The Wanderer: Implementing Markerless Augmented Reality with Object Position Awareness

Mina Samir, Ahmed Hanie, Aly Aboulgheit, Karim Hossam
Supervised by: Dr. Ayman Ezzat and Eng. Noha Elmasry

November 28, 2017

1 Introduction

1.1 Purpose of this document

The purpose of this Software Requirements Specification document is to outline the requirements for a markerless augmented reality game. The Wanderer is aimed to use Object recognition to superimpose the virtual objects on real time objects, and with the use of Geolocation in able to locate and position the virtual object. The Wanderer is a Role Playing Game that aims for the user to go to certain areas indoors and outdoors in order to take quests, trade and get resources to craft items. The project will consist of an android based mobile with a camera and connected to the internet with that the user could be entertained with the game.

1.2 Scope of this document

The Wanderer is aimed to entertain all people that enjoy moving around and having fun. The game is a location-based augmented reality game that everyone should enjoy. The user will feel that they are inside the game, taking adventures and completing tasks that will help them level up and become more adaptive toward the gameplay.

1.3 Overview

Augmented reality games has taken over the world by a surprise starting with Ingress to Pokemon GO. This field makes the user entertained and help developers understand ans use the markerless augmented reality algorithms. The idea of The Wanderer is that the user go to a specified real time place to accomplish a certain quest, this quest could be fetching something from a place and returning it, delivering a certain item from a place to another, or combating some enemies. The quest will be located using the smartphone's sensors fused together. GPS to locate the quest in the map, using gyroscope to locate the orientation on the quest, and the compass to locate the quest's direction relative to the geographic cardinal directions , once the user enters the field, the sensor fusion will activate to locate the quest. The quest will be superimposed by using the markerless augmented reality's algorithm object recognition and with the help of the smartphone's sensor fusion, the item will be augmented on real time objects. Same strategy will be allied on the enemy, or items.

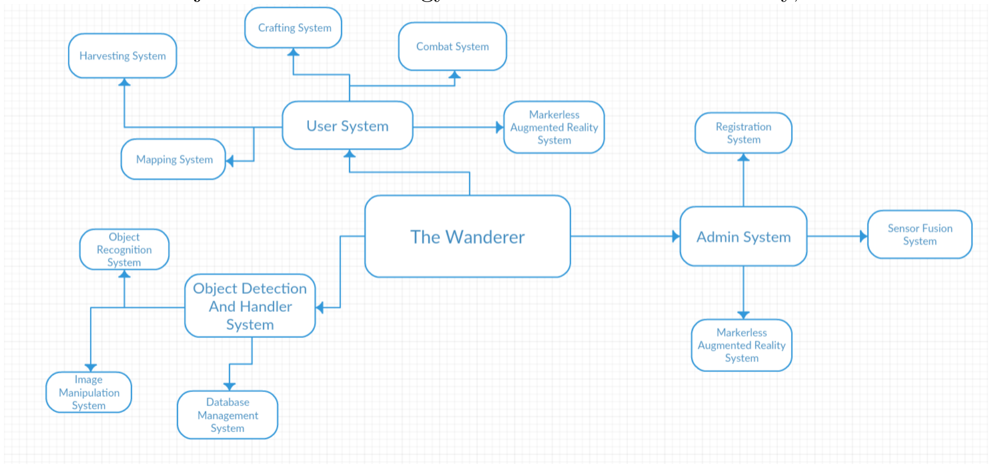

Crafting System
Combat System
Harvesting System
User System
Markerless Augmented Reality System
Registration System
Mapping System
The Wanderer
Admin System
Sensor Fusion System
Object Recognition System
Object Detection And Handler System
Markerless Augmented Reality System
Image Manipulation System
Database Management System

Figure 1: Block Diagram

1. User System:

   (a) Markerless Augmented Reality System: This system manages the superimposed objects in the game. Allows the user to interact with them

   (b) Combat System: This system aims to attack an NPC, defend foreign attacks, equip items and unequip items during combat.

   (c) Crafting System: This system allows the user to craft items in game like weapons, armors and consumables using harvested resources.

   (d) Harvesting System: This system allows the user to harvest resources from a predefined places to use them for crafting or trading.

   (e) Mapping System: This system is a map that has all the predefined places for the user to interact with them like shops, quests and resources.

2. Admin System:

   (a) Registration System: This system manages the information of the users and admins like the log in or sign up.

