Offline Data Synchronization in Order Management System

[1]Sandya A [2]Soumya J [3]Sree Laxmi Durga M [4]Swetha B
[5]Dr. L Swarna Jyothi, [6] Prof. Jagadesh R M
[1,2,3, 4]-Students Dept. of CSE, BITM, Bellary,
[5]-Head of ECE, RRCE, Bangalore,
[6]-Professor, Dept. of CSE, BITM, Bellary.

Abstract: We live in an increasingly connected world. However, in many cases we cannot rely on connectivity 100 percent of the time. Your users may travel, they may temporarily lose wireless connectivity, there may be latency or bandwidth problems, or you may need to take down parts of the network for maintenance. Even if users do have good network connectivity, your applications may not be able to access network resources all of the time. A requested service could be busy, down, or just temporarily unavailable.

An application is *occasionally* connected if at times it cannot interact with services or data over a network in a timely manner. If you can allow your users to be productive with their applications when they are offline, and still provide them with the benefits of a connected application when the connection is working, you can increase user productivity and efficiency and increase the usability of your applications.

Keywords: OCC, IPMS, OMS, HVS, PIMS, Client, Server

1 Introduction:

A prospective solution to the problem stated above is presented by Occasionally Connected Computing (OCC). This software architecture is aimed to allow the user to use an application and register required changes into the application, even without a network Connection. The application registers the user's request and sends these requests to the server as and when network connection is re-established. The architecture can be implemented and incorporated in a typical client-server model by creating a local database at the client end also. The client database is a mirror image of the server database. Whenever the client places a request, the client application processes this request locally and updates its local database. The application then checks for network availability. If a valid network connection is available, the client application sends a database update request to the server. If no connection is available, the client application waits for network availability. This methodology ensures completion of processes even without a network connection. This modification can be made to Insurance Policy Management System (IPMS), Property Inventory Management System (PIMS), Health Vault System (HVS) and Order Management System (OMS) as discussed above.

The below Figure1 show a graphical illustration of the general system structure for better understanding. The clients shown in figure1 have temporary local database. The connection between the clients and the server may or may not be available. The server contains the main database of the system. The interfaces eases interaction between the device and the server by exposing API's for retrieval of data, adding data, and editing data present in the database of the server.

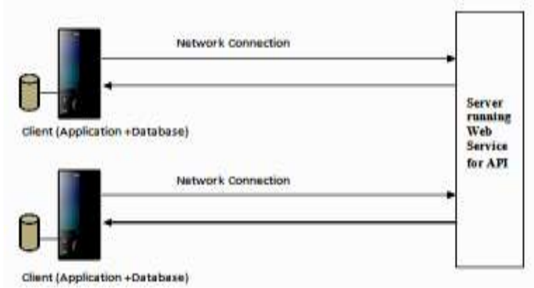

Network Connection
Client (Application +Database)
Network Connection
Client (Application +Database)
Server running Web Service for API

Figure1. General System Structure

One of the primary benefits of smart clients over Web-based applications is that they can allow users to continue working when the application cannot connect to network resources. Occasionally connected smart clients are capable of performing work when not connected to a network resource and then updating network resources in the background at a later time. The update may happen almost immediately, but sometimes it can happen days or even weeks later.

This work looks at the issues that you face as you build applications with offline capabilities. It reviews different strategies for designing offline applications, discusses in detail design considerations, examines how to structure applications to use tasks, and looks at how your applications should handle data.

2 Web applications currently in offline mode

Web applications run through a web browser like Internet Explorer. The program sits on a web server, rather than on the PC, or local server for traditional applications. On a basic website pages are static. Web application pages interact with users requesting and responding to users. The most common example is online shopping applications .Web applications typically use a database to store permanent information such as product descriptions and costs, and customer orders. Web Applications deliver many business benefits compared to office based solutions.

But the major disadvantage of web application appears if the internet connectivity is lost in the middle of transaction. Many systems don't operate in environments where client machines have uninterrupted connectivity with server machines. The network is not reliable. Most of the applications do work online. There are many chances of data loss or data corruption if internet connectivity is lost in the middle of transaction.

