Final Report

Utah Transit Authority

One Stop Application

Group 1145-A

IS 6596-004 Summer 2021

Caleb Henney (u1344440)

Brian Truong (u0701557)

Executive Summary

Utah Transit Authority's (UTA) main objective for the project is to create an application that can provide all needed transportation information to their customers on any mobile platform. Customers will have the ability to view a dashboard that provides transportation information and actions such as plan their trip, locate a transportation vehicle, access UTA customer services, purchase tickets, and view trip history. UTA wants to make this a one stop shop to enhance the overall experience for all customers, hence the name, One Stop App. Due to such a large scope and a limited timeframe, the team agreed to focus on certain interactions the customer can have with the application. These include basic app interfacing, logging in, and creating an account. This report outlines our project overview, results, lessons learned, and next steps for the project itself.

Project Overview

Utah Transit Authority's (UTA) vision is to provide an integrated system of innovative, accessible and efficient public transportation services that increase access to opportunities and contribute to a healthy environment for the people of the Wasatch region. They provide service to over 80 percent of Utah's population, making it one of the geographically-largest transit agencies in the country. With customers utilizing more than 44 million trips every year, UTA's main objective for the project is to have customers be connected to enhance ride experience by providing users the ability to view a dashboard that shows customer all the data related to customer transportations actions such as trip planning, locating a transportation vehicle, accessing UTA customer services, purchasing tickets, viewing trip history and more. Customers are currently required to use multiple platforms to obtain all needed transportation information.

The team consists of two Master of Science in Information Systems (MSIS) candidates, Caleb Henney and Brian Truong, the sponsor, Edison Pascascio, and our faculty advisor, John Degrey. Both candidates derive from different backgrounds with coding experiences gained through the academic program. The team agreed to have both individuals take on the role of project managers and developers due to the restraint of time and team size.

The project began with the candidates meeting with the sponsor and their team members to obtain a better understanding of the company, objective of the project, and the creation of the statement of work. With the large project scope, limited time frame and members, the team had to decide what tasks and deliverables would be most appropriate to establish for the foundation of the project. This resulted in the decision of focusing on a basic Android application skeleton with limited functionality that will be continuously built upon by future teams. The team's goal is to allow customers to create and manage their account through the application, have a login page that will

authenticate login credentials with the database, encrypt passwords for security purposes, view a basic dashboard, and view electronic fare card (EFC) and trip information. This will be done primarily using technologies such as Microsoft Visual Studio and Microsoft Azure DevOps for coding and repository, which includes the coding language C# for scripting, Xamarin for mobile implementation, and Microsoft SQL Management for the database.

Communication between the team and sponsor will be conducted through Slack and Microsoft Teams. The work will be performed in two-week sprints with bi-weekly meetings to review activities from previous sprints and discuss upcoming sprint and milestone progress.

Results

The results of the project are a working skeleton application with functional login page, dashboard page, and base sign up page.

The following is a video link to team's finished deliverables: Demo

The project focuses on multi-platform development with a focus on Android services. The code will be usable with multiple operating systems. The original statement of work (SOW) focused on basic user interactions within the app, such as creating usernames/passwords, editing customer information, links to payment information, and Electronic Fee Collection (EFC) services. The scope shrunk twice during the course of creation. The initial scope shrink occurred near the start of the semester with payment information. Our sponsor informed us that we would not be allowed to handle payment information for security risks, so it was scrapped from the project. The second scope shrink happened near the end of the semester after the meeting with the technologist in residence (TIR). By that point, the team had successfully created the login page and dashboard skeleton, so we decided to cut all aspects of EFC interaction. The sponsor agreed to this decision and the project went forward without it. One factor of the scope that did not change was function over form. The team avoided complicated user interfaces due to team size and time constraints.

The main challenge of the deliverables was to learn each of the technologies from nothing. The main resource the team had to aid in learning was Google. Countless message boards, blogs, articles, and videos were scavenged to grow the team's knowledge of the technologies that would be involved. The technologies used on the project included Azure Devops, Visual Studio, Xamarin, Microsoft SQL Management Studio, and C#.

