Cheapskates: Project Documentation

# What is Cheapskates?

Cheapskates is a restaurant locator for restaurants within a specified budget. This app is designed for people with restricted budgets who do not want to spend a lot of money when eating out while still enjoying the benefits. The user specifies a budget and a cuisine and the app will find all restaurants within a 2km radius of the user that are within the budget. Cheapskates uses the Zomato API that has a database of restaurants and their information.

# How does Cheapskates Work?

1. A pop-up box is programmed to appear to allow locations to be used by the program, the user clicks yes if they want to use the app because the app will not work without location services enabled.
2. The program verifies that the location services are allowed and then saves the latitude and longitude of the user in SharedPreferences to access later. If the user does not have a location, the program saves the location as Boston University by default. This prevents the app from crashing in the emulator or if the location services were not allowed.
3. The user is prompted to either enter an amount between 1 and 999 in the text bar and then hit confirm, if they enter a value outside of this range, the program prompts them to try again. They also have the option to press one of three pre-set budget buttons ($10, $25 and $50).
4. Once the user as selected a budget, the program saves the budget into SharedPreferences to access later.
5. Once the budget is created, the program opens another activity where the user is prompted to select the cuisine they would like to see the restaurant for. Each button is equal to the Zomato cuisine id for that category, it is also saved in SharedPreferences for use later.
6. The user now has the option to hit a home button that starts an intent to take them back to the HomeActivity page.
7. Now that the budget and cuisine has been selected, the app begins the search for restaurants, it uses the Zomato API request URL string including the latitude and longitude, the radius which is 2000m, the cuisine id, sorting it by distance in ascending order.
8. Once the connection to Zomato has been made, the information relevant to us thanks to the previous step is obtained using a JSON parsing called Volley.
9. Using the JSON, the program goes through the Zomato array, ignoring all restaurants outside of the budget and putting the restaurant names into a list that is shown in listview on the ListOfRestaurantsActivity page.
10. the listview is displayed and is each element is a clickable button, the user therefore can view the restaurant’s info if they wish by clicking the name of the restaurant.
11. When the user clicks the restaurant name, the JSON object retrieved from the response relevant to that restaurant is saved with the intent to shown the information of the restaurant.
12. Additionally, the user has access to a back button, that starts an intent to take them back to the cuisines page, they can also change their budget from the toolbar, this change budget textbox works just like the textbox on the home page, it stores the budget in SharedPreferences and then starts an intent to go to the loading page.
13. Once the user as selected a restaurant to view the info of, this info is displayed from the list used previously to display the names of the restaurant, it shows the name, address, locality and rating.
14. Finally, the user has the option to press the show in maps button, this button takes the user to another page where a marker is placed at the location of the restaurant in a google maps object, the latitude and longitude of the restaurant are found using the list with restaurant info and is converted into a double to be used in the google maps object. The camera is moved to the marker and set to be zoomed at an 18.