Boston University

Department of Electrical & Computer Engineering

amadeus

First Prototype Testing Report

Team 7

Dev Bhatia - devb@bu.edu

Brandon Desiata - desiata@bu.edu

Ryuichi Ohhata - ryu74@bu.edu

James Wasson - jwasson@bu.edu

Samantha Au - samau@bu.edu

November 21, 2021

Contents

Chapter 1

Required Materials

Being a primarily software project, Amadeus will be demonstrated on a laptop with a database server running in the background. Beyond a laptop to run the application, the required software is as follows:

- Node.JS - Backend
- REACT - Frontend
- Expo - Mobile Conversion
- PostgresQL - Database
- Git - Repository and Version Control

Additionally, we have set up a backend server for the data pipeline such that the data can be accessed remotely. The demo today was running locally on the demo machine, but the remote server will be used in the near future.

Chapter 2

Demonstration Set Up

2.1 Set Up

The setup for testing Amadeus is far more simple than many hardware related projects, primarily because everything will be run on an iPhone tethered to a laptop running Node.JS and React. The test could be run equally on an Android phone. The code must be switched to the main branch on Amadeus git repository, which houses the tested/non-experimental code. Once the tether is set up, we can run a simple command to initiate the development server and run it in a mobile target via Expo.

2.2 Pre-Testing Setup Procedure

The setup procedure will be used to ensure that the three central aspects of the application - the frontend, backend, and database - can all send data across each other. The focus of the test will be the data pipeline, as the UI can be built more efficiently with it in place. The procedure will be as follows:

- Git Confirmation
    1. Switch to the Master branch in the Amadeus repository
    2. Perform a **git pull** to ensure that the code version is up to date
- .env and config.js Checks
    1. Ensure that the .env file in the backend contains the correct **DATABASE URL** for the PostgresQL server and **CLIENT ID** for the Google integration
    2. Check the config.js file in the frontend to ensure the **API ENDPOINTS** are accurate to the target database
- Node Package Manager Startup
    1. Perform an **npm install** to ensure that all of the desired packages have been installed
- Database Confirmation
    1. Using PGadmin, check the addresses and port number of the local host

Chapter 3

Testing Procedure and Success Criteria

3.1 Testing Procedure

The testing procedure will be used primarily to demonstrate Amadeus' data pipeline. The flow of the pipeline can be viewed in Figure 3.1.

1. Start the development server by running **npm start** or **expo start**
2. Scan the QR code on an iPhone with the Expo App installed
3. Upon reaching the sign-in screen, click the **Sign In With Google** button
4. The person demonstrating the prototype will sign into their Google account after clicking **continue**
5. Upon landing at the genre selection screen, the user should utilize the picker to select a genre (any will work) and press the **confirm** button
6. This will take the user to the instrument selection screen. The user should enter an instrument (text input) and confirm via the button
7. The user will now land at the birthday selection screen. The user should input a birthdate into the picker module and press confirm
8. Should the previous steps be successful, the user will end at the home screen.

