Maroon and Gold: Mobile App

By

Justin Reigel

APPROVED:

Ajay Bansal Director

Timothy Lindquist Second Committee Member

Maroon and Gold: Mobile Application

Justin Reigel

Bachelors in Software Engineering

Polytechnic Campus, Arizona State University

Abstract - Currently, students at Arizona State University are restricted to cards when using their college's local currency. This currency, Maroon and Gold dollars (M&G), is a primary source of meal plans for many students. When relying on card readers, students risk security and convenience. Another major issue is that businesses must purchase external hardware in order to use the M&G System. An online or mobile system could eliminate the need for a physical card and allow businesses to function without external card readers.

Such a system would have access to financial information of businesses and students at ASU. Thus, the system require severe scrutiny by a well-trusted team of professionals before being implemented. My objective was to help bring such a system to life. To do this, I decided to make a mobile application prototype to serve as a baseline and to demonstrate the features of such a system. As a baseline, it needed to have a realistic, professional appearance while accurately demonstrating functionality. Before developing the app, I set out to determine the User Interactions and User Experience designs (UI/UX) by conducting a series of informal interviews with local students and businesses. After the designs were finalized, I started implementation of the actual application in Android Studio.

I. INTRODUCTION

Before I discuss the benefits of integrating a mobile app with the M&G system, I will briefly describe the current system and M&G. M&G is a currency used in the meal plan for many students at Arizona State University. The current system, which I will later refer to as the card-dependent system, makes transactions through dedicated card readers at each cooperating business. These dedicated card readers only accept M&G and are usually adjacent to the credit card readers. To use the dedicated card readers, each student is assigned a card with a unique, constant number. Although the current system is sufficient, the current M&G user faces multiple challenges at the moment. Firstly, the M&G card number is a unique, unchanging number assigned to each student. This means that, while lost cards can be replaced, a student's very sensitive, transactional information is permanently compromised if their card number is discovered by someone else. A user can still access the account by simply reciting the unchanging card number at any M&G accepting business. Secondly, the card must be on the student at all times if they want to make a transaction. That is, unless the student wants to list their card number upon making a transaction. Thirdly, students do not have a convenient method for viewing their account balances in real time. A student must go to a M&G reload machine or view their balance on a receipt after purchases [4]. With this method, even if a student keeps track of their receipts, it is impossible to know their balance before a transaction. If their account is compromised, they may not know until they attempt to make a purchase and discover that their account is empty. This is perhaps the largest drawback to the current card dependent system. Lastly, the hardware requirement is a financial burden to businesses that plan to conduct business with students. While this fee may be relatively small for a large company with a few local chains, it is a real concern for small businesses or mobile vendors. For example, if ASU wants to incorporate transportation into their M&G system, Uber would have to get a card reader in each individual vehicle. Since ASU already imposes a base fee and takes a percentage of all sales, this hardware cost may really limit the number of business partnerships for ASU. For these reasons, I decided to make a prototype for a potential solution for these problems. A mobile app could keep the information bound to a student's unique password that can be changed and liked to their email address. The financial information can be displayed in real time via the app, and businesses could be a part of the M&G system without dedicated card readers. However, this system is not without drawbacks. The two largest risks would be the security risk, if implemented improperly, and the cost to implement the system securely. Another issue is that the mobile application does not fully solve the cumbersome aspect of the card. Instead of carrying a card around, a user will instead carry their cell phone around. While cell phones are perhaps the most commonly carried objects among students, it is still an external object. On top of that, the mobile application is less reliable if their phone runs out of batteries or crashes for an unrelated reason. Taking these aspects into consideration, one could argue that cards are more convenient and reliable to carry. Regardless, a mobile application would be incredibly useful for solving the other three aforementioned challenges faced by the current M&G user.

II. RELATED WORK

Currently, there is no related software application when it comes to M&G. However, the conceptual idea for a mobile application for transactions was inspired by the current mobile trend with restaurants. Having a mobile application to manage orders and conduct transactions is a widely growing trend across the world. While all of the implementations vary in some way or another, they have the same base concept. That is, they can allow users to make remote, card-free orders at local restaurants. This greatly reduces lines and the amount of needed interaction between the employees and customers. Generally, I see that these mobile applications tend to fall into two major functional categories: the one-sided transactional model and the two-sided transactional model. These are not formal models, but rather observations that I have made.

