# Problem and Solution Statement

MoneyClip Mobile (MCM) is a web-based payments infrastructure providing fee-free transactions to customers and merchants through the use of mobile smartphones.  Mr. Joshua Cross of Hermes Commerce, Inc. is planning to make this mobile payment system more robust by providing targeted advertisements, coupons, and managing customer’s royalty programs.

This capstone project will develop the localization features for the MCM advertisement, coupons, and loyalty network.  The software will be developed for the existing MCM network and database to ease the transition of integrating this feature.  The team will be developing the same copy of the existing network and database on a local server to develop in and test the added feature.  The prototype of the localization features implemented on the database, web service, and iOS/Android applications will be presented to Mr. Cross at the end of this project.

Part 1: Problem Statement

This project is sponsored by Mr. Joshua Cross, CTO of Hermes Commerce Inc. in Flagstaff, Arizona. Hermes Commerce Inc. produces an application called MoneyClip Mobile (MCM) that allows users to transfer funds to and from other users, merchants or customers, without involving physical money or credit and debit cards. Mr. Cross is responsible for the development of MCM as well as for the business aspects of the product. MCM addresses the need of users to exchange funds regardless of distance.

The general problem facing MCM is competition with popular technologies such as Google Wallet and Square. Mr. Cross’ goal for overcoming this challenge is to combine all the functionalities of the competition into one application and find a niche for that application. In order to accomplish this, MCM application needs to be able to transfer funds between customers and merchants.  In addition, the application needs to determine when a customer is making a purchase from a merchant in order for the merchant to push the charge to the customer’s MCM account. Also, MCM will give customers control over the frequency and types of advertisements or coupons they receive, and it will give merchants options for when to push advertisements or coupons depending on customer trends.

This project involves implementing a localization function that notifies merchants when MCM customers are in their vicinity, allowing the merchant to push a payment request to the customer’s MCM account if the customer makes a purchase. This functionality allows true moneyless transactions. The third and final chunk of functionality that we will be developing is the data analysis algorithms that will be used for effective direct advertising. We will be implementing trend finding algorithms that will be used for direct advertising. These algorithms will use past user transactions to find patterns and compute probabilities that will be available to merchants to aid their advertising decisions.

Part 2: Problem Solution

System Improvements

One piece of functionality we are adding is the ability to allow customers to select the amounts and sources of advertisements that are pushed to their account via the iPhone application, Android application, or website. They should be able to customize their settings to choose differing levels of participation from the advertisements.  Merchant settings should also be available for change on the website or either mobile application.

The next function being added is the analysis of customer traffic to find patterns that merchants can use for direct advertising. The merchants will have the option to push an advertisement depending on the chance that a customer will respond. For instance, if a customer just ate at a restaurant downtown, there is 40% chance that they will get drinks when they are done. This information will be available to merchants so a bar can set their account to push an advertisement whenever a customer makes a purchase at a restaurant downtown. In this situation, there is a very low chance that the customer will eat again, so another restaurant will probably not push an advertisement to this customer. Related to this, advertisements must be pushed to customers with minimal time lag to make sure that they have not left the area before they receive the advertisement.

Of the data stored by MCM, we will use the customer transaction history to determine purchasing patterns. The algorithms we implement will produce statistics for merchants. The pattern recognition algorithms will analyze past customer transactions in order to predict future purchases.  Merchants will be able to look at the probability of a certain customer making a purchase from them and determine whether or not they want to push an advertisement to that customer. The localization functionality requires additional user settings, such as types of advertisements or coupons they are accepting on their account.

System Impact

Our additions will make MCM a more comprehensive and robust commerce application. Therefore, our sponsor will have a better product to pitch, potentially making his job easier.  The development team of MCM will be able to utilize localization activity code from the Android and iOS applications and use them to further their development of the applications.

By using MCM with localization features, there will be fewer steps in the money exchange, and thus transactions will be faster and lines will move faster. Theft may also decrease because there would be less cash to steal, making it not worth the effort and consequences.

With the added localization functionality, MCM becomes a comprehensive commerce application. Many other applications focus on certain aspects of commerce but none so far combine them. MCM will have the potential of removing the need for physical money entirely because you will be able to move money between customers, and between customers and merchants regardless of distance. Merchants will also be able to advertise more effectively, and customers will be able to tailor their account for advertisements and coupons that they are interested in.

**Process Overview**

Team Roles

Team Leader:                        Kimberly Oyama

The team leader will be the main contact with between the team and Dr. Georgas, the team mentor. This person will keep track of the project tasks (who they are assigned to, when they are due, and their progress) and maintain communication lines between team members in case they aren’t keeping each other in the loop. This person will handle any conflicts that arise throughout the course of the project, with the exception of version conflicts which are the responsibility of the two programmers involved.

Customer Communicator: Kimberly Oyama

The customer communicator is in charge of making sure the customer is receiving the information they need about the project in a professional manner. This person will relay any information they receive from the customer to the rest of the team (the team leader will forward pertinent information to Dr. Georgas). When the team has information or questions for the client, it is the communicator’s job to format the message according to the client’s preference and clearly express the team’s intentions.

Recorder:                            Daren Rodhouse

The recorder will take notes during the meetings. This job may fall on more than one person because it is difficult to engage in conversation and take notes at the same time. This will be done by a pair of members so that each can be involved and keep track of what the other contributes, such as main recorder and assistant. This pair can change for each meeting. One of the pair will be assigned the agenda for the next week’s meeting and the other will be responsible for compiling the minutes and sharing them with the other team members, which includes adding them to the repository and sending a notification email.