3 Proposed Software System

Occasionally connected smart clients are extremely useful in many common situations. Many offline scenarios involve the user explicitly disconnecting from the network and working without a network connection, for example **:** Considering an e-commerce site where

- The customer search for the required category of product and place order for the product
- While placing order, suppose that if the connection goes off ,his order for the product will not be cancelled instead it is stored in the temporary buffer and later, when the connection is on, the order is placed automatically.
- Occasionally connected smart clients should be designed to take maximum advantage of a connection when it is available, ensuring that both applications.

Benefits of this system are:

- Allow users to continue working when the application cannot connect to network resources.
- Reduce the data loss.
- Increase Robustness.
- Provide user with a friendly environment where he will not be distracted with network failure.
- Avoid re-entering data.

3.1 Online Application as offline for ordering

This paper contains a discussion of the issues you might face when designing and building smart client applications that are occasionally connected to the network. This covers the concept of connectivity, describes the two main approaches to implementing offline capabilities, and discusses some of the things you need to consider to make your application available when offline.

To given an application full occasionally connected capabilities, we need to provide an infrastructure that allows users to work when they have no connection to network resources. This infrastructure should include data caching, so that all required data is available on the client, and storage of the details of users work, which can be used to synchronize the client and network resources when the user goes back online. The exact features and capabilities that your application needs to support occasionally connected operations depends on its connectivity, operational environment, and the functionality that the user expects when online and offline.

3.1.2 Design of Order Management System

Order Management System is a web-based platform to store and maintain orders information. The Order Management record stores individual’s order information. The individual customer orders are taken and stored in the database, so as to identify the orders made by the customers for the available products. The customer can be identified by their customer id. Once the order has been placed by providing the required details, A notification is sent to the user that his/her order has been placed and will be delivered within the specified period.

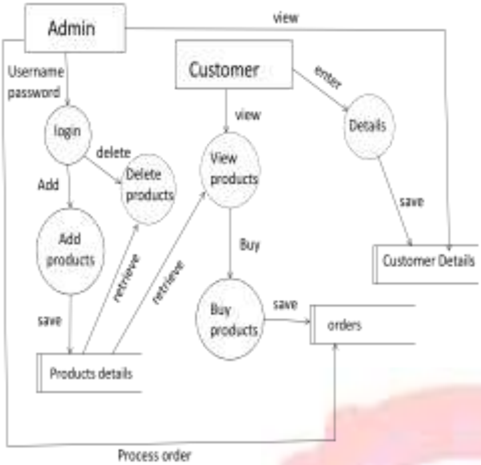


Figure2. Data Flow Diagram for Order Management System

3.1.3 Implementation and Results

Admin Module:

In this module Administrator can register the new products, view the customer orders, and process the customer orders.

Figure3. Admin login

Figure4. Admin Procedure

Figure5. To add categories

Figure6. To add products

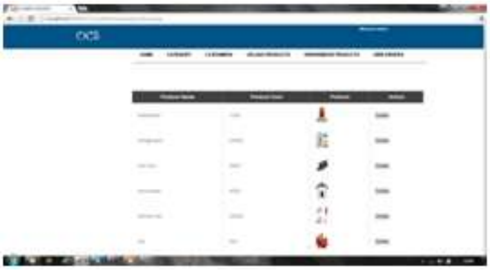


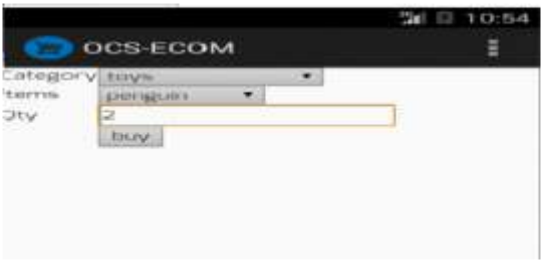
Figure7. View all products

Figure8. View the customer details

Figure9. View the customer order

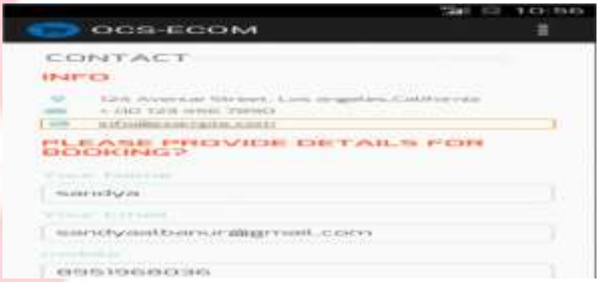
Customer Module:

Once the customer is registered to E-commerce site, he/she can login and place an order for an item. The order will be placed even if the network connection is low.



