



SYSTEM DESIGN

JohnCena | Sprint 2

Pivotal Tracker

» **Ugandan Knuckles**

TABLE OF CONTENTS



Abstract	3
Environment Interaction	3
System Architecture	3
System Decomposition	3
CRC Model	5
System Architecture	6

Abstract



Environment Interaction

We provide a view for interaction by a single agent—the user. We employ user-input (which may include gestures or text-input) to trigger events such as to query and scrape web-pages. This means that, depending on user input, we can identify which information must be parsed in such a manner as to limit unnecessary processing. We depend on APIs to be available, running, and dependable for our app to work consistently. We chose to use Android Studio to implement this app, which restricts the software to only run on Android-compatible hardware. Some information is stored in Firebase, Android's Database system, so we require Firebase to be available for our app to function as well.

System Architecture

Our system is divided into 3 major parts. This is our MVC (Model, View, Controller). We have visualized models that match our design, and put together interfaces which control how the view displays based on the state of the model. The system architecture diagram on page 6 is the direct product of the MVC pattern.

System Decomposition

Our listener classes act as the controllers that connect the views to the models. The listeners listen for events and adjust the views accordingly. The model is where we query all the data. When we get user input, the model handles the action and finds charity

information. Some of this information includes the charity logo, purpose, social media links, and revenue information. The views are the UI components that the user will see and are the basis of all user interactions. To handle errors, we are setting up error messages with try-catch blocks to predict runtime vulnerability. These messages are meant to be purely instructional so as to explain to the user what the issue is. Aside from system errors, user errors may also arise in our app, such as invalid user input. To deal with this, we have chosen to use the safety net of the API's we are using—which deal with invalid input on our behalf.

CRC Model

MapActivity	
<ul style="list-style-type: none"> • Knows radius around current location • Shows Charities on the map as markers (based on radius) • Acquires charity information 	LocationSuccessListener CameraMoveListener PlaceSelectListener SummaryLogoScraper UrlAndDonationLinkScraper User MarkerClickerListener
LocationSuccessListener	
<ul style="list-style-type: none"> • Gets the current location of the user 	User
CameraMoveListener	
<ul style="list-style-type: none"> • Moves the map based on user gestures 	User
SummaryLogoScraper	
<ul style="list-style-type: none"> • Parse Summary and Logo information for Charity 	
UrlAndDonationLinkScraper	
<ul style="list-style-type: none"> • Parse Donation and Links information for Charity 	
User	
<ul style="list-style-type: none"> • Know personalized account information 	
MarkerClickerListener	

<ul style="list-style-type: none">• Find directions from current position to charities currently displayed on the map	User
---	-------------

System Architecture

The following diagram demonstrates the interconnection of components, subdivided by state of the app in relation to the user-interface.