Structure:

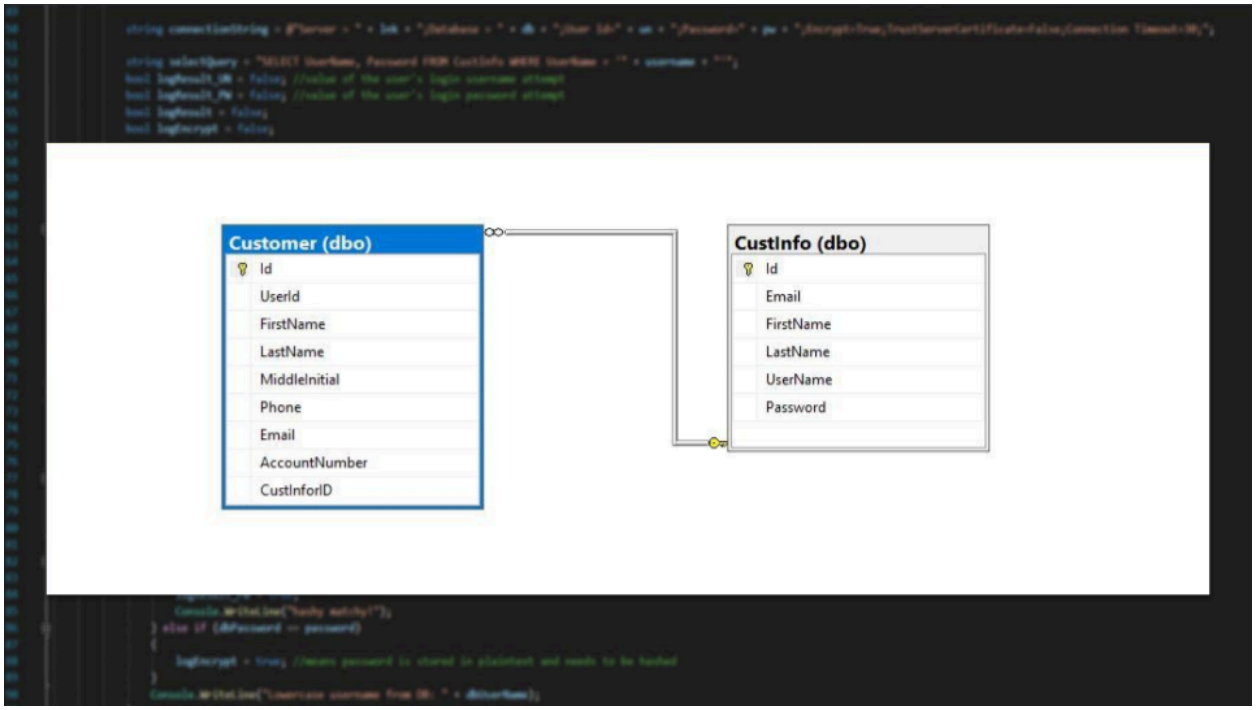
The main structure would be how app users would navigate the application and how each page would flow into the other. The main goal was to establish the functional skeleton for future teams to expand upon. This goal led to the dashboard page.The dashboard has space for user details, EFC summary, payment information, and settings. At this point, the application has little functionality beyond the login page. The main challenge of this section was just how to structure/write the application. Eventually, our team was able to find a series of videos and blog posts outlining best practices and strategies for creating an online application. The team followed these tutorials and created the basic structure: A login page (without authentication/encryption) and an empty dashboard. The next task would be to flesh out the login page.

Login Page:

The first hurdle of the application process was login authentication. The plan to complete this task was the same as in the SOW: upon login, check the database for the credentials entered, verify matches and allow the user to login if their credentials were in the database. The main technical challenge to this process would be to have the C# code communicate with the SQL database.

Database:

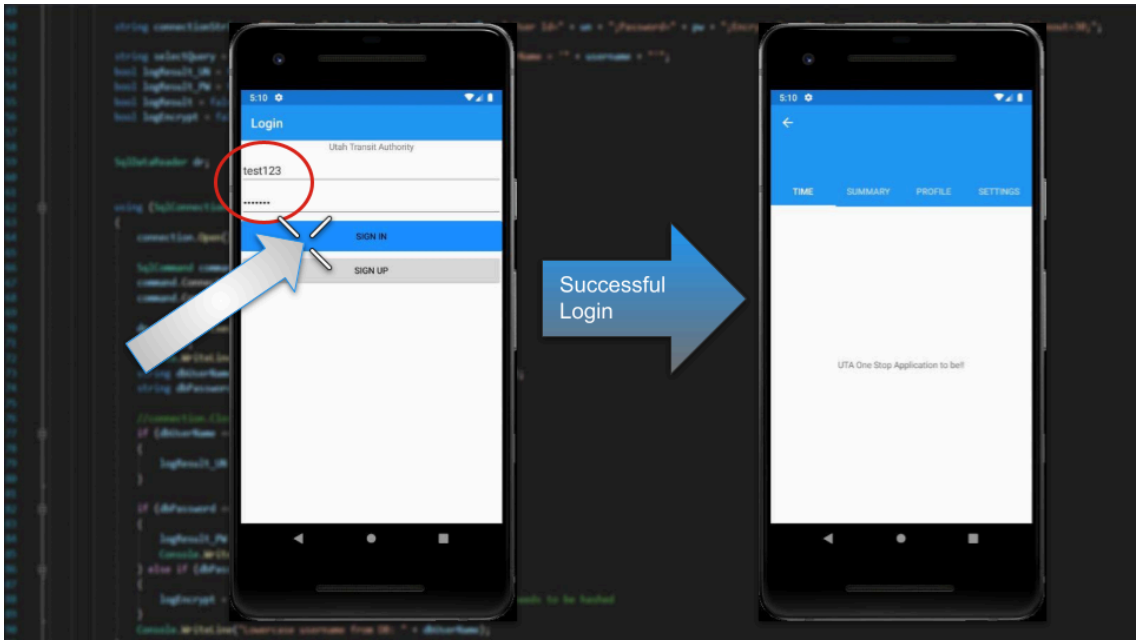
The structure of the database, like the overall plan, was to be simple. The database would be two tables: one to interact directly with the application (CustInfo) and one to retrieve UTA customer information from their interactions with the company's technology (Customer), the latter of which would be expanded on by future teams.



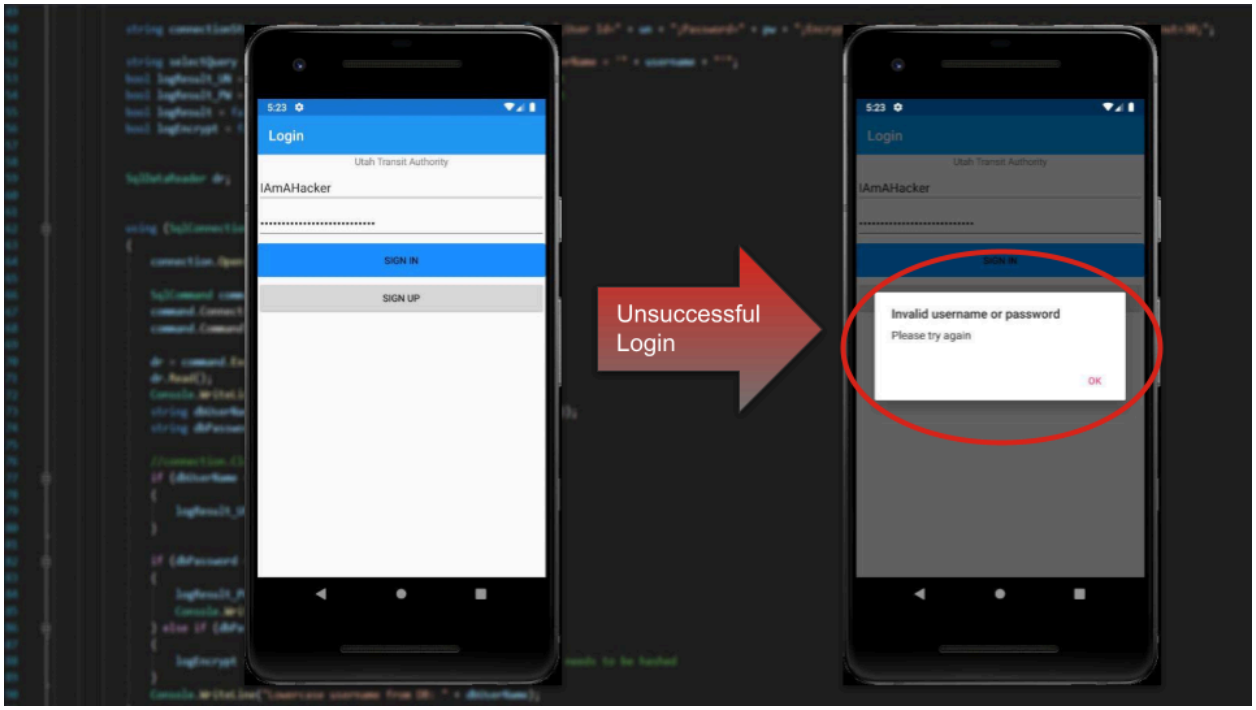
The application communicates with a very simple database that can be expanded on by future teams.

