Genuine - Design Manual

Professor: Brian King
Product owner: Yuki Yao
Scrum master: Joshua Lee
Developers: Sasha Pisarchik Shketav, Ahmad Rehman

Introduction

Genuine is a mental health app designed to focus on the needs of people who need instant help. The home page consists of a mood tracking page, a habit tracker, and a graph of the user's mood across a certain amount of time. The app contains a navigation bar that allows users to switch between scenes. Currently the app only contains a home page, and a ai chat bot. The ai chat bot uses the chatgpt api for users to have conversations with an ai whenever they please.

User Stories - Damien Fabre

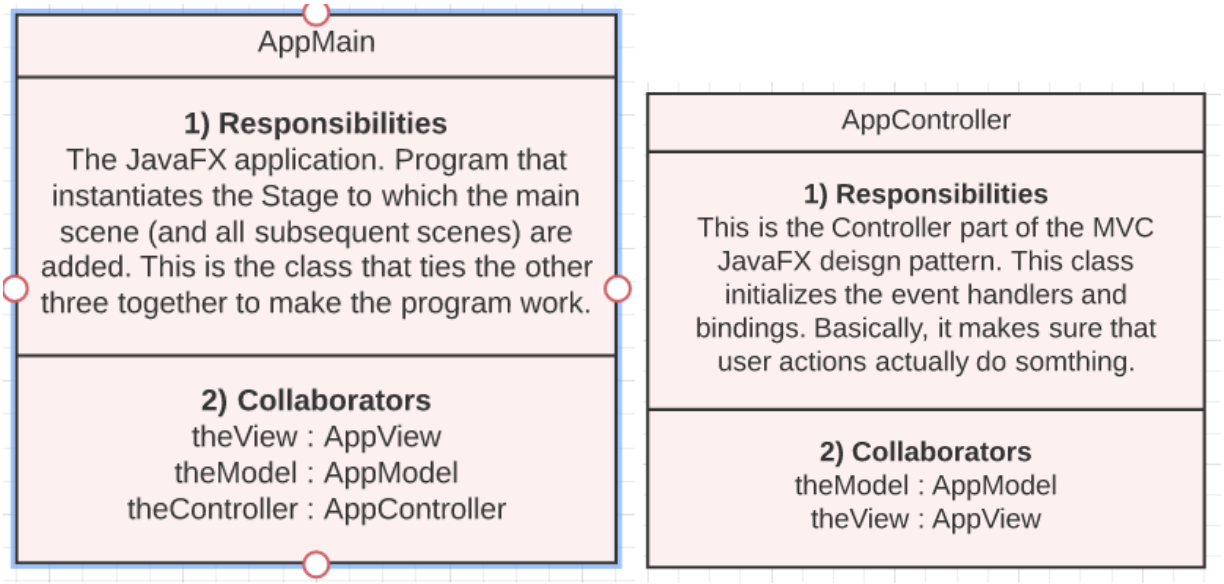
We were able to complete this user story. Damien wanted a system where he can enter the things he did that day that is related to his mental health. Genuine is able to track what your mood is based on the user, and also create custom habits that relate to each specific person. Damien also said that he wants to enter his daily check in. We were able to achieve this and satisfy this user story.

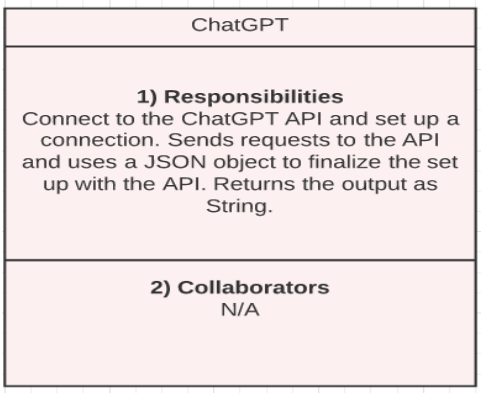
User Stories - Lena Philippe

We were not able to complete this user story. Lena Philippe wanted to improve her mental health. We were not able to track mental health progress, and therefore we were not able to prove if a user's mental health has been improved. Lena also wanted to have access to their own personal account and see how the app can work for themselves. Currently, the app is not able to be accessed by individuals with their own personal account. It can only save information on the users computer if they have the software, but it will not be able to handle personal accounts

