Boston University

Department of Electrical & Computer Engineering

amadeus

Second 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

March 17, 2022

Contents

Chapter 1

Required Materials

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

- Python Flask - Backend
- REACT Native + NPM - Frontend
- Expo - Mobile Conversion
- PostgreSQL - Database
- Git - Repository and Version Control

This will be very reminiscent of the requirements from the "First Semester Testing Report." Despite this, major modifications to the core application have been made - many of which will be tested and demonstrated today.

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 Python and React Native. The code must be switched to the main branch on Amadeus git repository, which houses the tested and 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 through Expo - an open-source platform for making universal native apps for Android, iOS, and the web with JavaScript and React Native.

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. Additionally, we will ensure that all of the packages required for the frontend are installed correctly. The procedure will be as follows:

- Codebase Update
    1. Switch to the Main branch in the Amadeus repository
    2. Perform a **git pull** to ensure that the code version is up to date
- Environment Variable Checks
    1. Ensure that the .env file in the backend contains the correct **DATABASE URL** for the PostgreSQL server, **CLIENT ID** for the Google integration, and **JWT SECRET** of the user's choice
- Virtual Environment Creation
    1. Create a virtual environment using **python3 -m venv env**
    2. Activate said environment using **source env/bin/activate**
- Package Installation
    1. Run **pip install -r requirements.txt** to ensure that all of the desired packages have been installed. The entire list of requirements can be installed directly from the text file.

- 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 and main menu flow. The expected program flow is shown in Figure 3.2.

1. Start the backend server by running App.py in the backend API folder
2. Start the frontend server by running **npm start** or **expo start**
3. Scan the QR code on an iPhone with the Expo App installed
4. Upon reaching the sign-in screen, click the **Sign In With Google** button
5. The person demonstrating the prototype will sign into their Google account after clicking **continue**
6. 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
7. This will take the user to the instrument selection screen. The user should enter an instrument (text input) and confirm via the button
8. The user will now land at the birthday selection screen. The user should input a birthdate into the picker module and confirm via the button
9. The user will now be prompted for their current location. The demonstrator will allow their device to pass the privacy check and continue.
10. Should the previous steps be successful, the user will end up at the updated home screen. The default location for the home screen is the **"Home"** screen.
11. Clicking on the "Enter Gig" button on the home screen will allow the user to enter a gig location, genre, name, and description.
12. The user can use the navigation bar at the bottom of the screen to switch to the **"Messages"** screen. At the messages screen, the user can send a test message to one of the established users via the "discover" tab.

13. The user can use the navigation bar at the bottom of the screen to switch to the **"Map"** screen. The user can scroll around the map, and search for a specific location.
14. The user can lastly go to the **"Profile"** screen. The Profile screen currently displays a sample profile which the user can scroll through. Some of the details within the profile should have been pulled from google - including a photo and name.

3.2 Measurable Criteria for Success

Success will be primarily evaluated in the form of data being successfully being transferred between each setup screen and the database. Additionally, we must ensure that each screen can be interacted with in the expected way. Lastly, the app and all demonstrated screens should flow easily and efficiently. To expand on these points:

1. By step 10 in the testing procedure, the postgresQL database should have been populated with the following fields for a single user:
    - ID
    - Name
    - Email
    - Date of Birth
    - Genre
    - Instrument
    - Picture
2. The application should compile in the development server without errors
3. The pressing of the "continue" button in each setup screen should progressively upload the targeted field to the database and shift to the following screen
4. The pressing of each button on the navigation bar should switch to the correct tab
5. The final structure of the database should adhere to figure 3.2
6. Each primary screen of the application should have some level of interactability. More specifically -
    - The **"Home"** screen should allow for the user to select the "Enter Gig" option. Autocomplete that corresponds with google maps location data should appear. Upon entry, the gig's latitude and longitude should appear in the database.
    - The **"Messages"** screen should allow for the user to enter an individual chat and send a message
    - The **"Map"** screen should allow for the user to scroll around a live map and search for locations
    - The **"Profile"** screen should allow for the user to view their Google photo and name.