A. ONE-SIDED MODEL:

In the one-sided model, the user is able to complete the entire transaction from their phone. The drawback for this is the lack of confirmation. Since the transactions are at local restaurants, auto-confirmation is a risky system. A restaurant may be too busy, out of certain ingredients, etc. In order to have a confirmed transaction with a one-sided transactional model, the user would have to have to hand the phone over to a cashier for confirmation or have a very thorough system that monitors the availability and menu of each distinct location. This issue is where the two-sided transactional model comes into play. Passing a phone around can be tedious and spread germs unnecessarily. Along with this, a one-sided model that relies on a cashier has a flaw in terms of security. A customer could confirm a transaction from their device and claim it was confirmed by an employee. This could be resolved quickly upon further investigation, but it causes an inconvenience that would not exist in the two-sided model.

B. TWO-SIDED MODEL:

In the two sided model, the employee would have their own device to monitor the incoming orders and accept the transactions with. This makes the two-sided model easier to monitor transaction confirmations with. However, the dedicated device for confirmations can potentially be seen as unnecessary and wasteful. The convenience that the two-sided model creates may not be worth creating a dedicated device for confirming transactions. The one-sided transactional model requires 0 dedicated hardware from the business, aside from the servers, since it simply uses the mobile phones of the customers.

Note: The main deciding factor between these two factors depends on the amount of trust given to the average customer and the budget of the system.

III. BACKGROUND

A. PENCIL PROJECT:

Pencil is an open source, cross-platform, GUI prototyping tool that was created by Evlous [2]. Pencil allows the user to easily create shapes and text boxes that can be linked to other images. This allows the user to quickly and easily create a prototype with basic user interaction. While the Pencil files do not actually handle input or generate images, each image can be linked to other pre-existing images. Thus, Pencil is an incredible tool for creating a basic GUI with limited UI/UX functionality.

The purpose of this basic prototype is to give the developers the freedom to quickly get through the UI/UX interaction phases of the design cycle without having to invest a large amount of time or money into the implementations. This is crucial since designs during the early phases are constantly changed and discarded.

When using Pencil for my initial design phases, I used this GUI tool in order to visualize the appearance and navigation throughout my app. The basic image linking in this tool allowed me to simulate activity stack navigation by clicking on the different buttons from my Home Screen. The final Pencil prototype that I created allowed the user to demo navigation from the Home Screen into the Purchase, Transaction, and Confirm transaction screens. The actual functionality of these screens could not be implemented using Pencil, but it was sufficient for explaining the basic design and layout for the application. If users had difficulties understanding the purpose of a page that did not have functionality built in, I took it as a need to make the design more intuitive.

B. ANDROID STUDIO:

Android Studio, based on IntelliJ IDEA, is an IDE for developing and testing Android applications [3]. This IDE neatly manages the java classes, manifest files, and other resources for the Android application. The build system is based on Gradle and the IDE comes with built in templates for Android applications. This IDE is very intuitive and makes it easy to visualize/test the app, code with required modules, and share the code.

In my experience, the emulator was very slow because of my limited hardware. Essentially, I had to do all of the testing on my local device. Luckily, I had a device that fit the requirements for my target release. It was also very easy to design the individual app activities through the xml editor. This is due to the design tab with the xml files. I could enter basic xml code and see the results of the design in the design tab. It also allows the developer to simply drag and drop design elements and it will automatically generate the appropriate xml code. This flexibility and real time display made it extremely easy to test the front end without even running the application on my emulator or local device.

C. SQLITE:

The whole purpose of the application is to demonstrate the potential functionality with ASU's database. In my application, I used a simple database in order to simulate this functionality. I did this by using SQLite, a local SQL database engine [1]. Since it is a local database, the whole purpose of my database is just for demonstration purposes. The database in my application does not persist or exist outside of the local device. However, SQLite demonstrates the database relationships and the queries that I would use with a real database.

SQLite is open source and easy to use and is easy to implement with Android Studio. I simply imported SQLite and then created the tables/queries required for my database. From there, I call upon the queries in various things like signing up, making transactions, and viewing purchases.

D. DATABASE RELATIONSHIPS:

SQLite is incredibly useful for self-contained databases, but it still does not do all of the work. It is just a useful tool for implementing the database relationships that I specify. In my application, I had 3 tables for handling my data (see fig. 3-1). The Users table, which is used for registering accounts and logging in. The Proposed Transactions table, which is used for monitoring the transactions that the user has requested. After a user submits an amount of money to a specified business, a row is added to the Proposed Transactions table. If the transaction is approved by the business, it will be moved to the Completed Purchases table and time-stamped. If the request is rejected, it will simply be removed from the Proposed Transactions table.

Each user can have 0-* Proposed Transactions and Completed Purchases (as represented by the 0< arrow notation next to these tables). However, each proposed transaction and completed purchase can have one and only one user (denoted by the || notation next to the users table). This means that every user has their distinct set of proposed transactions and completed purchases.

