HeyFriday!

Team Starlords

05/05/2020

Kahlie Hunter, Kylia Hurley, Diego Rodriguez, and Preston Williamson

*We have Abided by the UNCG Academic Integrity Policy. 02/04/2020 *

Table of Contents

1. Introduction

Purpose (*1.1)*

The purpose of team Starlords desktop application; named “HeyFriday!”, is for users to be able to plan and schedule an outing at ease. Whether it be on a date, family event, or just some friends wanting to hang out. “HeyFriday" allows users to find restaurants, movies and other events going on in their area.

Document Conventions (*1.2)*

Intended Audience (*1.3)*

The intended audience for this SRD is the professor grading it, as well as anyone else who is interested.

Definitions/Jargon (*1.4)*

N/A

Project Scope (*1.5)*

Within the scope of this project, we were allowed to use the Java programming language to create a desktop application.

Technical Challenges (*1.6)*

- Connecting and utilizing information from a REST API
- Applying persistent data storage through Apache Derby

References (*1.7)*

For information regarding Apache Derby https://db.apache.org/derby/

Zomato API https://developers.zomato.com/api

Movie DB API https://developers.themoviedb.org/

Eventful API https://api.eventful.com/

2. Project Overview

Product Features (*2.1)*

Our application allows a user to create an account and search for things to do in their area. Such things include events, restaurants, and movies. When searching for a restaurant, a list of cuisine options is generated based on restaurants within a 15 mile radius of the user. Upon selecting a cuisine option a list of restaurants will be generated for the user to choose from. Each restaurant option has basic information regarding the restaurant as well as a clickable link to the Zomato web page for that restaurant. When searching for a movie to watch, the user is given a list of movie genres to choose from. Upon selecting their choice of genres, a list of movies that match their genre choice(s) will be available for them to select from. This allows them to read a description of the movie. When searching for events, the user can choose from a list of different categories, and based on the selected category they will be shown different events in their area that match that category. The user can also keep track of events via the application's calendar feature.

User Characteristics (*2.2)*

The users of HeyFriday ideally are those who are trying to go out but don't know what to do. HeyFriday is designed to solve that problem.

Operating Environment (*2.3)*

The operating environment of HeyFriday is a computer because it is a desktop application.

Design/Implementation Constraints (*2.4)*

Constraints for design/implementation include:

- Must connect to an external API
- Must implement persistent data storage with CRUD operations
- Adhere to MVC architecture

Assumptions/Dependencies (*2.5)*

APIs must be up and running in order for some application features to work.

3. Functional Requirements

Primary Requirement (*3.1)*

The primary features of HeyFriday include:

- The ability to search for a movie to watch by a user chosen genre. This information is pulled from the MovieDB API.
- Being able to find some local events that may be going on in the user's area. This information is pulled from the Eventful API.

- Giving information about local restaurants based on a cuisine chosen by the user. This information is pulled from Zomato API.
- Putting events on a calendar to keep track of things.

Secondary Requirement (*3.2)*

In order to use these features the user must create an account. All user credentials are stored in an embedded database (within the program) powered by Apache Derby. This also means that all user information is specific to the location of where this program is installed.

4. Technical Requirements

Operating Systems (*4.1)*

Because this project is Java based, it should be able to run on any operating system that has a Java virtual machine installed.

Interface Requirements (*4.2)*

User Interface(*4.2.1)*

In order to use the HeyFriday application the user must have a computer with a monitor, keyboard, and mouse.

Hardware Interface (*4.2.2)*

There is no external hardware required for this application to function properly outside those outlined in the user interface.

Software Interface (*4.2.3)*

A computer with Java installed is required to run this application.

Communication Interface (*4.2.4)*

This program connects to four different external APIs which implies that the computer on which this application is running must have an internet connection. The first is the GoogleMaps API which is used to determine a user's latitude and longitude based on a zip code given at the sign up page. The second is the Zomato API which is used to give a list of cuisines and restaurants based on latitude, longitude, and cuisine choice. The third is the Eventful API which is used to list out different categories of events and events themselves. These are based on zip code. The fourth API is the movie api. Which is used to display different movie genres, titles, and information.

5. Non Functional Requirements

Performance Requirements (*5.1)*

There are no strict performance requirements for this application. It should retrieve results from APIs in a reasonable amount of time.

Safety/Recovery Requirements *(5.2)*

There are no safety requirements for this application, and as such, no explicit safety precautions have been taken.

Security Requirements (*5.3)*

Each user account is associated with a username and password. Both of which are chosen by the user upon sign up. The username is the user's email. They must have these credentials in order to

login and use the application. Email and password fields cannot be left blank when signing up, and they have a maximum character limit of 50 each.

Policy Requirements (*5.4)*

All code must adhere to the style guide given by the professor.

Software Quality Attributes (*5.5)*

Availability (*5.5.1)*

The availability of this application is dependent upon the user. It is openly available for use wherever there is a computer with this application downloaded and whenever the user wants to use it.

Correctness (*5.5.2)*

The correctness of this application's information is very important for its general use. It is important that search results for events and restaurants are correctly displayed in regards to the user's zip code. Meaning that the distance of an event or restaurant should be a value that is less than or equal to that of the specified parameters in the program. Else, the user will not go to an event or restaurant if it is too far away. For these reasons, correctness is important for the application's general use.

Maintainability (*5.5.3)*

Mandatory maintenance of this application is minimal. If any bugs are found within the application then there will need to be updates to patch them. But the core functions of this application have been tested prior to its due date. Ideally most of the maintenance will be in the form of features upgrades, where a variety of different features could be added to improve user experience.

Reusability (*5.5.4)*

Different parts of this application have the potential to be used in other projects. The code that works with the Zomato API (interface, translator, adapter) can be used elsewhere if another project needs information from the Zomato API. Right now the interface provides functionality for two types of get request that the Zomato API can fulfill.

Portability(*5.5.5)*

The application is portable because it can be run on any machine that supports Java.

Process Requirements (*5.6)*

Methodology (*5.6.1)*

An agile like methodology was used for this project. All documentation was created after most of the code was written. Different features were added as they were needed/as information was learned during the class.

Time Constraint (*5.6.2)*

The time constraint for creating this project was the duration of the Spring 2020 semester.

Cost/Delivery Date (*5.6.3)*

The delivery date for this project was May 5, 2020, which is the final exam day for the class for which this project was created for. There were no explicit costs for completing this project. The only other cost remotely associated with this project is the cost of attending the class for which this project is required.