Brayden Lomax

Professor Harris

Computing 2

March 26, 2021

**Project Proposal**

Project Title: The Race Team Maintenance Checker

Description:

A maintenance application that operates in the console and checks the status of parts that are on the motorcycle. For this project, I’ll be focusing on dirtbikes but the program could be repurposed to include other types of motorcycles. The program will run maintenance on a dirtbike and come back with things that need to be replaced, checked, tightened, etc. The program will take in the amount of time the rider puts on the machine and that will be what the program uses to run checks to see the status of the part that is being checked. The program will also track the time that’s been logged on the motorcycle as a whole and each individual part. After a part has been replaced the time will be reset back to zero. The user will also have the option to change out parts before the recommended maintenance warning. The race team aspect of the project is where I plan to implement a whole team of bikes that have different attributes and different maintenance schedules for parts. Practice bikes, race bikes, and mod bikes all will have different lengths of time before certain parts need to be replaced. This will offer up a lot of opportunities to track every little part of things that can be a headache to track normally which leads to motocross mechanics being behind on maintenance. This program will allow for optimal performance as the rider and mechanic will know exactly how long a part has been on and how many hours it has on it and whether or not that part needs to be replaced. Parts won’t be the only thing being tracked. Things like oil, filters, bearing grease, etc. can be tracked to add for even deeper tracking. The race team aspect of it where there are many bikes being tracked at one time can help race families and race teams keep track of the numerous bikes that are being used on a typical race weekend.

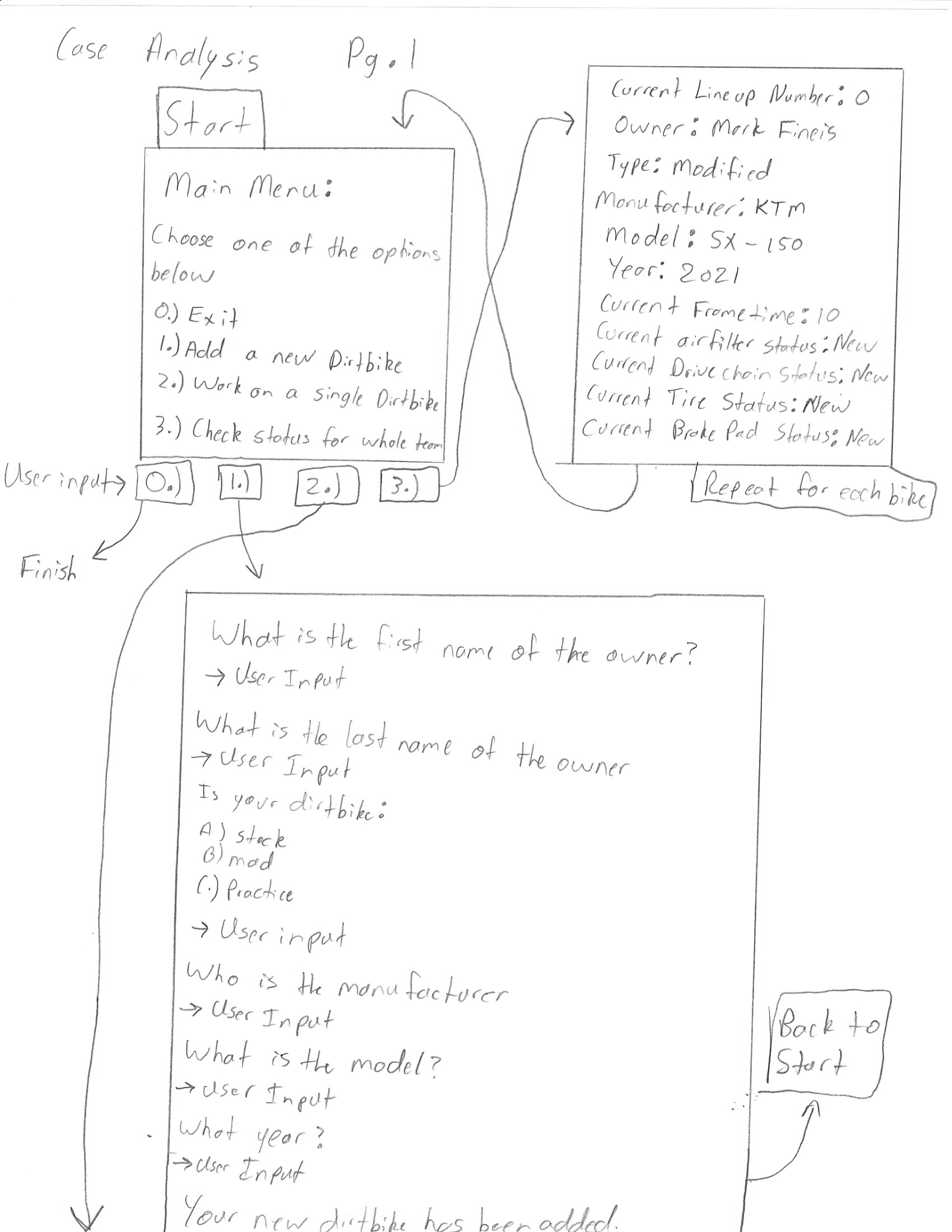
Intended User: Race teams or race families who want to keep track of the status of their machines and the maintenance needed to be done on them.

