**Music Matters Booking System**

**Architectural Spike Report**

Authors :

Chase Dumbacher

Sarah Pham

Austin Mongold

Austin Newkirk

Submitted in partial fulfillment of the requirements for COMP 4710 Senior Design to the Department of Computer Science and Software Engineering,Samuel Ginn College of Engineering, Auburn University

Auburn, Alabama

January 31,2021

**Table Of Contents**

System Metaphor………………………………………………………..3

Cycle Intent ……………………………………………………………..3

Design Decision …………………………………………………….......3

User Stories …………………………………………………………..4-8

Design Documentation.……………………………………………...….9

Lesson Learned ………………………………………………………..10

Test Documentation …………………………………………………...10

**System Metaphor**

*Completed By Chase Dumbacher*

To design and create an app and online calendar booking system that is capable of recording and organizing artist names, performance time and dates, and venue names as well as automatically creating and emailing invoices and confirmation documents.

**Cycle Intent**

*Completed By Austin Newkirk*

The Intent of this cycle was to familiarize ourselves with the previous work laid out by the teams before us, communicate with our sponsor Mike Moody, and begin the foundations of our work. After having many conversations with Mr. Moody, we decided to focus up on the computer application side of the project, as the iOS application is mostly completed, little work has happened on the computer side, and the importance of this computer side for our sponsor. After negotiations and considerations, we’ve decided to create a website (likely using .NET Framework) using the database from the iOS, as this will likely be easier and take up less time.

**Design Decisions**

*Completed By Sarah Pham*

Upon meeting with our sponsor, Mike, he prioritized how important it was that the application be usable on his computer. A major design flaw that Mike noticed was that he was not able to view his monthly calendar’s events all at once. Which led to the conclusion that his phone was too small to view his bookings and therefore asking for a desktop version. After speaking to a few group members who had handled this project before us, it was stated that the project was not able to be written/reformatted for a computer version. However, after much research it was found that the task could potentially be accomplished but it would be tedious and could lead to more problems. Therefore, our group decided to start from scratch and create the website portion using C# and connecting to the same backend database as the app. There have also been a few bugs found within the app whenever the app was being reviewed by a previous group member. So currently we are researching about creating the website and potentially fixing the major bugs within the app.

**User Stories**

**Functionality**

Database Data Sync-ability

*Completed By Chase Dumbacher*

**User Story :** As a user, I want to see a change made in the iOS app change data

Live on the Firebase database.

**Description :** When an artist or venue or any form of such data is changed, said data will be sent to the Firebase data and change the entry there.

**Task :** Connect or assure that the program connects the data run by the app

Is connected to the firebase database.

**Summary :**

**Planned Hours :** 1

**Actual Hours :**

**Coder(s) :** N/A

**Tester(s) :** Chase Dumbacher

**Status :** Completed

Website Dummy

*Completed By Chase Dumbacher*

**User Story :** As a user, I want to have a basic website setup and accessible.

**Description :** A website will be accessible through a users local computer and will

Be used as an example and starting point for the envisioned finished website.

**Task :** Create a website to be run locally through ASP.NET

**Summary :**

**Planned Hours :** 4

**Actual Hours :**

**Coder(s) :**

**Tester(s) :**

**Status :** Not Started

**Usability**

Cross Platform Compatibility

*Completed By Chase Dumbacher*

**User Story :** As a user, I want to see the system be usable from both iOS and PC devices.

**Description :** If a user wishes to view the system calendar on their PC they can open a website or application, but if the user wishes to view the system calendar they can open it on their iOS device.

**Task :** Create a web based interface that is capable of taking in the Firebase data and displaying it in a simple non-complete fashion.

**Summary :**

**Planned Hours : 8**

**Actual Hours :**

**Coder(s) : N/A**

**Tester(s) : N/A**

**Status :** Researching/Planning

Ease Of View

*Completed By Chase Dumbacher*

**User Story :** As a user, I want to see the calendar displayed by the system look similar to the vertex42 or google calendars displays.

**Description :** The calendar for the booking system on the PC application will big big enough to read clearly and will display artist names and performance times without having to click to expand.

**Task :**

**Summary :**

**Planned Hours :** 10

**Actual Hours :**

**Coder(s) :**

**Tester(s) :**

**Status :** Not Started

**Reliability**  
 Database Security

*Completed By Chase Dumbacher*

**User Story :** As a user, I want the database to be read and write safe.

**Description :** The database data will only be readable and writable if the user has the specific roles required

**Task :** Change the rules in the firebase database to require specific roles for

Read and write access

**Summary :**

**Planned Hours :**

**Actual Hours :** 1

**Coder(s) :** Chase Dumbacher

**Tester(s) :**

**Status :** Pending

Data Transfer Reliability

*Completed By Austin Mongold*

**User Story :** As a user, I want to see both applications consistently be able to communicate with the database without errors as often as possible.

