TOKEN MANAGEMENT SYSTEM

OMAR GASTELUM – BRANDY YANG

# PLANNING

## Problem Statement

## The app will not have any affiliation with a company, but it may serve as a solution to manage tokens for games and other items at an event. The idea came from a volunteer committee member at the Arcadia Festival of Bands. The volunteer committee organizes 2 festivals a year where they have small fair-style games. To play the games, one must provide a specified number of tickets that corresponds to each game. The last two festivals sold out of tickets which lead to the committee adding more games and supplying more tickets. They are also considering using the tickets to sell food and other items.

The problem at hand is that there is no way to determine how many more tickets will be sold at the next festival, especially if more games and other items like food will be added. Will doubling the number of physical tickets suffice to meet the demand? Or will fewer tickets be sold at the next festival?

The goal of the project is to provide flexibility to the number of tickets that will be sold by building a program that will generate tokens that can be loaded to cards based on demand. The cards can then be scanned to play games, and purchase food and other items. The system will have the ability to reload tokens to the cards for further purchases. The system will be built with the C# programming language and run on a Windows system. The database will be either accessed directly from the program or through an API. A hardware solution will be facilitated via a USB QR code scanner and a C# library focused on QR barcode scanners.

The target audience for the project will be anyone using tokens to get access or purchase something. This will work ideally in a fair or festival setting where some users purchase cards and others verify them using a scanner but also at any event with a similar use case.

## Early Visual Concept

Graphical user interface

Description automatically generated

# DESIGN

## Database

## To manage the relationship between the events, cards, and items for purchase a database design was implemented using UML. The cards and events have a one-to-many relationship whereas the events and items have a many-to-many relationship. The many-to-many relationship was accomplished by creating an event-item join table that holds the ids of both the event and the item that pertains to that event. The card column holds the token quantity and is identified by the number of the card expressed by a barcode.

MySQL was our database of choice since it is very efficient, scalable, and widely supported. Our prior experience with MySQL also made it a good choice for the application.

Diagram

Description automatically generated with medium confidence

## User Interface

## The UI’s final design was done in Visual Studio since its capabilities were new to us.

The main page was first created using panels that includes a navigation bar, a logo, and a dynamic header. Also on the main page, we have custom window controls.

Graphical user interface, application, table

Description automatically generated

The next part included a general layout of the forms and buttons.

Graphical user interface, application, Teams

Description automatically generated

Finally, the design for a page where the user can select the event being managed.

Graphical user interface, application, Teams

Description automatically generated

## Use Cases

## A use case diagram was also created in draw.io. The diagram gave insight into how the application would flow from a user standpoint.

# IMPLEMENTATION

## Database

To implement the database, we used a C# library called MySqlClient. A class was implemented that connects to the database and calls the data reader.

Text

Description automatically generated

## Main UI

## The main UI has a lot of moving parts. First, in the constructor, we set a default button to be active and a default form to be displayed. In this case, it is the add items form. Then a few onClick methods control what form is being displayed based on what navigation item is clicked. They also change what buttons are active and what buttons are not.

Text

Description automatically generated

Graphical user interface, application, Teams

Description automatically generatedText

Description automatically generated

Graphical user interface, application, Teams

Description automatically generatedText

Description automatically generated

## General Functionality of the Forms

## The general functionality of the forms is mostly repeated in all the forms that include CRUD type of functionality. Generally, if a drop-down exists, the items are gathered in the constructor with a setComboBox method. Also in the constructor, the items of the page that the CRUD operations will be performed on are set onto a list. Items can be added to the list by filling out the form and clicking the appropriate button. The same is true for the delete buttons. Error handling is also included into all the forms.

Graphical user interface, application, Teams

Description automatically generatedText

Description automatically generated

Graphical user interface, application, Teams

Description automatically generatedText

Description automatically generated

## Creating Cards and Adding Tokens

Creating cards and adding tokens requires a barcode to be scanned and populated onto a textbox. The number of tokens to be added to the card is then added to another textbox. Clicking the add button will check the database for an existing card. If a card exists, the tokens are added to it, otherwise, a new card is added to the database along with the tokens specified.

Graphical user interface, application, Teams

Description automatically generatedText

Description automatically generated

Graphical user interface, application, Teams

Description automatically generatedText

Description automatically generated

## Transactions with Card Scanner

## In this section, the items added in the add items page are populated in the form of a drop-down. Once an item is selected, the tokens required for purchase are set. When the barcode on the card is scanned, the system checks if the number of tokens in the card is greater than or equal to the number of tokens required. If they are, those tokens are subtracted from the card, otherwise, an error is displayed stating that the tokens are insufficient.

Graphical user interface, application

Description automatically generatedText

Description automatically generated

Graphical user interface, application

Description automatically generatedText

Description automatically generated

# EVALUATION

## User Evaluation

## Aside from the testing done by us, it should be tested in production. The system needs to be tested during a real event to ensure that all components work properly. Some of the tasks that would need to be performed in a live environment would include all the CRUD operations for events, cards, and items. Items would also need to be added to an event. Performing transactions by barcode scanner would also be required during the evaluation.

## Existing Application Comparison

## The system can be compared to Dave & Buster’s card/token management system. Their system can be distributed to each individual game without having attendants scanning cards. Their cards also manage the tickets earned from playing the games which can then be cashed out for prizes. Although our system does not have those capabilities, they can easily be implemented.

# CONCLUSION

The system can be extended by including a payment system that could be used to purchase tokens to add to the cards. The system can also use the functionality of tracking tickets like Dave & Buster’s system.