Category toys
Items penguin
Qty 2
buy

Figure10. Customer select the product



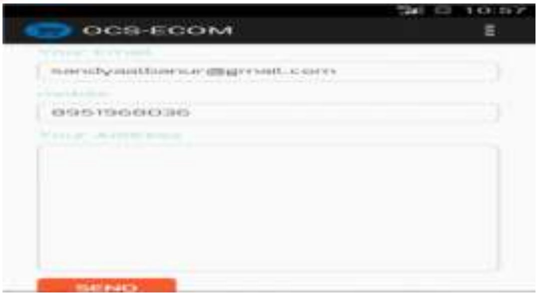
CONTACT INFO

124 Avenue Street, Los angeles,California
+ 00 123 456 7890

PLEASE PROVIDE DETAILS FOR BOOKING?

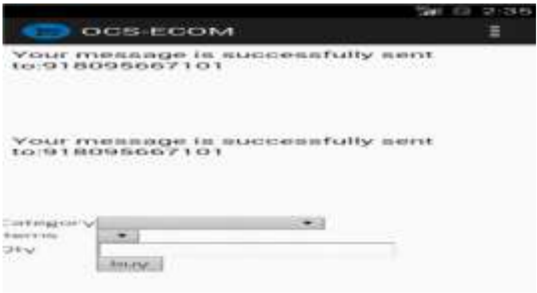
sandya
sandyaaitbanur@gmail.com
8951968036

Figure11. Customer enter details



sandyaaitbanur@gmail.com
8951968036

Figure12. Customer enter details



Your message is successfully sent to:918095667101

Your message is successfully sent to:918095667101

Figure13. Sending successful messages

4 CONCLUSION

Smart clients need to operate efficiently when connected and disconnected from the network. As you design your smart clients, you need to ensure that they can function effectively in both situations, and transition seamlessly between the two.

There are two broad strategies for designing smart client communications: service-oriented and data-centric. When you have determined which of these to use, you need to make some fundamental design decisions to allow your smart clients to work offline. In most cases, the clients should be designed to use asynchronous communication and simple network interactions. Clients will need to cache data for use when offline and you will need a method to handle data and business rule conflicts when the clients go back online. In many cases, offline clients allow users to perform a number of tasks that are dependent on one another. You will need to deal with these dependencies in the event that one of the tasks fails when it reaches the server.

The task-based approach can dramatically simplify the process of taking applications offline. Consider implementing this approach in your smart clients; it can also provide you with an effective way of handling dependencies, both at the server and at the client.

5 REFERENCES

[1] A Web-based, Offline-able, and Personalized Runtime Environment for executing applications on mobile device by Yung-Wei Kao, ChaiFeng Lin, Kuei-An Yang, Shyan-Ming Yuan C. Computer Standards & Interface 34 (2012) 212-224,. ELSEVIER.
[2] A Service Oriented Livestock Management System Using Occasionally Connected Mobile-Cloud Architecture by Chia-Chi Teng, Kevin Brown, Craig aro, Whitney Nielsen, Jason Wells. 2012 IEEE.
[3] Occasionally Connected Research and Application under the Architecture of Smart Client by Rao Xuli, Yang liuqing. The 7th International Conference on Computer Science & Education (ICCSE 2012) July 14-17, 2012, Melbourne Australia. 2012 IEEE.
[4] The Workflow Based Architecture for Mobile Infromation Access in Occasionally Connected Computing by Ali Bahrami, Changzhou Wang, Jun Yuan and Anne Hunt. IEEE International Conference on Services Computing (SCC'06) 2006 IEEE.
[5] Solving Problems in Software Applications through Data Synchronization in Case of Absence of the Network by Isak Shabani, Betim Cico, Agni Dika. International Journal of Computer Science Issues 2012.
[6] Offline Data Modeling in Software Applications through Smart Clients by Jagadish R M and Dr. L Swarna Jyothi

[7] Synchronization Framework for Occasionally connected Applications by Chitre United States Patent.

[8] Smart client technology applied in the management of military equipment by Micro computer information, 2008 (24).

[9] Communication services for advanced network applications by J Bresnahan, I Foster, J Insley, S Tuecke. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, Volume IV, pp. 1861-1867.