Figure 3.1: Screen Flow

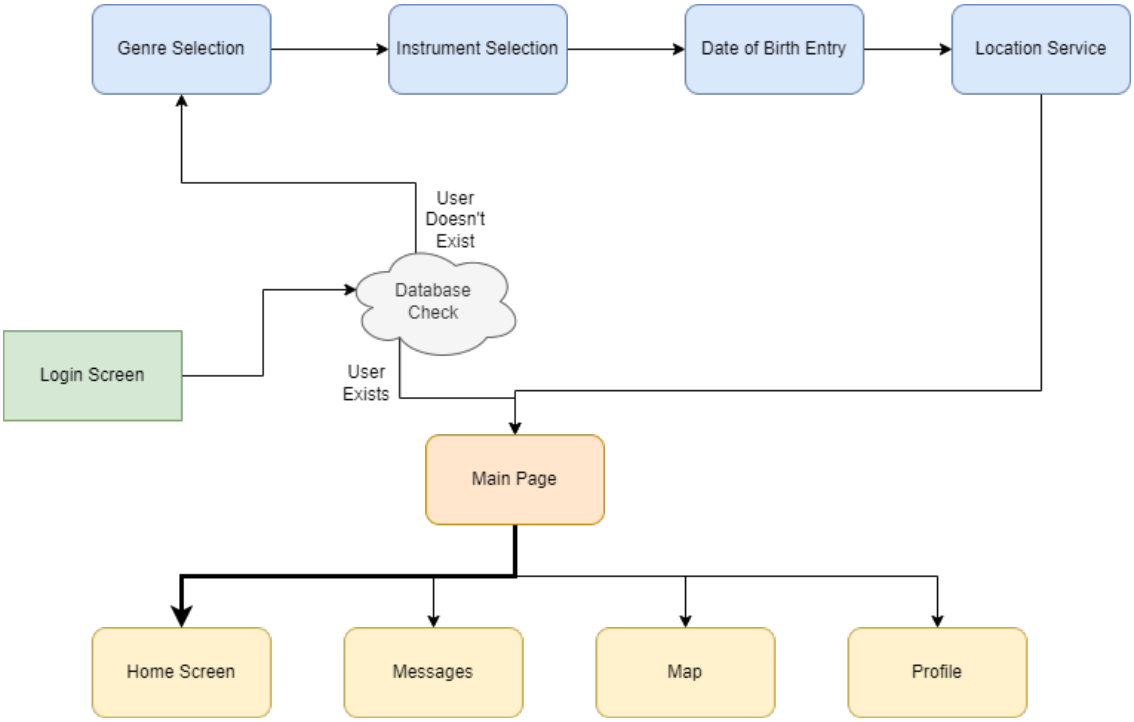
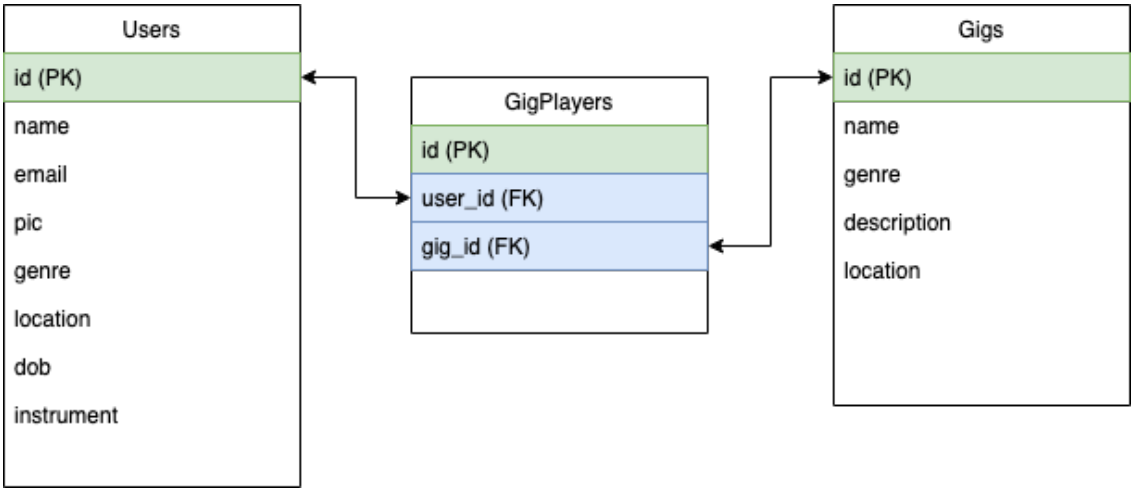


Figure 3.2: Database Table Structure



Chapter 4

Testing Results and Conclusions

The central goal of the Second Prototype Test was to ensure that the core architecture of the application was demonstrable alongside the newly-integrated features of the application. Ensuring that data could be sent across each individual screen - particularly the feature screens - was paramount to this test.

Seeing as we successfully completed every step in Section 3.2 - the Measurable Criteria for Success - we are confident in saying that the prototype test was a success. This is a clear indication that the core system functionalities are in place, most notably the following:

- User creation loop
- Home Page
- Gig creation
- Map Display
- Messaging
- Profile Displaying

Professor Alshaykh conducted our test and provided some helpful notes for proceeding, specifically alluding to our newfound capacity for more rapid development based on the fact that the overarching architecture was fully operational. Moving forward, we plan on following the remaining steps in our GAANT chart in order to finalize the application prior to final prototype testing. First and foremost in this list will be the incorporation of Apache Kafka in order to enable the notification system, followed by containerization and remote hosting. We are confident that we have enough time to finish the application.