Login Page (Contd.)

With the database in place, the project could continue. The hurdle of having C# and SQL communication was still present, so the team focused on this challenge. SQLClient was initially used to facilitate this communication. An entire week was dedicated to learning SQLClient syntax, testing the database interactions, and conquering its quirks. However troubles arose when moving SQLClient syntax from a web-based dummy application to the mobile-based Xamarin application. Upon this discovery, the team found on a message board that SQLClient would not work with mobile infrastructure. Our team decided to learn SQLite instead, which the message board claimed would work better with the mobile application. This change was decided mere days before one of our bi-weekly sprint meetings and would likely take another week to learn SQLite syntax, putting us a week behind schedule. The team consulted the sponsor and faculty advisor and learned that SQLClient could, in fact, work with Xamarin, so the team moved back. The SQLClient interaction worked and the team could have basic read interaction between the application and the database. The team used this interaction for login authentication. We were back on track.



Upon successful login, the user is taken to the dashboard page.



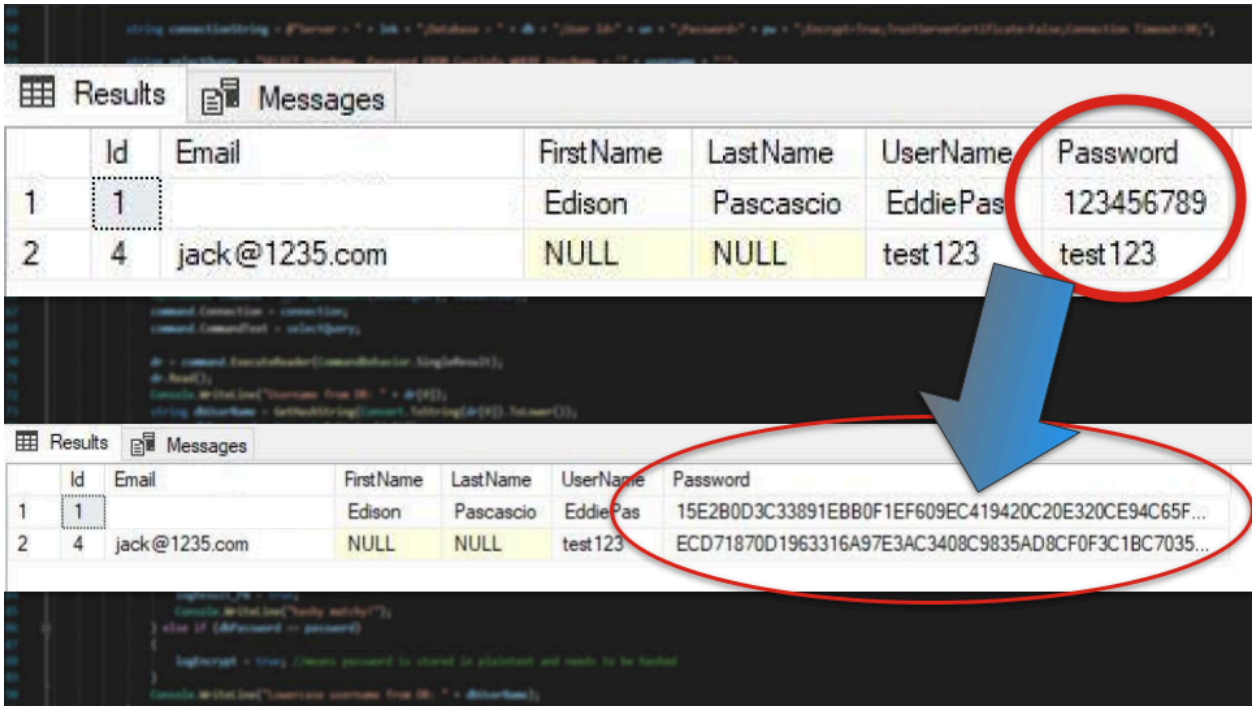
Upon login with invalid credentials, the user is shown an “invalid” message and must enter new credentials.

