and inform the server of its status and geographical position. Bear in mind each time that mobile application synchronizes with the server, it sends the data using JSON format to the server then server-side application checks if there is any new task for the related mechanic. If there is a new task, server-side application sends that to the device.

Determining this period is one of the most important tasks in this system because it directly affects the performance of the system. Suppose that we consider a two-minute period. In this way, in worst case, the mechanic receives the new task with two minutes delay. This period should be set according to the average number of new tasks in a day and the server’s workload.

The handheld device stops periodic syncing with the server as soon as it receives the new task and shows it to the mechanic. The mechanic starts doing his job and inform the center from starting the service using the handheld device. Hence, the mobile application sends the required information to server. The server-side application changes the status of the task and the mechanic. This process will repeat while the mechanic finishes the service and wants to inform the center. Fig. 5 and Fig. 6 depict the aforementioned process. Note that authentication process is not shown in this figure in order to simplify its understanding. The authentication process further would be delineated in detail.

There is still another problem to be solved. Suppose that the mechanic is in a place with weak signal or no network coverage. In such a condition the handheld device tries to sync with the server but because of the weak signal it does not succeed. On the other hand, a new task has been assigned to the mechanic. As long as the mechanic does not change its location and signal strength does not change, this problem...
will exist. This condition can exist even for an hour or more. Hence, we should consider a backup solution to overcome the challenge.

![Diagram](image)

**Fig. 5.** The periodic synchronizing of the mobile app.

One of the practical solutions which can act as a lifesaver in conditions which internet access encounters a problem because of the network coverage or signal strength, is using text message (SMS) in order to send the data and sync with the server. All of the steps previously mentioned can be done using SMS. In this scenario, if the handheld device does not succeed to post the JSON data to the server, it attempts one more time. If the second attempt fails, the device uses SMS to synchronize with the server [6] (Reference [6] shows how to implement this process using SMS). If there is a new task which should be sent to the device, the server should send it as a text message. Hence, we should add another module to the server-side application responsible for receiving and handling text messages. If sending the text message faces the problem too, the mobile application stores the message and sends it as soon as the internet connection is provided. “SMS is a store and forward system. In other words the server will receive sent messages and hold on to them until they can be delivered [7]”. This makes SMS an ideal way to deliver information or communicating in infrequent or unreliable connectivity [7].

The last piece of the puzzle is login process. As it was stated in Section III (MMS Architecture), the server-side application has an authentication module which is responsible for checking username and password when the user tries to login. If the login process succeeds the server-side app changes the mechanics status to online. Now they are authenticated to use the app. If the time from the last attempt of the device for syncing with the server exceeds a specific limit, the mechanic’s status changes to offline.

The final scenario which should be considered is a situation in which none of the mechanics are free. In this situation server-side application should make a queue of new tasks and assign them respectively to mechanics as soon as they get free.

### V. CONCLUSION

MMS can be applied to the companies which provide a specific service like mobile mechanic services in order to reduce the workload of the call center and expedite the process of service providing. We tried to focus on challenges in this system and presenting a practical solution for them. This system can be expanded using web services in order to interact to integrated software which the company may use. Also many other functionalities can be added to it such as GPS tracking using the information which handheld devices send to the server. This can be considered as future works in order to ameliorate the system’s functionality and needs more researches and consideration.

### REFERENCES


**M. Abarghoee** was born on November 30, 1987, in Tehran, Iran. He received his B.A. degree in software engineering at Islamic Azad University South Tehran Branch from 2006 to 2009.

In January 2011 he started working at Pars Rayan System Qeshm as a senior java developer. Next year he started working at Tourism Card Company as a software developer and IT consultant. He achieved Oracle Certification of Professional Java SE 6 Programmer (OCPJP 6) in January 2013. In June 2013 he started collaborating with Rightel Telecommunication Company, MTIC department, as a Java developer and service contractor.