   (b) Sensor Fusion System: This system takes the sensor readings from the mobile device (Gyroscope, Magnetometer, GPS and Accelerometer) sends them to the database for storage, and helps in adding or locating superimposed objects.
   (c) Markerless Augmented Reality System: This system helps the Admin to add augmented objects in the predefined places.

3. Object Detection And Handler System:
   (a) Object Recognition System: This system recognize real time objects in order to add the virtual object on them.
   (b) Image Manipulation System: This system is aimed to manipulate the taken image from the background of the virtual object, like cropping or resizing.
   (c) Database Management System: This system is aimed to send the results to the database for storage, also it the database gets a new frame, this system sends it to the server to be processed.

1.4 Business Context

Nowadays, Augmented reality has been an uprising topic that makes people move from one place to another in order to achieve a certain mission. It helps the user to grasped the link between reality and virtual reality. An augmened reality game is a way that people enjoys virual objects imposed on real life, especially after pokemon GO has made over 100 million downloads, the call for an augmented reality game has been called upon. The Wanderer will contain quests that the user will complete, these quests will have certain missions that the user will travel to the place mentioned to complete it. Crafting will be available and will allow the user to create items like swords, shields or amulets that will help the user to fight his enemies. To craft the user will go to a place that has resources, after collecting it the user will open the crafting menu, if the user has the necessary resources, then the user may craft it. The combat mechanize will be turn-based where the user will be given choices on how to inflect damage on the enemy , and the user may use the items which has been crafted to fight it. The enemy will then attack the user, which ever reaches the health to zero will win. The user will level up whichever the quest is completed, craft an item, defeated an enemy or collect a resource. Leveling up would help the user learn some skills which will help in combat or crafting or purchasing from a shop. The shops will be in a defined place where the user will purchase items or weapons to ease combat or collecting resources. Collecting resources, where there will be defined places that the user will go to to collect them, the user will go and click on the mine or others to collect. The transactions will be made by a virtual currency (Coins) that the user gains upon completing a quest or defeating an enemy.

2 General Description

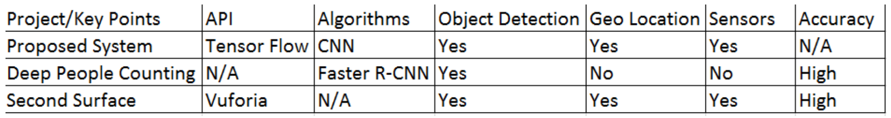
2.1 Product Functions

1. Object detection accuracy
2. Object detection speed
3. Differentiating between multiple instances of the same object
4. Markerless augmented reality
5. Real-time computation of object detection on mobile's camera
6. Real-time update of augmented information as multiple users use the system at the same time
7. Performing object detection on a server
8. Solving Interior Location tracking
9. Using Sensors to help detecting real time objects, and superimposing the virtual objects
10. The user may trade with NPC.
11. The user can battle an NPC.
12. The user can gather resources from real time places.
13. The user will use these resources to craft items, to aid them in battles.
14. The user will takes quests from NPC, certin mission on which the user must complete.

2.2 Similar System Information

In the last year, Pokemon GO [1][2] has been a huge influence on a lot of people, it made people go and try to catch them all, and re-live every person's childhood. Pokemon GO is a free-to-play, location-based augmented reality game developed by Niantic for iOS and Android devices. Players create and customize their own avatars[3]. Once created, an avatar is displayed on a map based on the player's geographical location. Features on the map include 'PokStops' and 'Pokmon gyms'. PokStops provide players with items, such as eggs, Pok Balls, berries, and potions. These PokStops can be equipped with items called 'lure modules', which attract additional wild, and occasionally rare, Pokmon. Gyms serve as battle locations for team-based king of the hill matches. PokStops and gyms are typically located at places of interest [4]. As players move within their real world surroundings, their avatars move within the game's map. Different Pokmon species reside in different areas of the world; for example, water-type Pokmon are generally found near water[5].

When a player encounters a Pokmon, it may be viewed either in augmented reality (AR) mode or with a live rendered, generic background. AR mode uses the camera and gyroscope on the player's mobile device to display an image of a Pokmon as though it were in the real world. The Wanderer and Pokemon GO have some common grounds. The geolocation system is implemented on both applications to help identify the location of the augmented object. Both of the applications are markerless augmented reality game. The difference is that The Wanderer uses object detection to superimpose the augmented object on a real time object, where Pokemon GO uses only geolocation to superimpose the augmented object. In the system Second Surface [6], the system was able to add an augmented object superimposed on a real time objects markerlessly. The researches used Vuforia [6] and dictionaries to store classifiers to be able to pin the posts without markers. Our project will use the markerless part in this paper which will help us to place an object without marker.