3.2 Measurable Criteria for Success

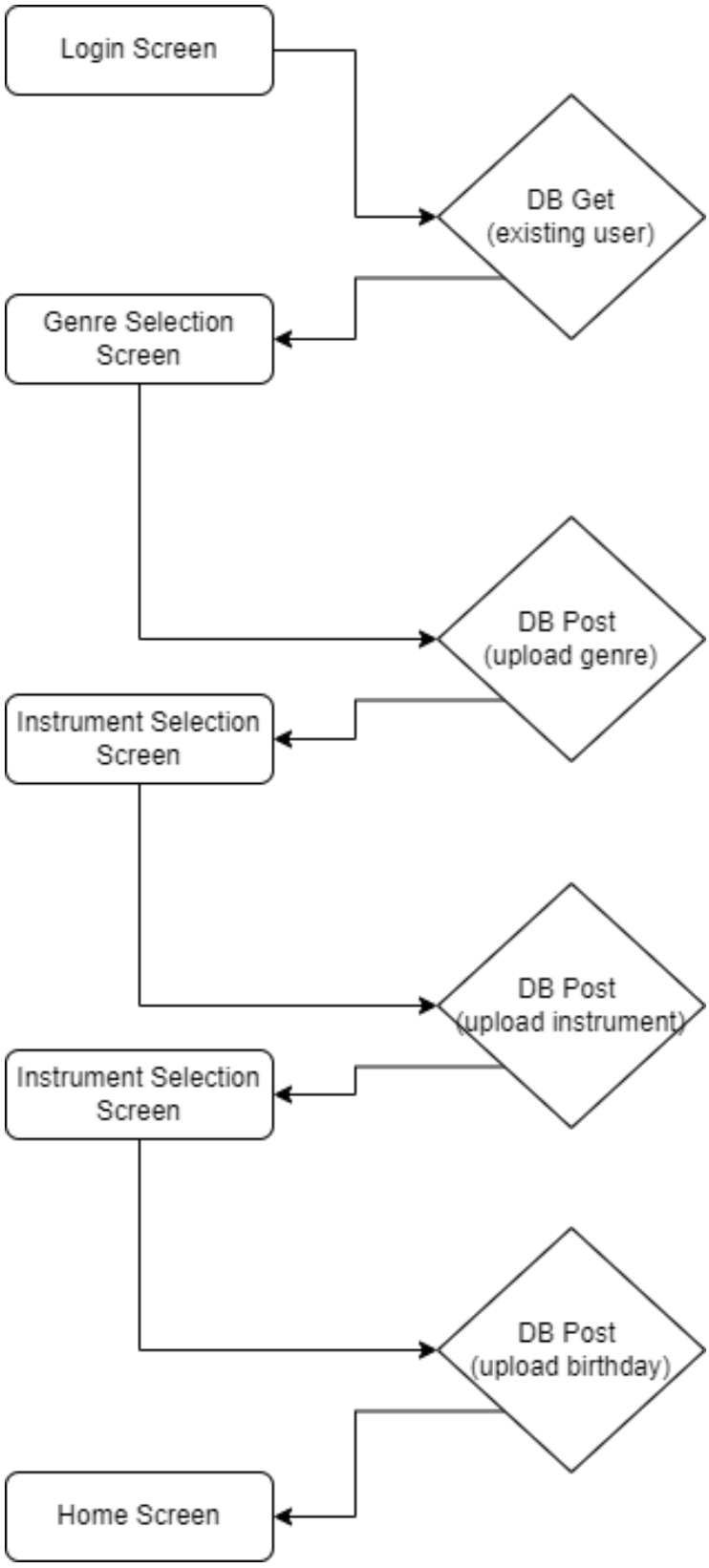
Success will be primarily evaluated in the form of data being successfully being transferred between each screen and the database. Additionally, the app and all demonstrated screens should flow easily and efficiently. More specifically:

1. By step 8 in the testing procedure, the postgresQL database should have been populated with the following fields for a single user:
    - Name
    - Email

- Date of Birth
- Genre
- Instrument
- Picture

2. Every screen (besides the login screen) should display the user's name, pulled from the google authentication
3. The application should compile in the development server without errors
4. The pressing of the "continue" button in each screen should progressively upload the targeted field to the database and shift to the following screen

Figure 3.1: Prototype Data Pipeline



Chapter 4

Testing Results and Conclusions

Seeing as the central goal of the testing was to ensure that the database could communicate with the frontend and backend of the application, the measurements taken revolved around that specifically - effectively yes/no based on if each parameter in the "Measurable Criteria for Success" occurred successfully.

Because every parameter within Section 3.2 happened without issue during the test, we began to plan for the remainder of the semester. With a tested database used in combination with *Axios*, the main goal becomes integration. There are still many disjoint pieces of the project within individual branches of the repository, so with the data pipeline in place the team can commence the process of merging and migrating everything.

With some advice from Professor Hirsch, we confidently believe that the test was an overall success.