Login Page (Contd.)

With this new interaction, the team could communicate with the database in the app. The next challenge would be to encrypt user credentials.

Encryption:

At this point in the application creation, the database stored passwords in plaintext, so the user passwords could be easily accessible to potential hackers. The plan for this step in production would be to design the login page to check for encrypted passwords and, upon finding a plaintext password (that matches user-entered credentials), re-storing the passwords as encrypted passwords. The application would be able to read encrypted passwords' hashes, but not decrypt the passwords themselves to create some form of security. The challenges to this process were to find functions to hash the entered passwords, insert the encrypted passwords to the database, and read the encrypted passwords in the first place.



Upon successful login, plaintext passwords that are stored in the database are hashed.

```
string dbUserName = Convert.ToString(dr[0]).ToLower();
string dbPassword = Convert.ToString(dr[1]);
if (dbUserName == username.ToLower()) //Usernames are not case sensitive
{
    logResult_UN = true;
}
if (dbPassword == GetHashString(password)) //passwords are case sensitive
{
    logResult_PW = true;
}
if (dbPassword == password)
{
    logResult_PW = true; //if the password is stored in plaintext (and both still match), the app will still let the user in
    logEncrypt = true; //but it will also go on to encrypt the plaintext password
}
connection.Close();
if (logResult_UN == true && logResult_PW == true) //if both username and password match, then the user is logged in
{
    logResult = true;
}
```

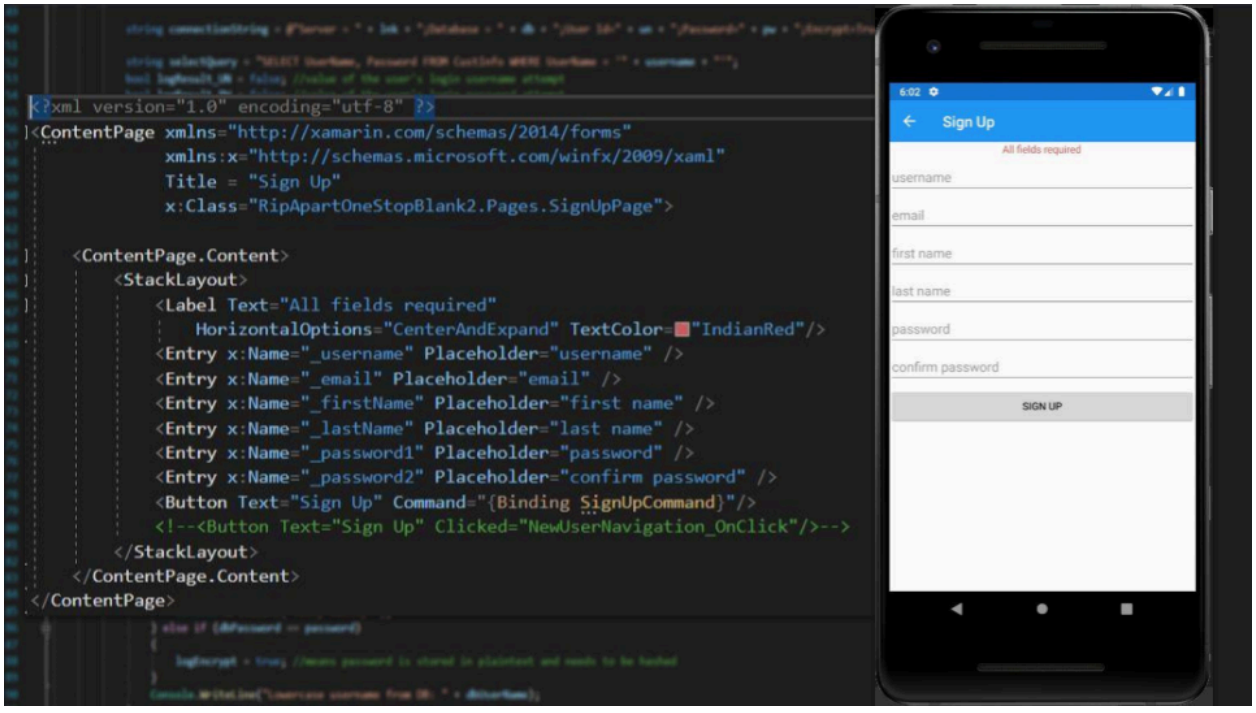
Sample of the login validation process.

Encryption (Contd.)

Thankfully, the challenges of reading/writing hashes were accomplished by finding two functions that would hash and read strings. The process of moving the functions from the dummy app to Xamarin was relatively painless. The team reached a minor hiccup when inserting the new, encrypted passwords into the database. The initial thought was to use an INSERT statement with the SQL database, but that method returned nothing but errors. Upon further investigation, the team discovered that UPDATE statements would work perfectly with our code. The team did have some concerns with the process's security as we are not security experts. We advised the sponsor to double check the security of this process. The problem of unencrypted passwords was fixed, the end of the project was nearly upon us and our next task was the sign up page.

Sign Up Page:

The objective behind the sign up page was to have the user add their credentials to the database. Encryption would be automatically included on the password, the user would then be sent to the login screen to login with their new account. The team ran into a few snags with this section. The first issue was having XAML communicate with the separate C# pages, the second was having information entered in the application registered and moldable within C#, and the third was struggling with our own application framework. Each of these problems would unfortunately go unsolved as UTA was in the process of transferring their DevOps infrastructure from a government organization to a business organization and the team lost access to the servers around the time to practice the capstone presentation. At present, the page is basic text input with a functionless sign up button, little more than a XAML page. The development of this page will have to be continued by future capstone teams.



The sign up page is currently non-functional and only displays the page seen here.

Lessons Learned

Learning C# and XAML:

While the team spent several months at the beginning of the project dedicated to learning C#, the team was largely caught off guard by XAML, Azure, and Visual Studio. The future team will have to look further ahead to find out each technology that could/should be used in the project. The team can even ask the sponsor what technologies they expect to be used on the project in order to have a better idea of what challenges they will face.

Database Access:

As mentioned in results, the team had trouble accessing the database through C#. SQLClient was practiced and failed in the mobile environment, the team decided to switch to SQLite just to find out that SQLClient worked fine with the mobile environment. To counter this, the team will go straight to sponsor and faculty advisor first to counteract this flip-flop effect.

From Government to Business:

With UTA changing their Azure environment from government to business, the team lost access to the coding servers temporarily at the very end of the project. Sometimes problems occur and the team plans to include measures for a loss of access in the SOW in the future.

Scope Management:

With our new knowledge of the technology and work involved, the team has learned to better wrangle expectations and evaluate team capability/bandwidth. With this in mind, the team can better wrangle the scope of future projects.

Next Steps

Before:

At time of writing, all code presented has been uploaded to Azure DevOps. Before UTA switched server structure the team had completed the login page, the skeleton app, and the sign up page. Should UTA require another code push, the team is able to do so.

Next:

The immediate steps for the team are to meet on August eleventh and discuss where the project is, where it will be in future, as well as present finished code to UTA higher management.

After:

After the current team has finished all other steps, the project will be taken on board by future MSIS students for their capstone projects. Their goal will be to build on the foundation that previous teams have created and work toward the application that UTA has envisioned.