Project/Key Points	API	Algorithms	Object Detection	Geo Location	Sensors	Accuracy
Proposed System	Tensor Flow	CNN	Yes	Yes	Yes	N/A
Deep People Counting	N/A	Faster R-CNN	Yes	No	No	High
Second Surface	Vuforia	N/A	Yes	Yes	Yes	High

2.3 User Characteristics

1. User must have basic android knowledge.
2. User is comfortable in going outside and play the game.
3. The user must know how to use the phone camera.

2.4 User Problem Statement

Our project mainly focus on creating A Markerless Real-time Augmented Reality Application that enhance awareness of Object positioning indoor/outdoor.

2.5 User Objectives

Game's items will be superimposed by using the object recognition algorithm and with the help of the smartphone's sensor fusion, the item will be augmented on real time objects. Object detection algorithm speed will have to increase in order to superimpose the augmented items quickly so the user will interact with it.

2.6 General Constraints

1. User must have an android device with a camera at least 1280x960 Capture Resolution.
2. User must be connected to the internet.

3. Android device must have atleast GPS, Gyroscope, Magnetometer and accelerometer sensors.
4. User must enable the sensors such as Global Positioning System or/and Gyroscope.
5. The android software must be at least 4.1 version (Jelly beans).

3 Functional Requirements

3.1 Server

1. ID: SR1
   - Title: ReadImage
   - DESC: The Server would read an image from the local file system through its path.
   - Input: file path string.
   - Action: Reading an image from the local file system.
   - Output: Image.
   - Pre-condition: The image must exist on the file system.
   - Post-condition: None.
   - DEP: SR7, SR10, SR9, MB3, MB6, MB7, MB1.
2. ID: SR2
   - Title: WriteImage
   - DESC: The Server would write an image to the local file system from a given path.
   - Input: file path string.
   - Action: Writing an image to the local file system.
   - Output: Image.
   - Pre-condition: Image variable in memory must not be empty.
   - Post-condition: None.
   - DEP: SR7, SR1, SR10, SR9, MB3, MB6, MB7, MB1.
3. ID: SR3
   - Title: CropImage
   - DESC: The Server would crop the image returning the area between the top left pixel and bottom right pixel.
   - Input: Image, top left pixel and bottom right pixel.

   - Action: Cropping an image into a specific size.
   - Output: Cropped Image.
   - Pre-condition: Image variable in memory must not be empty.
   - Post-condition: None.
   - DEP: SR1, SR7, SR10, SR9, MB3, MB6, MB7.

4. ID: SR4
   - Title: GetObjectLocationInImage
   - DESC: The server would find the object location inside an image using feature matching.
   - Input: Image Features
   - Action: Detecting the location of objects in this image.
   - Output: Locations of objects as pixels (x,y)
   - Pre-condition: Image Sent From the database.
   - Post-condition: Storing the results.
   - DEP: SR9, SR10, SR11, SR12.

5. ID: SR5
   - Title: SendReplyError
   - DESC: The server will send a reply with an error code to firebase database.
   - Input: None.
   - Action: Sending a reply to firebase database.
   - Output: Sent Conformation
   - Pre-condition: Exception fired or error caught.
   - Post-condition: None.
   - DEP: SR9, SR10, SR12.

6. ID: SR6
   - Title: StoreImageInformationInDB
   - DESC: The Server would send the image information to firebase database.
   - Input: Image features, keypoints, geohash and path in firebase storage.
   - Action: Storing image data in firebase database.
   - Output: Image unique key in database.
   - Pre-condition: All information must be valid and complete.

- Post-condition: None.
- DEP: SR1, MB1, SR9, SR10, MB3, MB6, MB7.

7. ID: SR7

    - Title: DownloadImageFromDB
    - DESC: The Server would download an image from firebase storage to the local file system.
    - Input: Image path in firebase storage as string.
    - Action: Downloading image to local file system.
    - Output: Image successfully downloaded and a path to the image in the local file system.
    - Pre-condition: The path to image must be correct in firebase.
    - Post-condition: None.
    - DEP: SR9, SR10.

8. ID: SR8

    - Title: ExtractImageFeaturesAndKeypoints
    - DESC: The Server would extract features from an image using ORB algorithm.
    - Input: Image.
    - Action: Extracting image features.
    - Output: Features array and keypoints array.
    - Pre-condition: Image variable in memory must not be empty and image must have sufficient distinct features for extraction.
    - Post-condition: None.
    - DEP: SR1, SR7.

9. ID: SR9

    - Title: InitializePyrebase
    - DESC: The server will initialize pyrebase which is a python wrapper for firebase connection class.
    - Input: key-value pair firebase configuration information.
    - Action: Initializing database connection.
    - Output: None.
    - Pre-condition: Configuration information must be correct.
    - Post-condition: None.
    - DEP: None.

10. ID: SR10
    - Title: AuthenticateUser
    - DESC: The server will sign in to firebase with a specified username and password to receive authority to manipulate database and storage.
    - Input: username and password.
    - Action: Authenticating firebase user.
    - Output: Authentication Token.
    - Pre-condition: A super user account must be created in the firebase web application.
    - Post-condition: None.
    - DEP: SR9.
11. ID: SR11
    - Title: QueryForImagesInGeohash
    - DESC: The server will query for all images in a specific range using geohash. Geohash is used as an external library.
    - Input: Geohash.
    - Action: Querying Images.
    - Output: Array of images information.
    - Pre-condition: Geohash must be valid.
    - Post-condition: None.
    - DEP: SR9, SR10.
12. ID: SR12
    - Title: ListenOnDatabaseForRequests
    - DESC: The server will listen on firebase database to be notified of new requests.
    - Input: None.
    - Action: Listening for requests.
    - Output: Periodically receive a request message containing image, request code and maybe a new object.
    - Pre-condition: Application must be connected to firebase.
    - Post-condition: None.
    - DEP: SR9, SR10.
13. ID: SR13

- Title: MatchFeatures
- DESC: The server will determine if a new image already exists in the database by matching its features with other images' features that already exist in the same geohash of the new image using Brute Force matcher.
- Input: New image features, Array of images' features in the same geohash.
- Action: Feature Matching.
- Output: The index of the image in the array if it was found or -1.
- Pre-condition: image features must be valid.
- Post-condition: None.
- DEP: SR9, SR10, SR12, SR11.

14. ID: SR14

    - Title: QueryForObjectsInImage
    - DESC: The server will query for objects in a specific image from firebase database.
    - Input: Image unique key in firebase database.
    - Action: Querying objects.
    - Output: Array of objects information.
    - Pre-condition: Image key must be correct.
    - Post-condition: None.
    - DEP: SR9, SR10, SR12, SR11.

15. ID: SR15

    - Title: SendReplyObjectsFound
    - DESC: The server will send a reply with objects found success code to firebase database.
    - Input: Objects locations and additional information array.
    - Action: Sending a reply to firebase database.
    - Output: Sent Conformation.
    - Pre-condition: None.
    - Post-condition: None.
    - DEP: SR9, SR10, SR12, SR11.

16. ID: SR16

    - Title: StoreObjectInformationInDatabase

- DESC: The server would send the object information to firebase database.
- Input: Object's features, image unique id.
- Action: Storing an object in firebase database.
- Output: Sent Conformation.
- Pre-condition: Object's information must be correct.
- Post-condition: None.
- DEP: SR9, SR10, SR12, SR11.

17. ID: SR17

    - Title: RefreshAuthenticationToken
    - DESC: The server would continuously refresh the authentication token each 45 minutes as it expires each hour using threading.
    - Input: Account's refresh token.
    - Action: Refreshing authentication token.
    - Output: Authentication Token.
    - Pre-condition: User must be already authenticated and refresh token must be correct.
    - Post-condition: None.
    - DEP: SR9, SR10, SR12, SR11.

18. ID: SR18

    - Title: DeleteRequest
    - DESC: The server would delete a request from firebase database after processing it.
    - Input: Request id.
    - Action: Deleting request from firebase database.
    - Output: Sent Confirmation.
    - Pre-condition: Request id must be correct.
    - Post-condition: None.
    - DEP: SR9, SR10, SR12, SR11.

19. ID: SR19

    - Title: EncryptPassword
    - DESC: The server will encrypt the password of the user to prevent hacking.
    - Input: User's Password

- Action: Encrypting the user's password.
- Output: The Encrypted Password
- Pre-condition: The user must enter their password.
- Post-condition: None.
- DEP: SR10.