The Proposed Transactions and Completed Purchases relationship is sort of unique. Technically, a completed purchase in my system is just a proposed transaction that has been accepted. Thus, one proposed transaction can only correspond to one and only one completed purchase. However, after a completed purchase is generated from a proposed transaction, that proposed transaction is deleted. Because of this, the transaction id is not persisted either.

E. USE CASE DIAGRAM:

The use case diagram for this mobile application, created in Microsoft Visio [5], (see fig. 3-2) consists of three actors, one subsystem, and five use cases. In this diagram, the mobile application is the subsystem that all of the actors access. The customer, on the top left, is the core user for this application. The customer is able to make transactions, confirm transactions, and review purchase history. Each of these actions first requires the user to login, which may require the user to register an account if they do not currently have an account.

The actor on the bottom left, the employee, is responsible for confirming a transaction with the customer. If both the customer and the employee agree to a transaction, then the database will attempt to process the transaction.

The actor on the right, the database, is responsible for handling the backend of the system. That is, is provides the customer with the data to review history, it accepts incoming transaction offers, and it processes confirmed transactions. This is shown below (see fig. 3-1) Once a transaction is confirmed by the customer and employee, the database will deduct the amount from the customer's account and move the proposed transaction into the completed purchases table.

IV. THE PROPOSED TECHNIQUE

In this section, I will describe the main techniques used when creating the application. That is, I will describe the main conceptual and functional features that relate to the overall purpose of the application. These features include the selected transactional model, the use of real-time account information, and the decision to make the application a native android application instead of a cross-platform application.

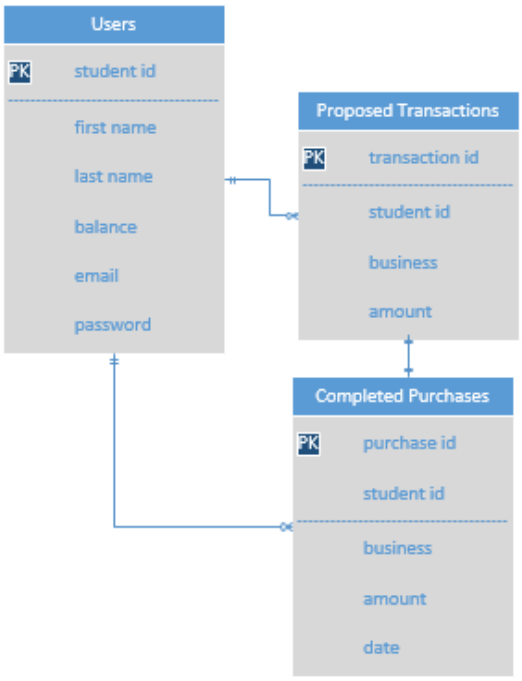

Fig. 3-1 Database Relationships

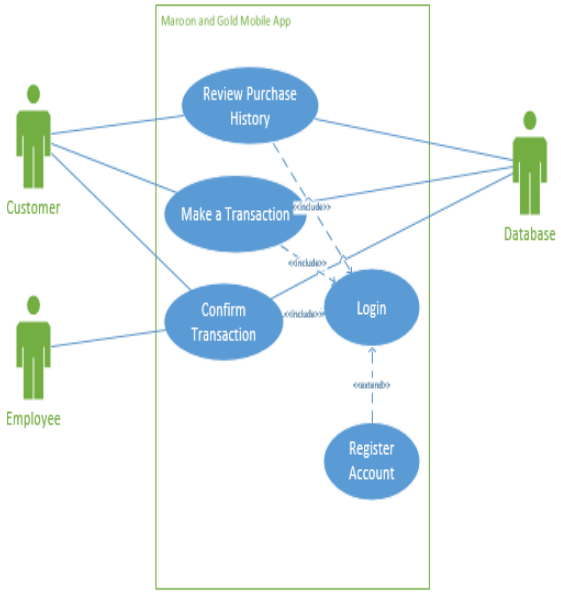

Fig. 3-2 Use Case Diagram

A. ONE-SIDED TRANSACTIONAL MODEL

At first, I had created a two-sided transactional model in my Pencil prototype. Upon further review, it was decided that the two-sided transactional model was not the most viable option for this product. One major reason for this product is that it would allow local businesses to easily conduct business with students. By making the application a two-sided transactional model, it would just be swapping a dedicated card reader for a dedicated transactional client. Thus, small businesses and mobile vendors would still be burdened by the M&G system. Along with this, it would require more development to split the users into two accounts: customers, and employees.

