# **Title:**

BingeWolves

# **Purpose:**

I have been playing world of warcraft for the past 7 years and have been through all the changes and new functions that were implemented that either helped or hurt the game, but it is still my favorite game of all time. Seeing all these new programs and functions influenced me to make an application for the World of Warcraft community as well as myself that would benefit us. So, I researched the community forums for problems or functions that users wanted and found out that looking up mount and pet locations was difficult and tedious. So, the purpose of this project is to make an GUI application that will eventually connect to a web service program I will make in the future. I am making this for the World of Warcraft community which will allow the user to search for their character and view that characters stats. It will also allow the user to see their characters collection like the number of mounts they collected, the number of pets they collected, and how many achievement points that character has earned. It would also allow the user to search for a specific mount and it will give them an image of the mount, the description, the location, and the enemy or quest that drops it. Lastly it would allow the user to search for a specific pet and it will give them an image of the pet, a description, the location, the species, and the enemy or quest that drops it.

# **Platforms:**

* Windows

Will be coded for Windows and turned into an executable file with the JRE bundled in it but will in the future try and make the application cross platform for Mac and Linux.

# **Intended Users:**

This application is intended for the World of Warcraft community so that they will be able to search for their character stats, collections, and be able to search for information on a specific mount or pet.

# **Idea Source:**

My idea for this kind of application came from a website that Blizzard released called the WoW Armory which is a site that you can look up your characters stats, achievements, collections, etc.

**Link to the World of Warcraft Armory:**

<https://worldofwarcraft.com/en-us/character/stormrage/Flol%C3%A1ny>

# **Development Environment/Tools:**

* Eclipse
* Maven
* Launch 4J

# **Development Languages:**

* Java

# **Limitations/Risks:**

The main issue that will cause limitations and risks with the program is the Blizzard API.

## Limitations:

* The information from the Blizzard API won’t be exactly up to date meaning that the information retrieved using the API will be couple hours behind the current game information. So, an application that will show you your current characters stats at that current moment as they are in game will not be possible unless blizzard allows for more recent data to be available when calling the API.
* Another limitation is that this application will require internet access to be able to access the Blizzard API and get the required data.

## Risks:

* This program could become useless if Blizzard shuts down or significantly alters their API.

# **Schedule:**

|  |  |
| --- | --- |
| Days | Description |
| January 12 – 13 | Research and plan out my project, Browse the Blizzard API, and Browse the World of Warcraft forums. |
| January 14 – 18 | Work on completing a Proposal/Requirements paper, User Stories, and Use cases. |
| January 19 – 25 | Work on completing a Test Matrix, Finalize Proposal/Requirements paper. |
| January 26 – 31 | Research and plan out how to program my application. |
| February 1 – 8 | Incorporate feedback from Proposal/Requirements paper and complete the Preliminary Design paper and Design End Product Image. |
| February 9 – 22 | Build upon the Preliminary Design and complete the Detailed Design paper and Plan out what classes and Methods I will use to build this application. |
| February 23 – 24 | Continue to plan out my program and design, and finish planning out classes and methods to use |
| February 25 – March 1 | Work on completing a Unit Test Suite, a Skeleton program, and work on my Source code |
| March 2 – 31 | Work on completing my Source code, Test code, and writing a user manual if needed. |
| April 1 – 8 | Work on completing a PowerPoint presentation and practice demonstrating my application. |

# **Deliverables:**

|  |  |
| --- | --- |
| Assignment | Due Date |
| Proposal/Requirements | January 25, 2019 |
| Test Matrix | January 25, 2019 |
| Preliminary Design | February 8, 2019 |
| Detailed Design | February 22, 2019 |
| Unit Test Suite (and Skeleton Code) | March 1, 2019 |
| Implementation | April 1, 2019 |
| Presentation | April 8, 2019 |

# **Estimated length of Program:**

I estimate that my program will be at least 1,000 lines of code.

# **Documentation:**

* User Manual
* Installation guide

# **User Training:**

Users should be able to use this application without any training. Users will only be required to input the name of his/her character or a specific mount or pet and from there they will be able to use the GUI functions to navigate the application.

# **Delivery/Installation Plan:**

After completing this application, I plan to turn my GUI application into an executable file with the JRE bundled with it so that users on Windows can run the application without worrying about setting up Java. For installation I plan to direct the users to my GitHub repository which will have the executable file uploaded. The user can then download it onto their computer and run it. I also plan to advertise it on the Blizzard API and the World of Warcraft Community forums. So that other developers and users can test and use my program, while also giving me feedback on what can be improved.

# **Functional Requirements:**

## System Features:

## Search for a Character

### Description:

This feature allows for a user to enter a World of Warcraft character name from either their list of characters or a random character’s name that they see in the game allowing them to see that character’s image, their stats, their collection of mounts, their pets, and their achievement points. A user can also go back to the search screen and reuse the search function to search for another character.

**Character.Search: Searching for a Character**

.CharacterName: The application will prompt the user to enter the name of their World of Warcraft character.

.Realm: The application will prompt the user to enter the realm of where their character resides.

.Region: The application will prompt the user to choose what region their character resides in for example NA, EU, etc.

.Search: The application will confirm that the name of the character, the realm, and the region exist and pull the data for it.

## Search for a Specific Mount

### Description:

This feature allows for a user to search for a specific mount in the game. The user will be prompted to type in the name of a mount, and it will then give them a picture of the mount, a short description of it, the location of it, and the enemy or quest that drops it. The user can then reuse the search function to search for a new mount.

**Mount.Search: Searching for a Mount**

.MountName: The application will prompt the user to enter in the name of the mount that they want to search for.

.Search: The application will check if the name entered matches and then will pull the data for that mount or return a message saying that the specified mount does not exist.

## Search for a Specific Pet

### Description:

This feature allows for a user to search for a specific pet in the game. The user will be prompted to type in the name of a pet. It will then give them a picture of the pet, a short description of it, the location of it, the species, and the enemy or quest that drops it. The user can then reuse the search function to search for a new pet.

**Pet.Search: Searching for a Pet**

.PetName: The application will prompt the user to enter in the name of the pet that they want to search for.

.Search: The application will check if the name entered matches and then will pull the data for that pet or return a message saying that the specified pet does not exist.

## Data Flow Diagram:

## 

## Software Interface:

This application will eventually interact with the WoW API Key system which will be a web service that I will be program in the future because of it making the project scope too large. This web service will contain part of the GUI application code that has the Blizzard API access token so that it will be protected from users being able to access it. But until then this application does not have any designated software it relies on, so it does not have any direct software interfaces.

## Hardware Interface:

Since this application does not have any designated hardware it does not have any direct hardware interfaces.

## Communication Interface:

* This application will be communicating with the Blizzard API and the application will ensure that the data being inputted is checked for errors.
* This application will use a TCP protocol when connecting to the Blizzard API and collecting the data required.

# **Nonfunctional Requirements:**

## Availability:

* The user must have internet access to be able to connect to the Blizzard API servers and gain access to the data.

## Performance:

* The time it takes from when the user clicks the search character button for the character stats and collection data being obtained and shown shall take no more than 2 second.
* The time it takes from when the user clicks the search mount button for the mount data to be shown shall take no more than 2 seconds.
* The time it takes from when the user clicks the search pet button for the pet data to be shown shall take no more than 2 seconds.
* The user shall only be submitting one character, mount, and pet search request at a single time.
* When the user is switching from the character data frame to either the character collections frame, the mount search frame, or the pet search frame there shall not be a delay longer than 1 second.
* Data requested from the Blizzard API shall fully download in an average of 2 seconds or less over a 100 megabits/seconds internet speed.

## Reliability:

* The user should be able to perform character searches, mount searches, and pet searches without any system errors happening.

## Robustness:

* If the Blizzard API should be down the application will show the user a message stating that the API is down and to try again later.
* If the user searches for a character, mount, or pet and that character, mount, or pet does not exist, the application will prompt the user that it was not found and to enter another name.

## Security:

* Users shall not be able to view the access tokens that the Blizzard API provides for the application(won’t be able to implement this until the WoW API Key system is developed).

## Usability:

* The application will allow a user to reuse the search function to search for a new character with a single interaction.
* The application will allow the user to reuse the search function to search for a new mount with a single interaction.
* The application will allow the user to reuse the search function to search for a new pet with a single interaction.
* The applications interface should be user friendly and easy to navigate.
* 95 percent of new users who have never used this application before shall be able to search for a character, a mount, and a pet on their first try after only reading the user manual for no more than 5 minutes.

## Disk Space/Memory:

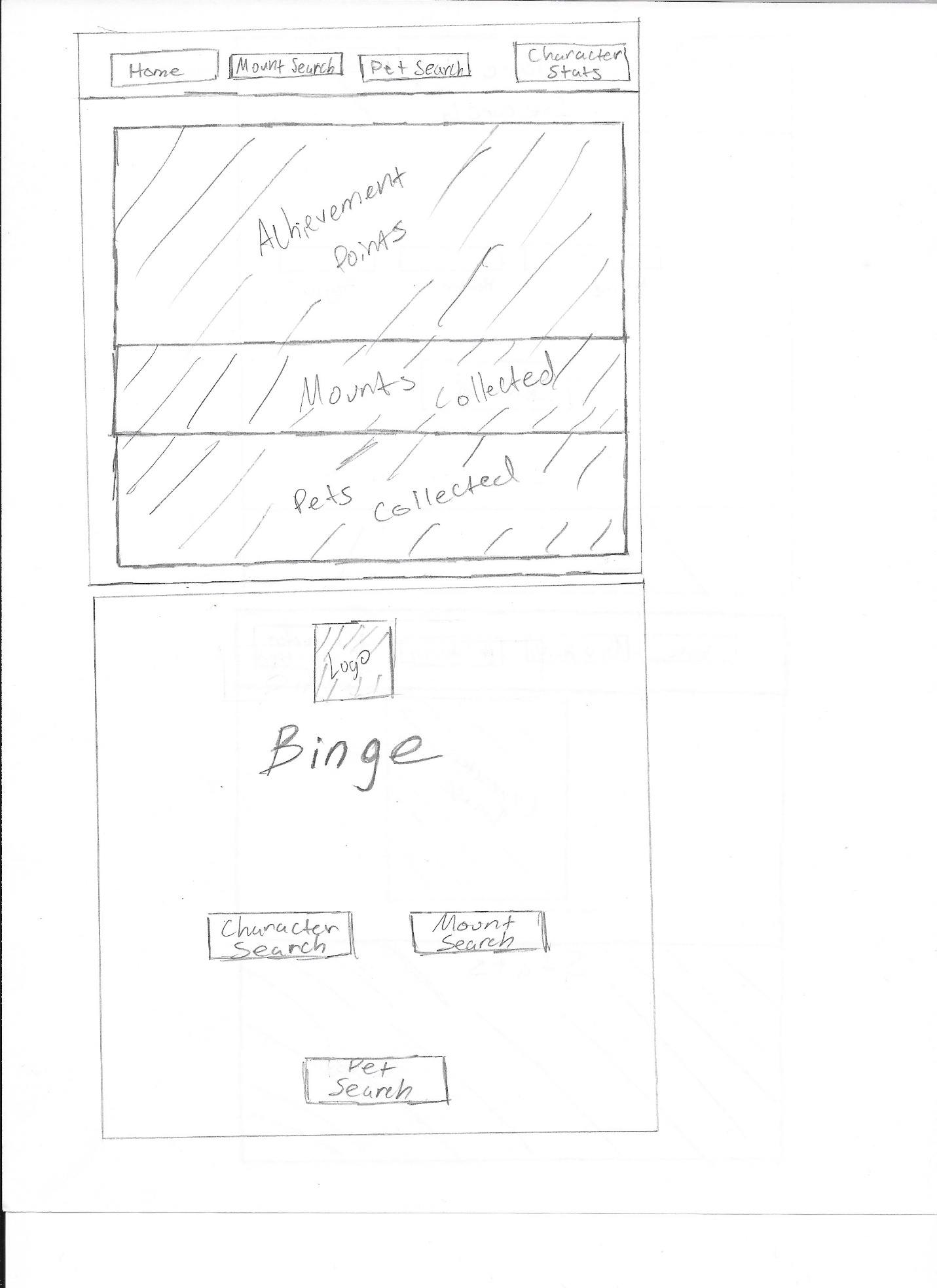
* This application will require users to have at least 500 megabytes of memory free and at least 500 megabytes of free disk space as well on their computer.

## Maintainability:

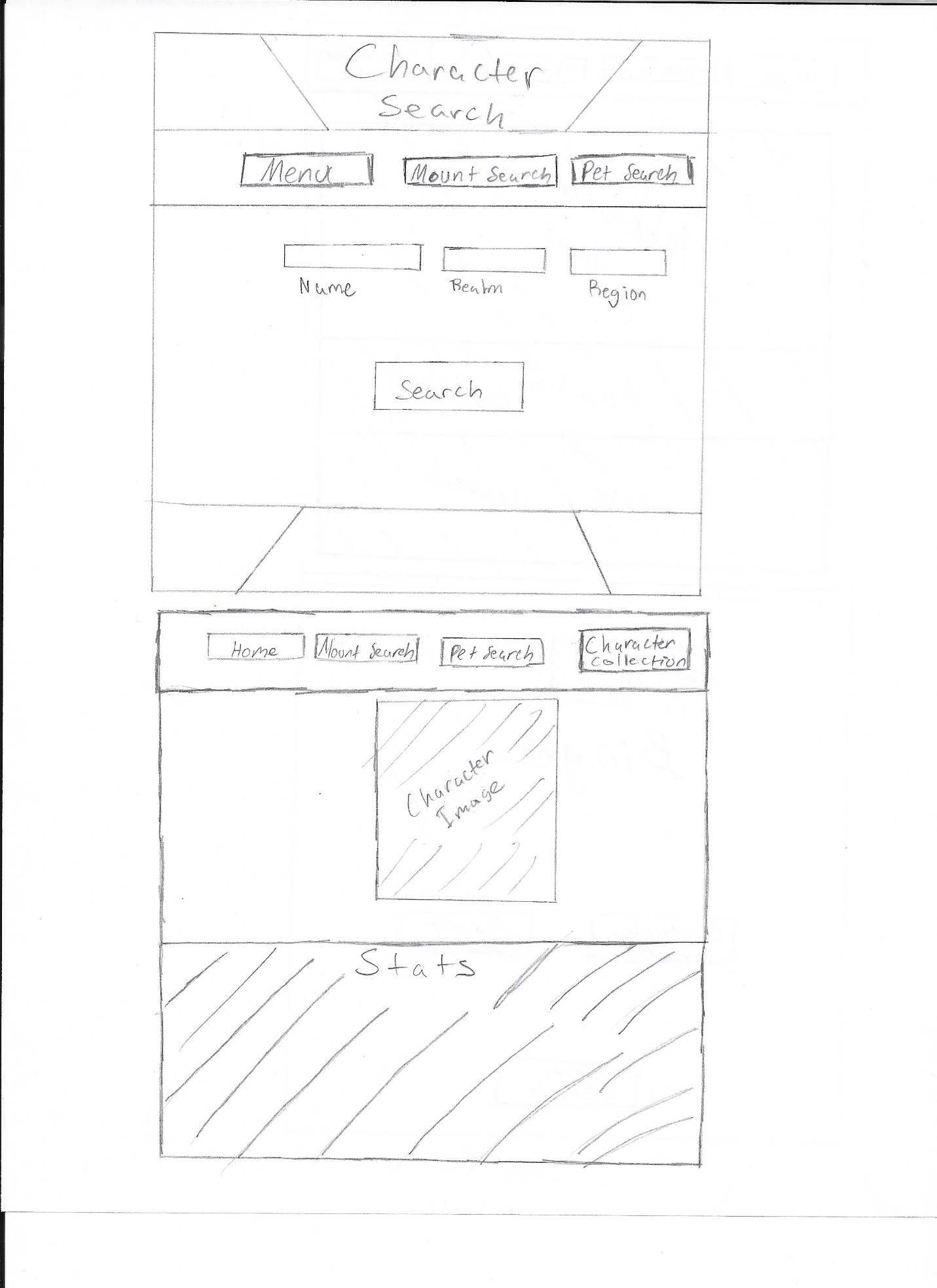
This application once released will not be able to be maintained remotely because of it not being connected to a server. The code should be written in a way so that it will favor the implementation of new functions. This application will not be very adaptable with other software that wants to change its interface, and neither will it very adaptable to changes in specific functionality. Even though it won’t be very adaptable it will be able to adapt to some changes in the operating environment.

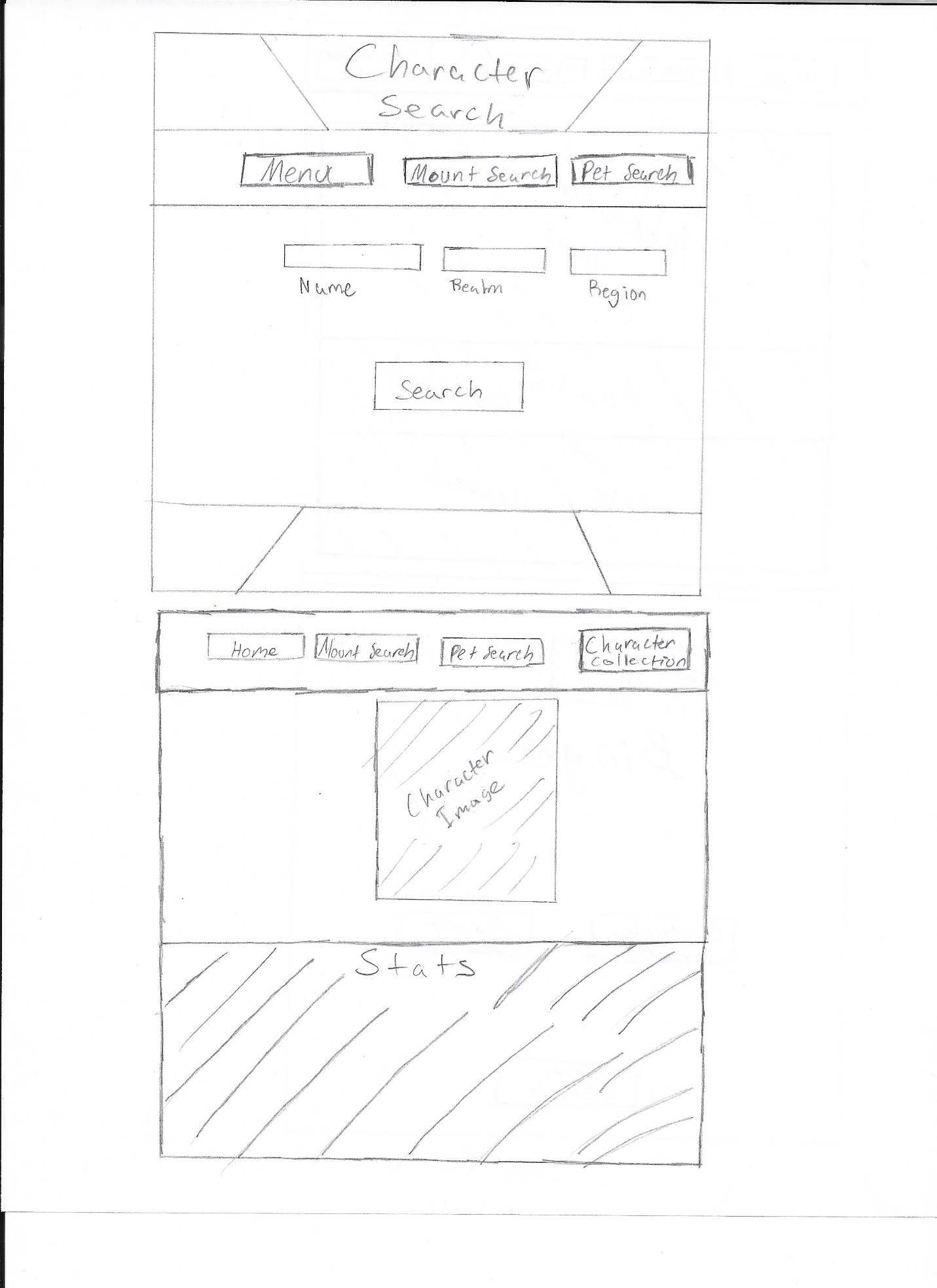
# **System Design:**

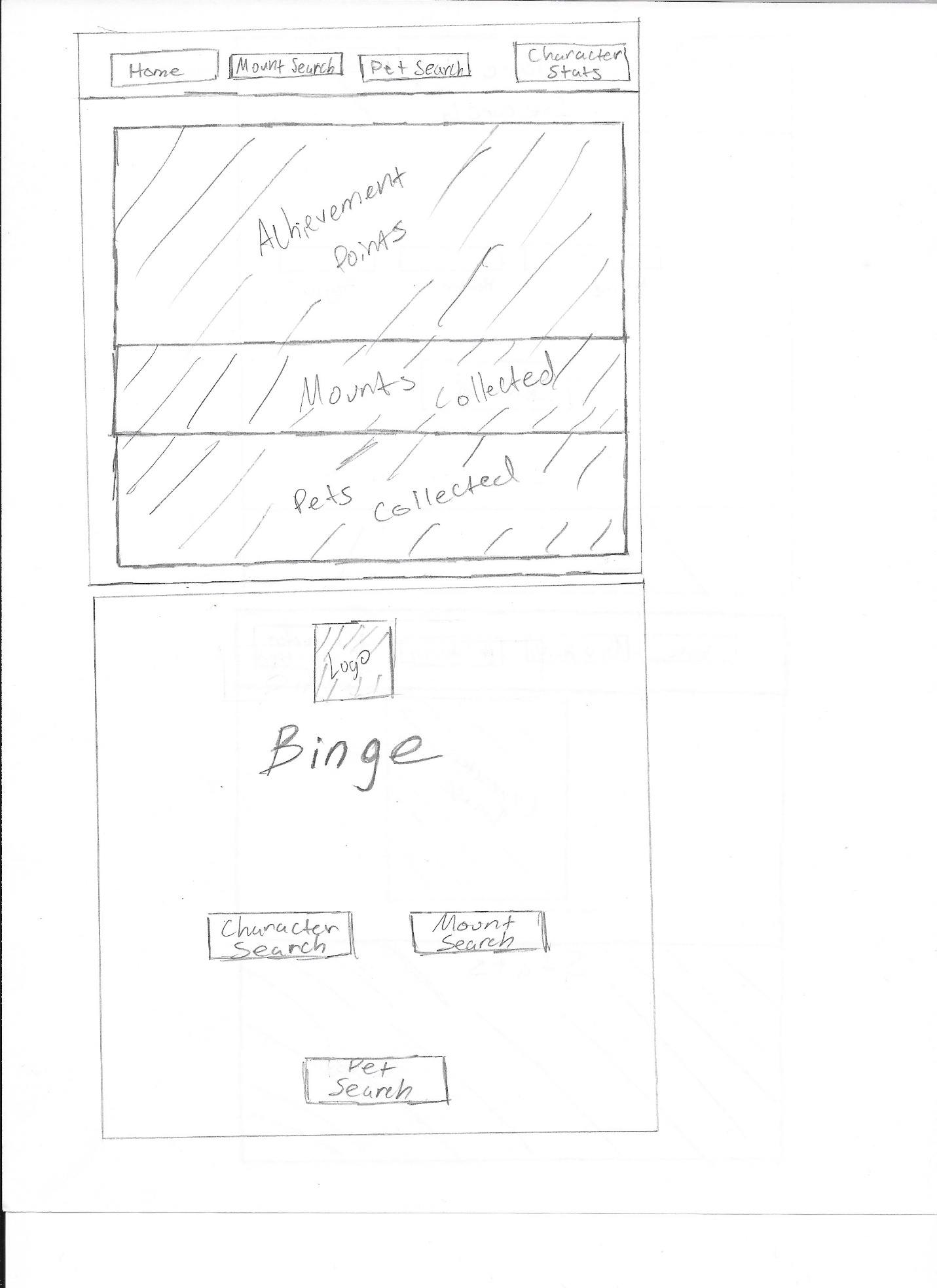
## Interface Sketches:



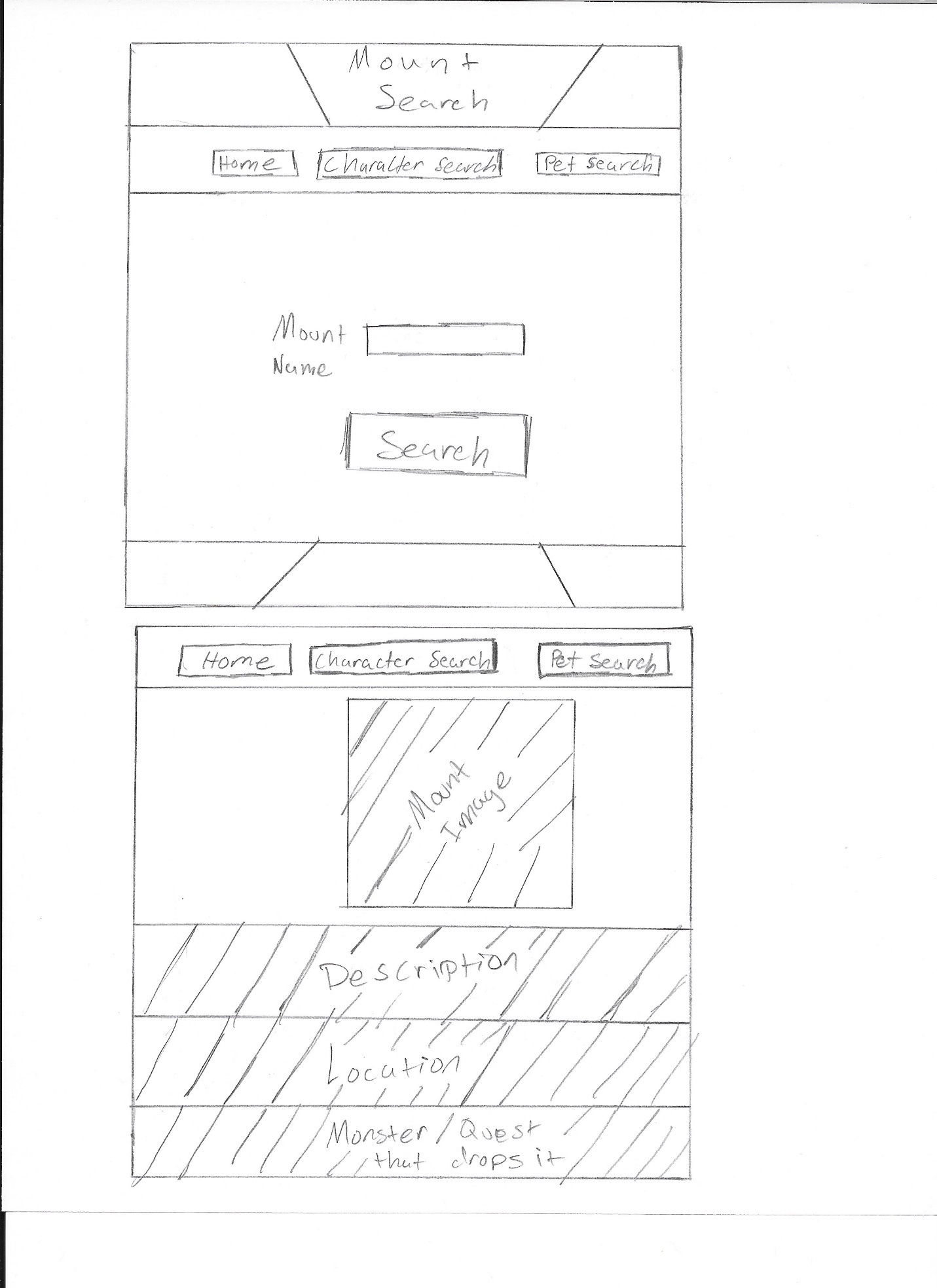
The image above is what the main screen of the Binge Application will look like and is what the users will use to navigate to the applications features.

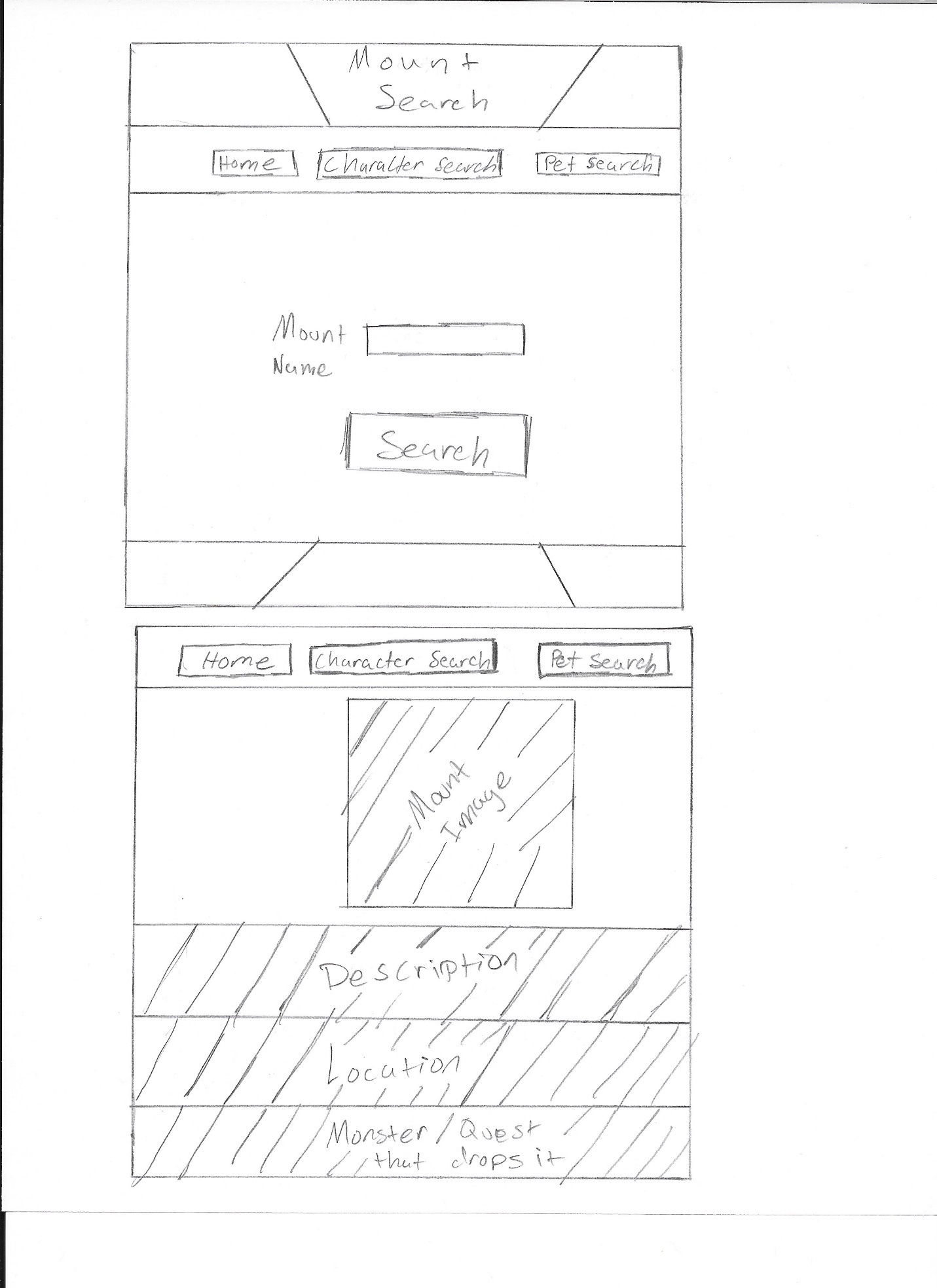




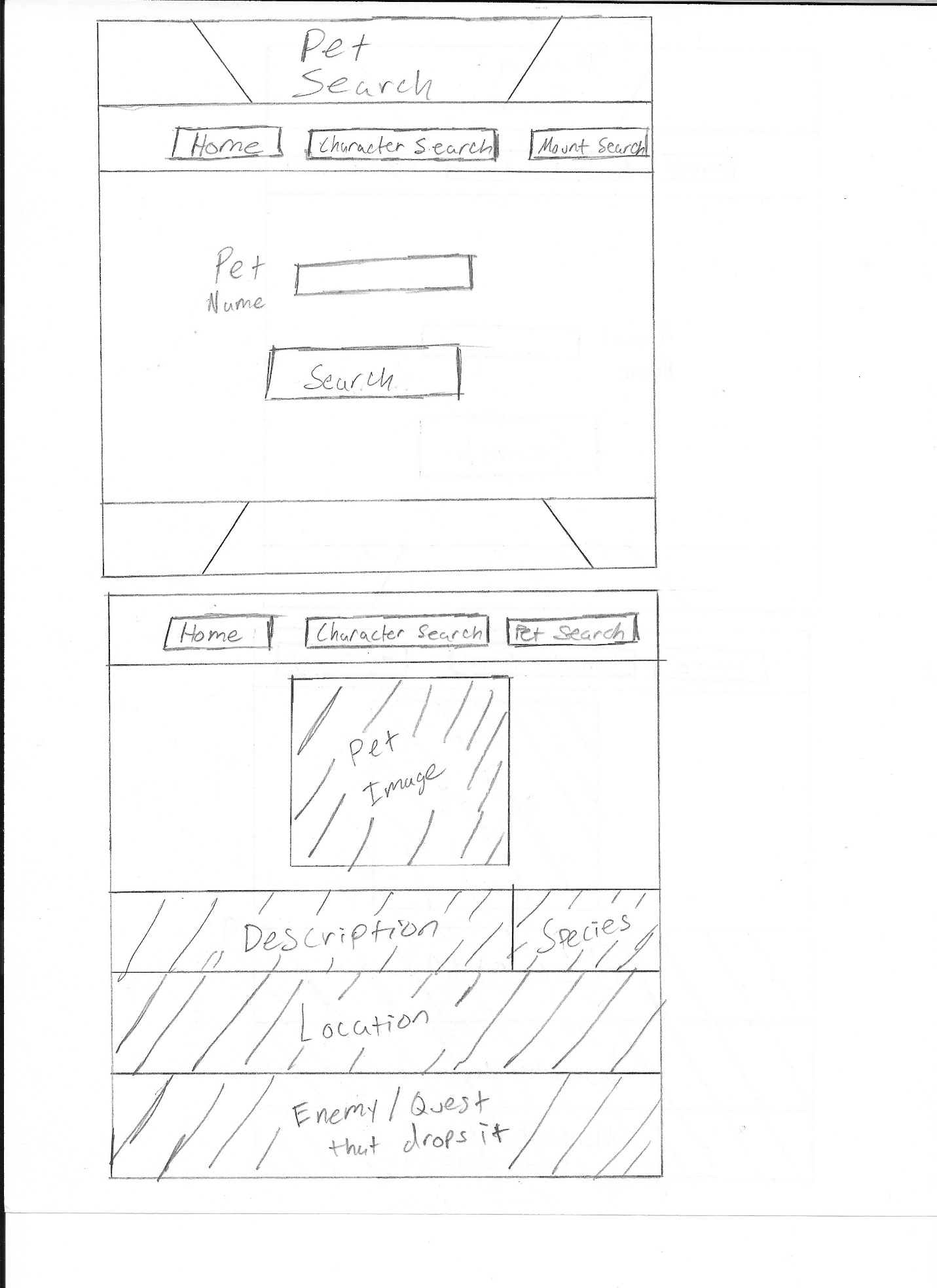


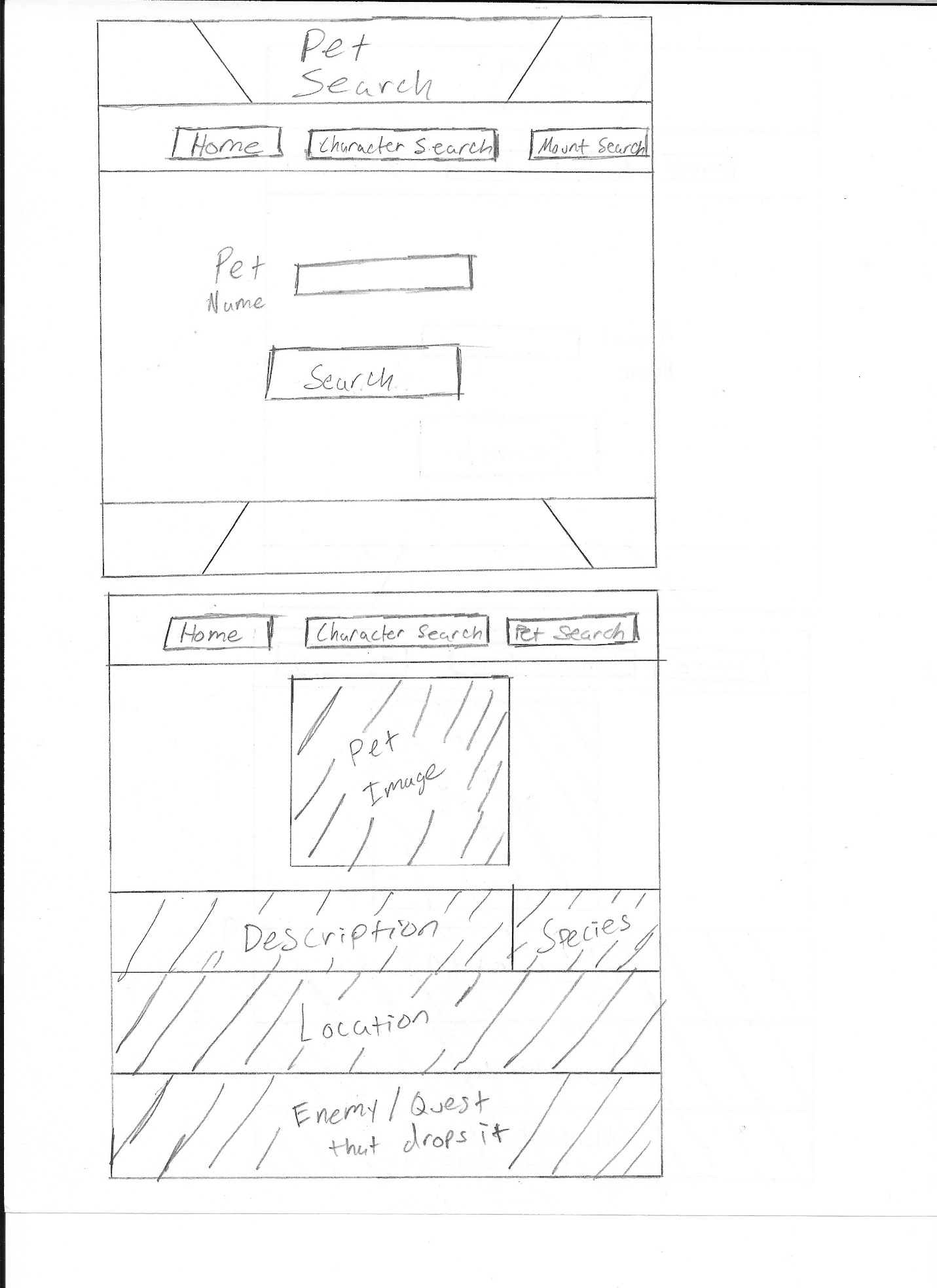
The three images above are all the screens for the Character Search feature the user will first be shown a screen where they will be asked to input the character they wish to search for. Then once they click the search button, they will be shown the next screen with the character image and stats. If the user wishes to view that character’s collection, they can click the character collection button which will then take the user to the third screen with that character’s collection. The user can then use the buttons to navigate back and forth or to the other features in the application.





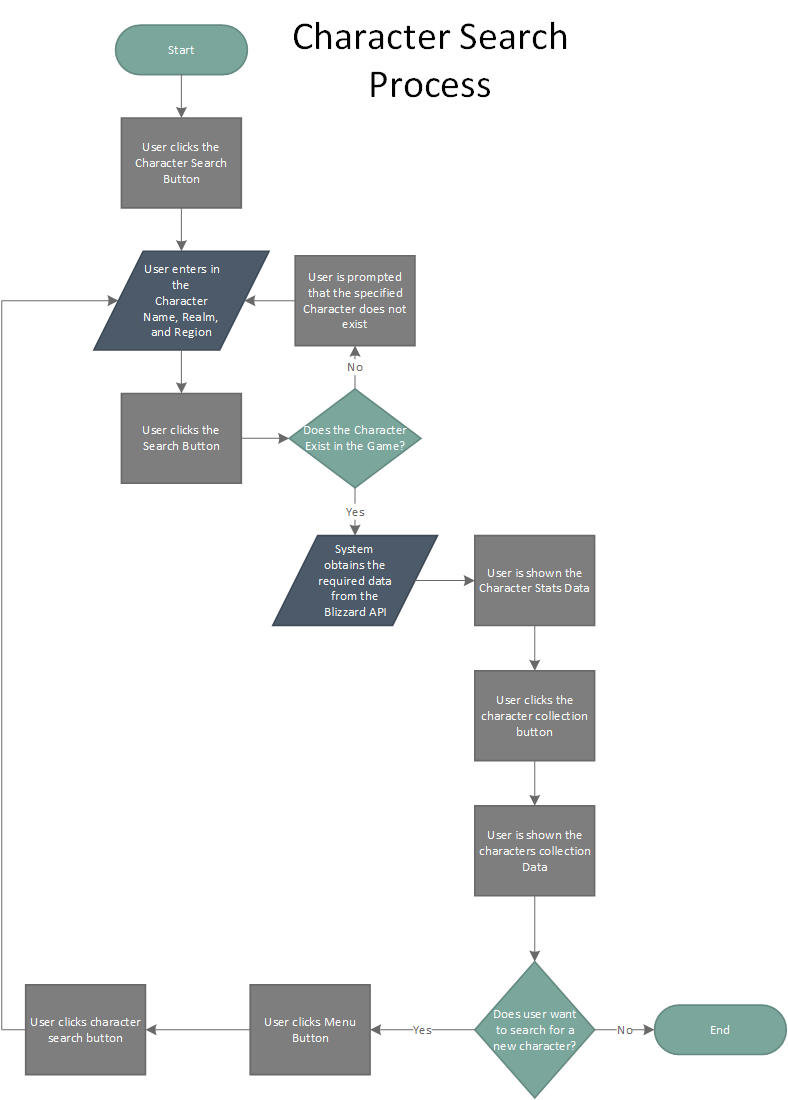
The two images above show the Mount Search feature. The user will first be shown a screen where they will input the Mount name, they wish to view then once they click the search button it will take them to a new screen where they will see the Mount data. The user can then use the buttons on top to navigate to the other features or back to the main screen.

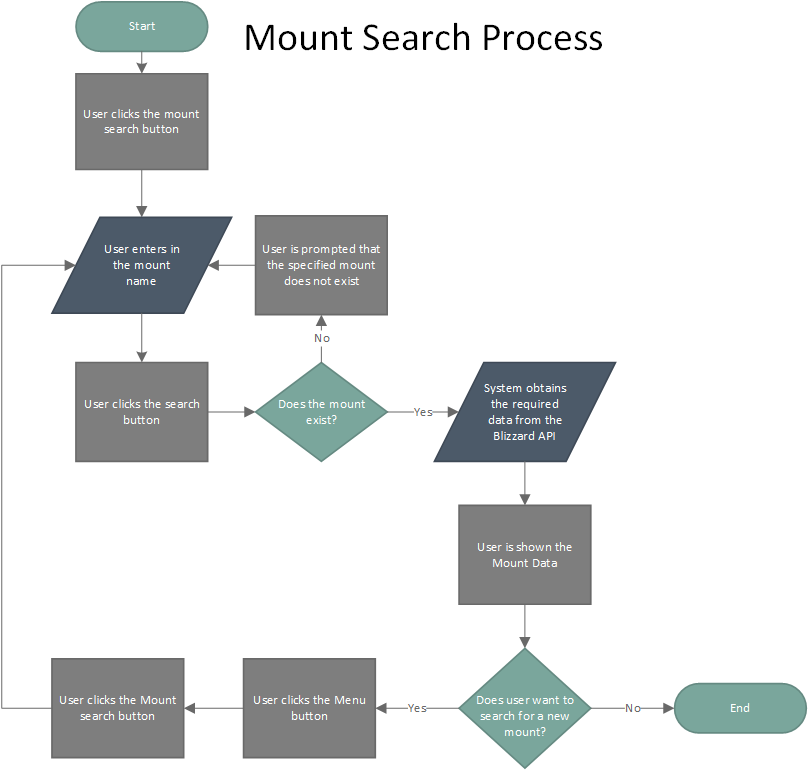


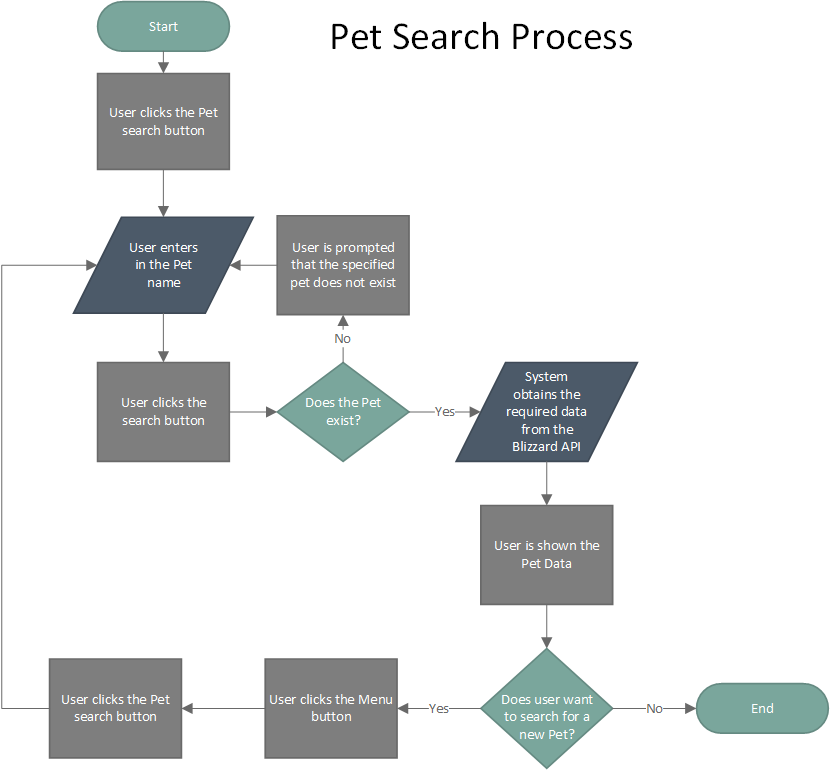


The two images above show the Pet Search feature. The user will first be shown a screen where they will input the Pet name, they wish to view then once they click the search button it will take them to a new screen where they will see the Pet data. The user can then use the buttons on top to navigate to the other features or back to the main screen.

## Flow Charts:



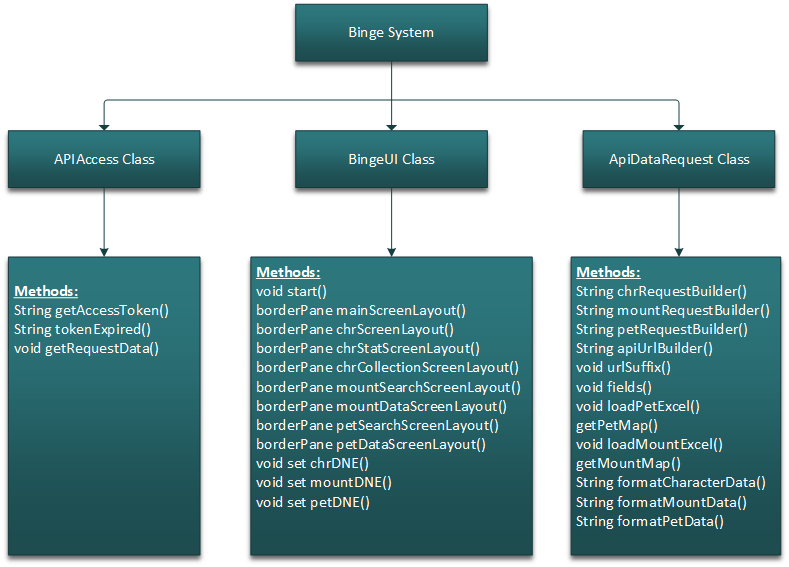




## Hierarchy/Structure Chart:



This chart shows all the UI functions and where that function will lead you when clicked.



The chart above lists all the three major classes that my application will have and all the methods that each class will have.

## Methods:

**APIAccess Class:**

Private String getAccessToken()

This method will obtain the access token required to make a request to the Blizzard APIs

Private String tokenExpired()

This method will check if the token is expired and obtain a new token if true

Public String getRequestedData(String requestUrl)

This method will get the URL request as a parameter and return the called data from the Blizzard APIs

**BingeUI Class:**

Public void start()

Because I am using JavaFx to make my GUI this method is required

Private BoderPane mainScreenLayout()

This method will make the layout of the main screen in the application.

Private BorderPane chrSearchScreenLayout()

This method will make the layout of the character search screen.

Private BorderPane chrStatsScreenLayout()

This method will make the screen that will layout the requested characters stats.

Private BorderPane chrCollectionsScreenLayout()

This method will make the screen that will layout the collections of the requested character.

Private BorderPane mountSearchScreenLayout()

This method will make the layout of the mount search screen.

Private BorderPane mountDataScreenLayout()

This method will make the layout that will list the requested mount data.

Private BorderPane petSearchScreenLayout()

This method will make the layout of the pet search screen.

Private BorderPane petDataScreenLayout()

This method will make the layout that will list the requested pet data.

Public void set chrDNE()

This method will set the label to show the user that the character they inputted does not exist.

Public void set mountDNE()

This method will set the label to show the user that the mount they inputted does not exist.

Public void set petDNE()

This method will set the label to show the user that the pet they inputted does not exist.

**ApiDataRequest Class:**

Public String chrRequestBuilder(String chrName, String realmName, String regionCB)

This will build the character request URL that will be passed to the apiUrlBuilder method

Public String mountRequestBuilder(String mountName)

This will build the mount request URL that will be passed to the apiUrlBuilder method

Public String petRequestBuilder(String petName)

This will build the pet request URL that will be passed to the apiUrlBuilder method

Public apiUrlBuilder()

This method will make the actual URL and pass it to the getRequestedData method

Private void urlSuffix()

This method will have the default start of the request URL

Private void fields()

This method will have a switch with the character fields, the mount fields, and the pet fields.

Private void loadPetExcel()

This method will load the pet excel file that will contain the pet name and the species id

Private Map<String, String> getPetMap()

This method will return a map of the data in the pet excel file

Private void loadMountExcel()

This method will load the mount excel file which will have the name of the mount, the display id, the description, and the location.

Private Map<String, String> getMountMap()

This method will return a map of the of the data in the mount excel file.

**CharacterNotFoundException Class:**

Public CharacterNotFoundException()

This is a custom exception that will be thrown when the specified character is not found

**MountNotFoundException Class:**

Public MountNotFoundException()

This is a custom exception that will be thrown when the specified mount is not found

**PetNotFoundException Class:**

Public PetNotFoundException()

This is a custom exception that will be thrown when the specified pet is not found

## Pseudocode:

Obtaining an Access Token

This routine will obtain an access code from Oauth using my client credentials which is obtained from making a client on the Blizzard developer page. It will use an Http Request and pass the Oauth URL with the parameters my client ID and Client password. This request will obtain a Json with the access key, the access type, and the time expired. Then this routine will parse through the Json for the access token and return it.

Set the client key to the Client ID

Set the client secret to the Client password

Set the access token URL to the Oauth URL

Initialize an empty string called accessToken

Make a new Closeable Http Client

Initialize a new List called params and set it equal to a new Vector

Add the client key to params

Add the Client secret to params

Add the grant type and the client credentials to params

Initialize an Http Post request to the access token URL

Construct the URL with the params

Make a Http Response with the URL

Initialize an empty string token to the Json received

Initialize a Json Element called jElement and set it equal to a new Json Parser to parse token

Initialize a Json object called jToken and set it equal to jElement converted as a Json object

Set accessToken to access token

Return accessToken