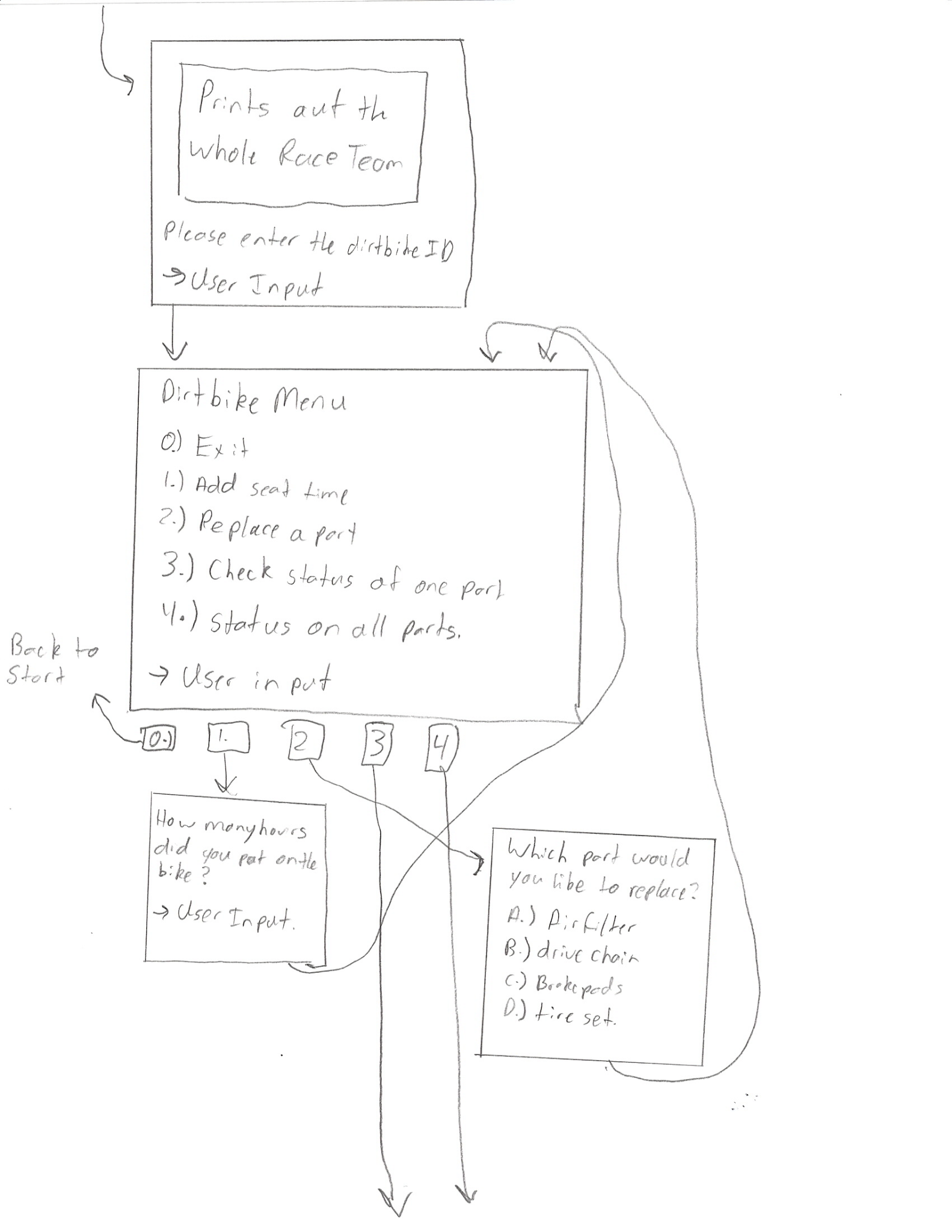
Problem to solve:

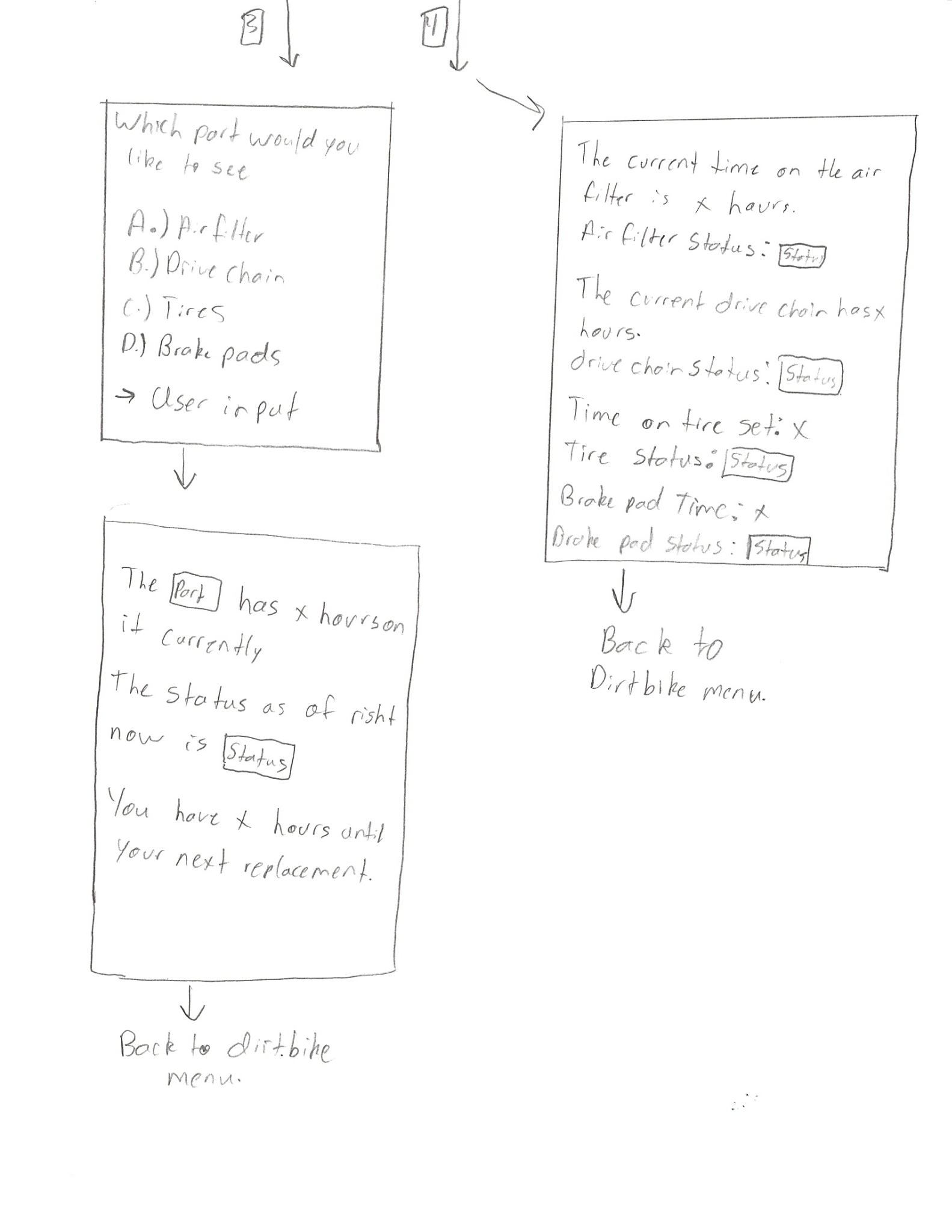
Riders who have maintenance issues because of the lack of maintenance done to their machines because they don't know when certain things need to be replaced and when certain parts need to be looked at.

Tech needed: A text file which will hold the data for each dirtbike being tracked in the program.

Case Analysis:

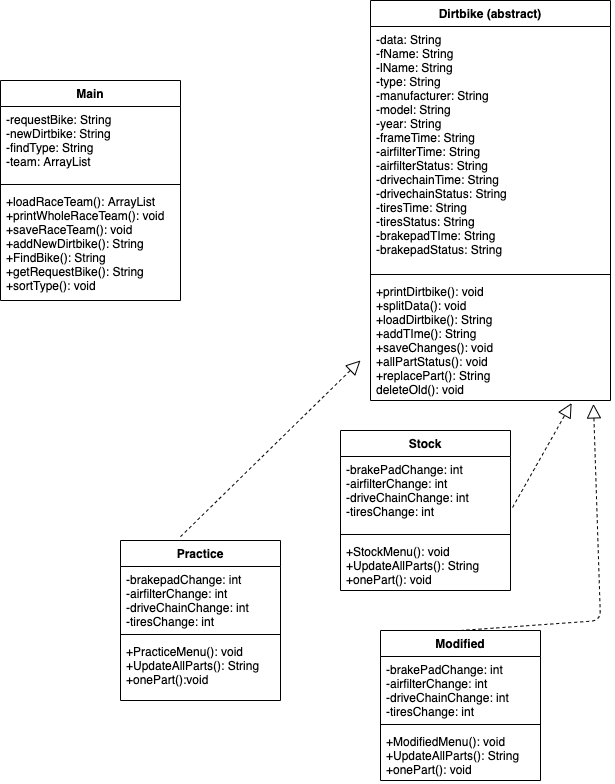






Data Design:

This data is a bunch of integers that represent the time on the motorcycle. The best way to categorize all this data is by using objects to break up all the tasks and maybe use files to keep all the data local. The data will need to be able to change because of users replacing parts.

UML:  


Algorithm:

Final Project Algorithm.

The main goal of this program is to serve as a maintenance checker for a race team of dirt bikes.

Main:

Goal: This will be the main interface that the user gets brought to when using the program. It will cover the basic function of the program.

Input: This class will receive both input from the user and a text file containing the data of all the dirt bikes.

Output: This class will output a file containing the data of the bikes after the user interacts with the data. The class will also output menus and input statements to the user.

Steps:

loadRaceTeam():

Goal: To bring in an arraylist of dirt bikes into the class.

Input: The RaceTeam text file.

Output: An arraylist containing all the data from the text file.

Steps: The method will get a scanner and scan through the text file and use an array list variable and with each pass through the text file, the program will add that line of data into the array list variable.

printWholeRaceTeam():

Goal: To print out the whole arraylist of data to the user to show all the riders and the data associated with each.

Input: The arraylist variable containing the whole team.

Output: The data associated with each dirt bike in the text file.

Steps: Create a loop to loop through for the amount of dirt bikes in the array list. We can then get the specific dirt bike and send it into a string then parse that out to a bunch of variables. Then we can print out a bunch of print code to the user and format it to give a nice presentation to the user based on each dirt bike.

saveRaceTeam():

Goal: To save the new dirtbike being added to the text file.

Input: A string variable holding the new dirtbike to be added to the text file.

Output: A new dirt bike to the text file.

Steps: Use a buffered writer to write a new dirtbike into the text file and close the writer.

addNewBike():

Goal: To allow the user to add a new dirtbike to the race team.

Input: User input

Output: Input questions for the user and a string containing the new dirt bike.

Steps: The method will use print statements to ask the user questions on the new dirt bike that they are going to add to the program. The method will take all this data and add it to a new string that can be added into the text file on a new line to be read again and loaded back into the program.

FindBike():

Goal: To find a specific bike in the arraylist that the user would like to interact with.

Input: The arraylist containing the race team.

Output: A string containing the bike that the user wants.

Steps: The program will ask the user which bike the user wants to interact with and then the program will search through the arraylist and check for that specific bike then when it's found the program will pull it into a variable and pass that variable back to the class.

getRequestBike():

Goal: To get the string containing the requested bike.

Input: none

Output: a string

Steps: return request bike

sortType():

Goal: sort the dirtbike and find out which type of dirt bike the requested dirt bike was.

Input: none.

Output: None

Steps: this program will load up the requested bike and sort through that string until the type of dirt bike is found and then that will be sent back to the main function.

Dirtbike(Abstract):

Goal: To serve as a base for the interactions the user can have with a specific dirt bike.

Input: An arraylist of data.

Output: A menu and a new text file.

Steps:

printDirtbike():

Goal: to print out a dirt bike to the user.

Input: a string containing the dirt bike.

Output: A bunch of print statements to show the user the data associated with the dirt bike.

Steps: The program will use print statements to print out the data to the user.

splitData():

Goal: To split the string of data and pass it back to the variables.

Input: A string of data.

Output: The split up data to variables.

Steps: The program will assign the data in the string to individual variables to be passed back to the program.

loadBike():

Goal: to get the string containing the specific dirt bike.

Input: none.

Output: A string containing the requested dirt bike.

Steps: use the get request bike method to bring it in and assign it to a variable then send it back to the main function.

addTime():

Goal: To give the user the ability to add time to the bike.

Input: user input.

Output: none.

Steps: The method will ask the user for the amount of time they rode and then add it to each variable on the dirt bike then add it back to the main function.

saveChanges():  
Goal: to save the changes to the dirt bike and add it back later.

Input: None

Output: the new saved changes.

Steps: the program will add the new updated dirt bike back to the text file.

allPartStatus():

Goal: to get the status of all parts on the dirtbike.

Input: None

Output: A bunch of print statements.

Steps: The program will use print statements to print out the data to the user.

replacePart():

Goal: to replace a part on the dirtbike.  
Input: The users choice on which part to replace.

Output: none

Steps: the method will ask the user which part they would like to replace and whichever part is chosen, the program will reset the time on the specific part back to zero.

deleteOld():

Goal: to delete the old dirt bike before the manipulation from the text file.

Input: None.

Output: a new updated text file.

Steps: The program will use a text file as a temporary and add the dirtbikes back except the old dirt bike then it will add the new one.

Modified/Stock/Practice(classes):

Goal: these will serve as the classes for the user wanting to interact with a single dirtbike.

Input: A string containing one dirtbike.

Output: None

Steps:

(Modified/Stock/Practice)Menu():

Goal: to serve as a menu for the user to interact with.

Input: user input on menu choice.

Output: a menu to the user.

Steps: The program will print out a menu to the user then ask the user which option the user would like to do.

UpdateAllParts():

Goal: to update the statuses of all the parts on the dirtbike.

Input: None.

Output: The updated statuses of the dirt bike parts.

Steps: The method will use the specific times that the parts that need changed and then the method will compare that with the current times on each part. Depending on the comparison the statuses of the parts will be changed accordingly.

OnePart():

Goal: to show an in depth analysis of a single part on the dirtbike.

Input: a user's choice on the part they would like to see.

Output: the analysis of the one part.  
Steps: the method will ask the user which part they would like to see an analysis of and then the program will grab that part and perform an analysis on the part with its current status, current time, and will show the user how long they have before the part will need to be replaced.