**Description :** The applications should perform as intended (read/write) and work consistently without crashing (and if there is an error, a restart should fix it)

**Task :** Make sure both apps accurately send/retrieve data from the database,

and any possible errors can be fixed by an app restart.

**Summary :**

**Planned Hours : 16**

**Actual Hours :**

**Coder(s) :**

**Tester(s) :**

**Status :** Not Started

**Performance**

Quick Load Times

*Completed By Austin Mongold*

**User Story :** As a user, I want to see the applications run smoothly as often as possible.

**Description :** The apps should always perform smooth, and loading times should be kept at an absolute minimum.

**Task :** Find ways to improve load times, such as having a local database

whenever the user logs in for each application, and only uploading changes

to the online database.

**Summary :**

**Planned Hours : 6**

**Actual Hours :**

**Coder(s) :**

**Tester(s) :**

**Status :** Not Started  
  
  
  
Fast and Clean User Experience

*Completed By Austin Mongold*

**User Story :** As a user, I want to see the applications be as snappy, intuitive, smooth and responsive as possible.

**Description :** Work on things that make the apps feel intuitive and professional as well as making it all feel quick and responsive.

**Task :** Get the aforementioned areas of the app to perform as quick and

responsive as possible, but also make them look appealing and user

friendly.

**Summary :**

**Planned Hours : 4**

**Actual Hours :**

**Coder(s) :**

**Tester(s) :**

**Status :** Not Started

**Supportability**

Offline Support

*Completed By Austin Mongold*

**User Story :** As a user, I want the mobile app to have some offline support.

**Description :** The program should have limited functionality in the case of internet loss.

**Task :** Add functions like viewing last saved data and queueing data when

internet connection is regained as well as some other features.

**Summary :**

**Planned Hours : 4**

**Actual Hours :**

**Coder(s) :**

**Tester(s) :**

**Status :** Not Started

Error Checking/Recovery

*Completed By Austin Mongold*

**User Story :** As a user, I want the applications to have the ability to fix simple errors.

**Description :** The program should have quick options to error check and correct.

**Task :** Add a reset/refresh option in case of a simple bug, and add other

similar features to possibly combat other unforeseen bugs.

**Summary :**

**Planned Hours : 4**

**Actual Hours :**

**Coder(s) :**

**Tester(s) :**

**Status :** Not Started

**Design Documentation**

*Completed By Sarah Pham*

**Architecture**

Upon discussing the project architecture with some members of the last group, it was found that our iOS app is written through React Native using the .jsx language whilst using Google Firebase for the backend database. The sponsor, Mike, also wanted a desktop application. Keeping this in mind and after much research, it was concluded that React Native is capable of web and ios development simultaneously, however that task would be tedious and lead to more issues. Therefore (as mentioned in design decisions), we decided to create our own website using C# which will also connect to the same Google Firebase as the app. There also may be possible UI changes when creating the web app in comparison to the iOS app. The design of the iOS app seems to be very simple and in a prototype stage since the previous groups were focusing more on functionality than user interface. We will try to mimic the design of the app but will also try to improve on mostly the calendar portion where the user can view all events occurring in a month.

**Structure**

Whilst looking at the React Native project on Visual Studio Code (and having no experience), it would appear as the following. There are 5 main folders within the project that are .expo, assets, components, functions/handlers, and views. The expo folder enables the project to be compiled and built on a browser (by Expo) to be run on a simulator (iOS, Android, web browser etc). The local host also allows for the app to be downloaded through a qr-code for any devices. The assets folder contains all the graphics used on the app. Components contain a few javascript files that are composed of clickable items, such as buttons or drop downs.The functions folder is where a good portion of the internal code is found. There is also a handlers folder within functions which handles the emails, pdfs, and any associations with the calendar. The functions folder also contains the excel exports and the backend database Firebase. Last but not least, the views folder contains any UI related code.

We are also still researching how to create a website using C# and how to connect that to our current Firebase.

**Lessons Learned**

*Completed By Austin Newkirk*

We, as a group, have learned multiple lessons from this experience. It goes as followed:

1. We learned how important it is to have constant and detailed conversations with our sponsor (or customer), as well as ourselves. Without this near daily conversations, our workflow would be drastically different and inefficient. I can see this being an extremely important lesson for our futures, especially in the COVID work-at-home world we live in now.
2. Doing large amounts of research with unfamiliar aspects of already developed code saved us a lot of time. This project heavily uses libraries/aspects that are either difficult to get working, or straight up impossible to get working on multiple operating systems like iOS and Windows. Instead of diving head-first into the project and figuring this out as we go, we can make better, more informed decisions earlier without writing code. These lessons we covered in classes like Software Quality Assurance and Software Process, but real world experience really does show this importance.

**Test Documentation**

*Completed By Austin Newkirk*

TBA - **If we get a website homepage up soon, this will be edited.**

**If not, this will be it:**

Because of the nature of the cycle, there is very little to be tested. We made sure the app could run on iOS through an emulator, and it did indeed work: