



Boston University

Department of Electrical & Computer Engineering

# amadeus

## **Final Project 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

April 6, 2022

# Contents

<b>1</b>	<b>Required Materials</b>	<b>1</b>
<b>2</b>	<b>Demonstration Set Up</b>	<b>2</b>
2.1	Set Up . . . . .	2
2.2	Pre-Testing Setup Procedure . . . . .	2
<b>3</b>	<b>Testing Procedure and Success Criteria</b>	<b>4</b>
3.1	Testing Procedure . . . . .	4
3.2	Measurable Criteria for Success . . . . .	5
<b>4</b>	<b>Testing Results and Conclusions</b>	<b>7</b>

# 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 - Frontend
- Node.JS - Package Management, Runtime Environment
- 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
- Configuration Check
  1. Ensure that that config.js file in the frontend contains the correct endpoints for the API synchronization
- 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 Kafka Enricher and Kafka Consumer
3. Start the frontend server by running **npm start** or **expo start**
4. Scan the QR code on an iPhone with the Expo App installed
5. Upon reaching the sign-in screen, click the **Sign In With Google** button
6. The person demonstrating the prototype will sign into their Google account after clicking **continue**
7. Upon landing at the genre selection screen, the user should utilize the picker to select a genre and press the **confirm** button
8. This will take the user to the instrument selection screen. The user should enter an instrument (text input) and confirm via the button
9. 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
10. The user will now be prompted for their current location. The demonstrator will allow their device to pass the privacy check and continue.
11. 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.
12. The user can now search for another user by name and be taken to their profile. They can subscribe to the user by clicking the appropriate button.
13. Returning to the home screen, clicking on the "Enter Gig" button on the home screen will allow the user to enter a gig location, genre, name, and description.

14. 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.
15. 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.
16. The user can 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.
17. Another user (the one that the demonstrator subscribed to) will create a gig from their device. The demonstrator should have a notification pop up on their device.

## 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 the end of the user creation loop 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. Additionally, the Home screen should allow for the user to search for another user, and subscribe to their profile.
  - 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.

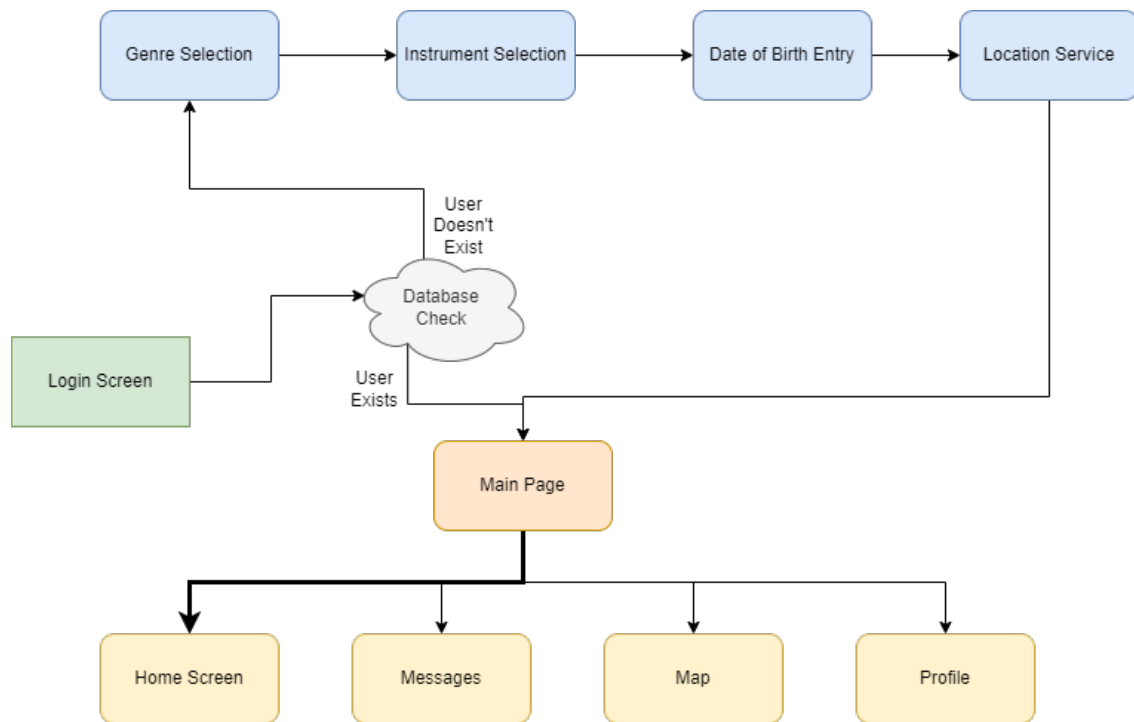


Figure 3.1: Screen Flow

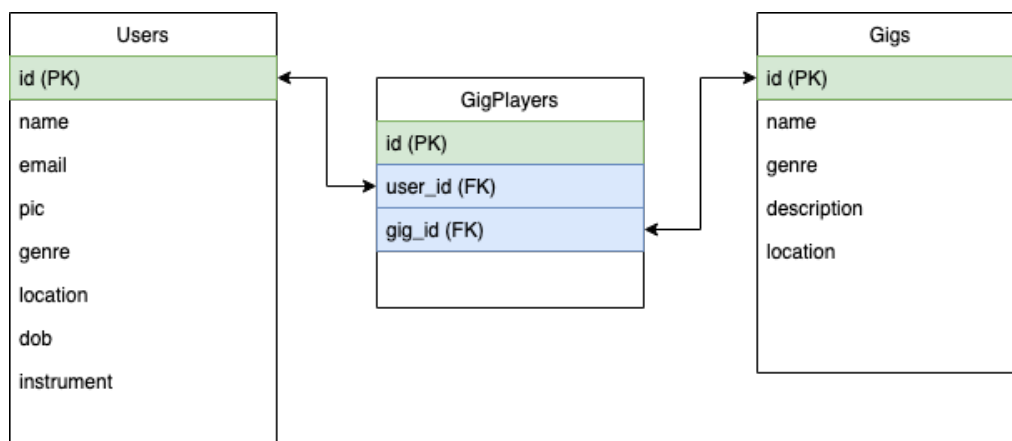


Figure 3.2: Database Table Structure



## Chapter 4

# Testing Results and Conclusions

The central goal of the Final Prototype Test was to demonstrate that the core architecture of the application was fully complete and the functionalities of the application were in place. Ensuring that each feature could operate without breaking the other parts of the application was paramount in this.

Despite some difficulties in the overall merging of the application, we would consider the overall testing session a success, as we were able to display each feature of the application -

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

- in some capacity. We had some difficulty in screen sizing, which is something that has been on our Trello board to fix. We additionally had some minor compilation issues which resulted from a hasty merge that was done immediately prior to testing. The testing party made note that every individual feature worked nonetheless.

Our goal prior to ECE day will primarily center around one last major round of integration and bug testing. We aim to deliver a professional-level, polished application that all iPhone users can interact with live on ECE day. With the core systems in place, this should be a relatively simple effort with all hands on deck.