The one-sided transactional model allows every user to be classified as a customer that can easily complete transactions from their phone only. This removes the burden of a dedicated client for the businesses and allows the employees to only have individual accounts. The "business accounts" will instead be created in the server and not through the application itself. Since they will have to go through ASU to have their business implemented, it will also prevent unwanted businesses from registering an account through the system. The previous idea was to have the businesses only register if they have a business id. However, if they have such an id, they are already registered in the system. For these reasons, the two-sided transaction model seems needlessly cumbersome and costly.

The previous problem still remains, that the user can confirm the transaction on behalf of the business. However, this would be easily discovered by reviewing security footage and questioning the local employees. On top of this, the user would simply lose money from their account through the deception. Thus, I believe the users can be trusted to get consent of the business before making a transaction, as it is mutually beneficial.

B. REAL- TIME ACCOUNT INFORMATION

Another key concept of the mobile app is to allow the users to view their balance and account info in real time. As discussed in the introduction, viewing your balance via receipt is tedious and unreliable. By viewing their account balance in real time, users will be able to immediately see if their account information has been compromised. This gives them the opportunity to immediately act and regain control over their account.

For implementation of this feature, I placed a user info section in the bottom left of each screen in the

application after the user has logged in. This section contains the user's student id, name, and account balance. The balance can be updated at any specified time interval. It is also updated upon logging in/out or making any transactions. If the user wants to be certain that their account balance has been updated, they can immediately re-log or make a transaction and reject it. The balance is updated upon each transaction request so the user knows exactly how much they have in their account before they confirm the transaction and have their balance deducted.

The user info section was placed in every activity after login since this information is relevant in every feature of the application. By making queries to the database upon time intervals, transaction requests, and logging in, the user can essentially guaranteed that they have the stated balance at that exact moment. However, there is error handling in place in case of last second balance changes. This is because it would be too resource intensive to update from the database every millisecond. Instead, it is real time for all practical purposes.

C. NATIVE APPLICATION

After learning the various pros/cons of native applications, a native application was shown to be the best implementation. This is due to the future of the application itself. If this application does get implemented, it needs to be secure and professional. From a design perspective, it is much safer (although it requires more resources) to implement the application as a native application for each device. If it were to be cross platform, certain views and features may not be translated properly between applications. This is particularly useful when using specified table views and leaving room for additional features.

V. IMPLEMENTATION

Tools: Android Studio, SQLite, Pencil

I used these various tools for specific parts of the application including design, database management, and the mobile application. During the first phase, I used Pencil to generate the basic design of the screens. Pencil was the tool of choice here because it allows for a light, quick prototype that demonstrates UI/UX designs. The lightness of the prototype was very important when trying to finalize the general design. The design in this phase was changed many times as I reviewed local students and businesses.

After implementing user feedback and finalizing a design, I used Android Studio in pair with SQLite to turn the Pencil design into a mobile application. Android Studio is what was essentially used to create the entire Android mobile application. SQLite was used for the sample database in order to show basic functionality including: users, transaction offers, and purchases.

VI. SUMMARY AND FUTURE WORK

This application is a minimum viable prototype connected to a local database in order to demonstrate the functionality of such an app. In order for the application to actually be implemented, it would require permission from ASU's administration. This is due to the sensitive information related to the financial accounts of the students. If this application is approved, it will require a trusted security team to connect the application to ASU's actual M&G database.

If the application is integrated with ASU's database, there are a few features that would be very helpful. When making a transaction, a user could select an item from a pre-populated menu. Along with this, a feature to add M&G from a credit card would also be very useful. These are nice extra features, but I do not think they would be relevant for the MVP. Instead, they would be extra features to implement if the project is supported.

ACKNOWLEDGMENT

I would like to thank the faculty members and students that have helped make this creative project possible. The primary director Dr. Ajay Bansal, was tremendously helpful throughout the entire app. From conceptual design and motivational support, to database experience and project guidance, he was a major help throughout the project. The second reader, Dr. Tim Lindquist, provided invaluable help with the mobile aspect of this project. Without his guidance, I wouldn't even know the importance of native applications over cross-platform applications. I would also like to thank Dr. Kevin Gary, whose classes taught me many of the specific tools and concepts used in this project. Finally, I like to thank all of the students and local businesses that participated in the early phases of design development, whom greatly helped improve the UI and UX of the application. I greatly appreciate all of the help in making this creative project.

REFERENCES

[1] SQLite: https://www.sqlite.org/

[2] Pencil: http://pencil.evolus.vn/

[3] Android Studio: http://developer.android.com/tools/studio/index.html

[4] M&G FAQs: http://sundevildining.asu.edu/tempe/mealplans

[5] Microsoft Visio: https://www.microsoft.com/en-us/evalcenter/evaluate-visio-professional-2016