Architect:                                Blayne Kennedy

The architect will make sure that our requirements document matches the client’s intended system. He will also make sure that, when we start implementing the system, it matches our architectural decisions made in the design phase.

Release Manager:                 Chihiro Sasaki

The release manager will make sure that we are submitting the correct versions as deliverables in class and to our client. This person will keep track of the control logs so we will know what builds are clean.   It is also the release manager’s responsibility to review commit logs for consistency and accuracy.  In the case that a release or commit breaks the system, it is the responsibility of the release manager to identify the problem and work with the responsible programmer to fix the issue as soon as it arises.

**Requirements --** Start with an intro paragraph that describes your requirements acquisition process, i.e., how you went about extracting requirements from your sponsor. Interviews? Site visits? etc. Then introduce and summarize the main requirements. Allude to the requirements doc in the notebook for details.

* Overall Goals: overview statement of what, in general, the application is expected to accomplish
* Requirements: List and briefly describe the major functional and performance requirements. Doesn't have to be as detailed as your requirements doc, just give highlights. So you could/should condense groups of detailed requirements into larger scale ones.
* Constraints: Any constraints the (a) the client put you under or (b) you determined would limit development in your problem analysis.

In the Functional Requirements section, the essential requirements are listed for the iOS/Android applications and the web service.  Each of the requirements for different pieces of the system is further divided down into users and merchants, the two main stakeholders of the system.

Next, the Environments Requirements section outlines some of the constraints and environmental requirements to be considered in the system design.  This includes limitations from the existing infrastructure, considerations about the content filtering system for advertisement creators, Google API limitations for using Google Maps, language considerations, and device operating system platform considerations.

The Non-Functional Requirements section introduces the major non-functional requirement considerations to the system.  This includes accessibility, expandability, maintainability, and usability concerns that are important to the usefulness and further development of the MCM system.  Each of the non-functional requirements is accompanied by a description, the condition in which they are valid, and a threshold of verifiability.

In the Potential Risks Section, the important risks pertaining to the localization aspect of MCM are covered.  They include any potential risks that affect any of the stakeholders involved in a relevant manner, along with their corresponding impact and possible mitigations.  The risks include security of localization integration, privacy protection interference to geolocation, the potential of merchants pushing too many advertisements and coupons to the point of annoyance, loss of cell service, inaccurate geolocation, failure of the MCM server, and the potential of user mobile devices recognizing more than one merchant in its proximity.

**Solution Statement ---** Here is where you describe what you proposed to build for the client. It starts with a general overview and then gives increasing detail in subsequent sections.

* Overall Solution -- Describe the overall solution that was proposed to the client. Feel free to include a screen snapshot or two from the final product, anything to help us visualize, at a high level, what you were planning to build. Should leave us with a solid notion of what your solution looks like and does.
* Functional Specifications --- Summarize the central functional specs. These should directly reflect the requirements that you described earlier. Same comments as for requirements apply, i.e., you may condense groups of detailed functional specs into single more abstract specs --- you don't want the whole huge detailed list here, just a solid grasp of the function you promised to provide.

* Architecture Overview --- Outline the architecture that you developed to implement the spec. This is the most important "implementation" section in the report --- without a good comprehension of this high-level design, anything you say later about low level details will be meaningless. Carefully diagram major modules and describe responsibilities of each

Overall System Architecture

The entire system consists of two types of user interactions: customer and merchant. Each interaction is supported by three different platforms, iOS 5 running on an iOS device, Jelly Bean 4.2 running on an Android device, and a website. Each of the six interfaces, as seen in Figure 1, connects to web services on the server, which then interact with the database.  The interfaces do not directly interact with each other; the web services acts as an intermediary among different interfaces.

Figure 1: Overall system with its submodules and their interactions.

The four mobile device interfaces are placed external to the web server.  The two website interfaces interact internally within the web server since the web services and the website live on the same server.

As seen in Figure 1, the main data flow between mobile devices and the web service happens through an HTTP request, where data passed is packaged in a JSON string.  The customer and merchant website components also interact with the web service through an HTTP request.  The web services interact with these interfaces in a RESTful way, using only POST, GET, PUT, and DELETE operations.

* As-built Design (all together) --- Here's where you report on exactly what you produced. Doesn't have to be as detailed as your as-built report (which should allow your successors to understand your design well enough to repair/extend it). Rather, it should give a technical reader a solid idea of how you implemented the design. Go through each module, give an implementional overview of it, then discuss the various services it provides, and how these services contribute to overall behavior (i.e. when/how/who called).

**Usability Testing and Future work** --- Describe your usability testing and outcomes

* Outline the tests you ran: subjects, method, materials
* Describe the outcomes of the tests: what did you learn? Might separate into "little" and "big" shortcomings/insights.
* Describe what you did about the exposed issues. Fixed it? Put it on the docket for version2?
* Summarize the usability: Did you create a usable application?

**Conclusion** --- Wrap it all up. Just general post-mortem on the whole project. Who did it go? Where the sponsors happy? Are you satisfied? This is NOT a place to reflect on your educational experience or personal growth; this is a professional document --- I want to know about the project you did as *consultants*, not about what you learned as students.