OOD

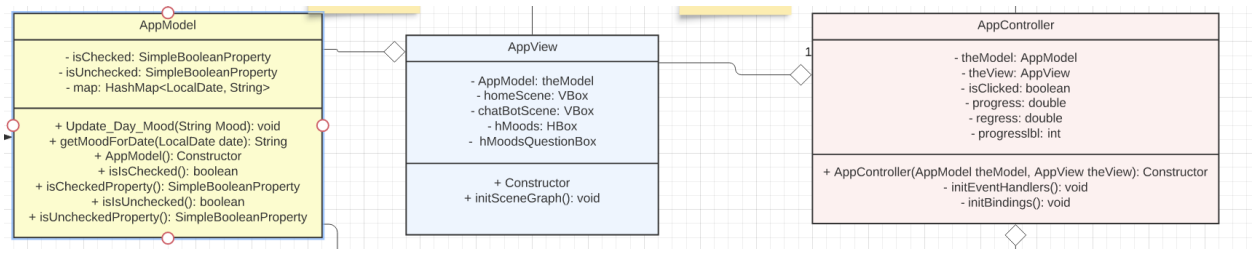
Our program uses 5 main classes. The AppModel, AppController, AppView, AppMain, and ChatGPT. The AppMain is responsible for running the whole program. It sets the scene and adds the main root to the scene. It uses the view, model, and controller. The AppModel is responsible for doing all the work that the controller will use. It handles all the actions and makes it simpler for the controller to use its methods. It has methods that will keep track of each button press and methods that return the key of the hashmap. The AppController is responsible for actually handling all the event handlers that the program will use. It gives the functionality to the button press, habit buttons, chatGPT ai chat bot, and plots the data to the graph. The AppView handles all the visual aspects of the program. The view doesn't use any scene builder and all the visual aspects were hard coded by our developers. The ChatGPT class is used so that we can connect to the ChatGPT API. This allows us to make our ai chat bot. We use chatGPT functionality to simulate a proper ai chat bot.





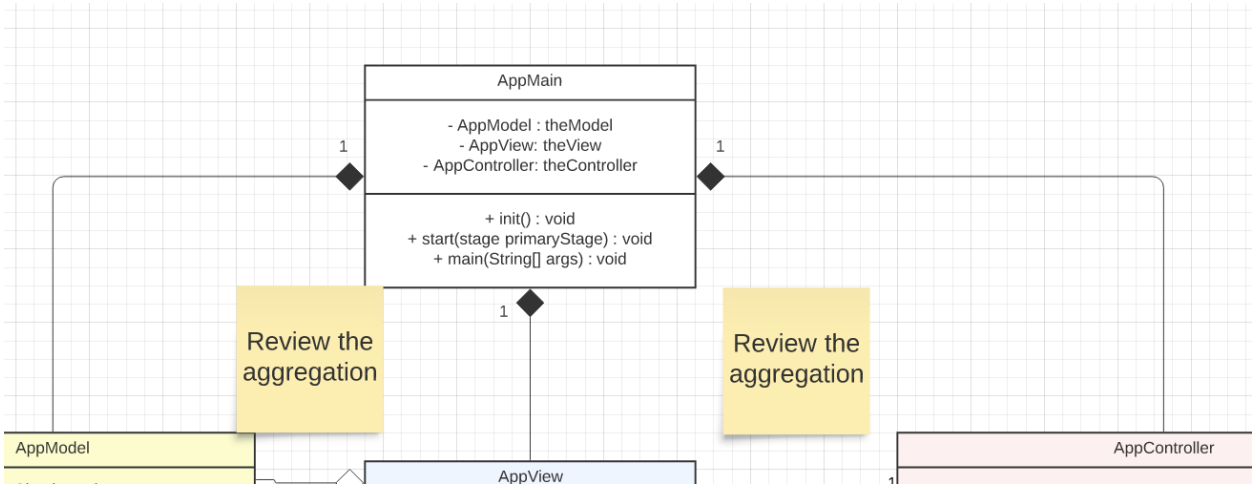
AppModel	AppView
1) Responsibilities This is the Model part of the MVC JavaFX deisgn pattern. This class holds the logic for the app, doing all the necessary calculations and return the desired data.	**1) Responsibilities** This is the View part of the MVC JavaFX deisgn pattern. This class is responsible for all the visual components of the app. This is the actual GUI design. The View contains the different interatables, the layout, and the styling.
2) Collaborators N/A	**2) Collaborators** theModel : AppModel

UML -MVC



Our MVC follows a aggregate relationship. The View creates an object of the model and the controller creates an object of both the view and the model. We believed that aggregation would be the best relationship because these classes need to be shared between each other. We would need instances of these classes between each other and that means that we couldn't create a composition relationship between the MVC design.

UML - main



The main program is how we run our program. The main has three composition relationships. We wanted to go with this design because we believed that if either one of the MVC had a problem, we cannot run the program. Therefore the main handles all the objects and puts them together in order for the program to run. The main creates the main scene and adds the root scene to the stage. It also creates objects of the three MVC in the constructor.

Intellij UML

The images below show our top level classes that intellij had generated. The picture to the left shows the view and the controller relationships. The view has many variables that we used. The image to the right has the rest of our main classes, including the model, main, and the test class that we created.