20. ID: SR20

    - Title: DecryptPassword
    - DESC: The server will dencrypt the password of the user to use it on further notice.
    - Input: User's Encrypted Password.
    - Action: Dencrypting User's Encrypted Password.
    - Output: User's Password
    - Pre-condition: The user must enter their password, and must be encrypted
    - Post-condition: None.
    - DEP: SR10, SR21.

3.2 Mobile

1. ID: MB1

    - Title: ResizeImage
    - DESC: The Mobile would resize an image to a specific size.
    - Input: Image, desired width and desired height.
    - Action: Resizing an image.
    - Output: Resized Image.
    - Pre-condition: Image variable in memory must not be empty and both width and height must not be negative or zero.
    - Post-condition: None.
    - DEP: None.

2. ID: MB2

    - Title: SendRequestCheckLocation
    - DESC: The Mobile would send a request to the server to get objects in a specific GPS location.
    - Input: Request code, geolocation(geohash).
    - Action: Sending a request to the server.

- Output: Server success and location of objects.
- Pre-condition: Server must be online.
- Post-condition: None.
- DEP: None.

3. ID: MB3
    - Title: SendRequestStoreObject
    - DESC: The Mobile would send a request to the server to get objects in a specific GPS location.
    - Input: Request code, geolocation(geohash)and object.
    - Action: Sending a request to the server.
    - Output: Server success and location of objects.
    - Pre-condition: Server must be online.
    - Post-condition: None.
    - DEP: None.
4. ID: MB4
    - Title: EncodeGeohash
    - DESC: The Mobile would hash GPS coordinates to a geohash.
    - Input: Latitude and longitude.
    - Action: Hashing GPS coordinates.
    - Output: geohash.
    - Pre-condition: GPS must be running and coordinates must be valid.
    - Post-condition: None.
    - DEP: None.
5. ID: MB5
    - Title: SuperimposeObject
    - DESC: The mobile would superimpose a virtual object in a specific location on the camera.
    - Input: Location as pixel(x,y).
    - Action: Superimposing a virtual object.
    - Output: Virtual object.
    - Pre-condition: Location must be within image pixels.
    - Post-condition: None.
    - DEP: None.

6. ID: MB6
    - Title: OpenCamera
    - DESC: Start the mobile's camera in video mode.
    - Input: Mobile camera.
    - Action: Starting the mobile camera.
    - Output: Video stream.
    - Pre-condition: Mobile must contain camera.
    - Post-condition: None.
    - DEP: None.
7. ID: MB7
    - Title: CaptureImageFromVideo
    - DESC: The mobile would capture a single image from the camera's video stream.
    - Input: Video.
    - Action: Capturing an image.
    - Output: Image.
    - Pre-condition: Camera video must be running.
    - Post-condition: None.
    - DEP: None.
8. ID: MB8
    - Title: DisplayMenu
    - DESC: The mobile would display a menu to the user.
    - Input: Menu object.
    - Action: Displaying UI to the user.
    - Output: Menu.
    - Pre-condition: Menu object must be initialized.
    - Post-condition: None.
    - DEP: None.
9. ID: MB9
    - Title: InitializeMenu
    - DESC: The mobile would initialize a menu's components.
    - Input: Menu object.
    - Action: Initializing a menu.

- Output: None.
- Pre-condition: None.
- Post-condition: None.
- DEP: None.

10. ID: MB10

    - Title: HideMenu
    - DESC: The mobile would hide a menu from the user.
    - Input: Menu object.
    - Action: Hiding UI from the user.
    - Output: None.
    - Pre-condition: Menu object must be initialized.
    - Post-condition: None.
    - DEP: MB9

11. ID: MB11

    - Title: AddListenter
    - DESC: The mobile would add an event listener to a menu's component.
    - Input: Menu component.
    - Action: Adding a listener to a menu's component.
    - Output: Callback function.
    - Pre-condition: Input function must be correct.
    - Post-condition: None.
    - DEP: None.

12. ID: MB12

    - Title: EditComponentText
    - DESC: The mobile would edit a component's text.
    - Input: Menu component.
    - Action: Editing a component's text.
    - Output: Component's text updated.
    - Pre-condition: Menu component must be null.
    - Post-condition: None.
    - DEP: None.

13. ID: MB13

- Title: PlayAnimation
- DESC: The mobile would play an animation of virtual object.
- Input: Virtual object animator component.
- Action: Playing an animation.
- Output: Virtual object performing animation.
- Pre-condition: Animator and virtual object must not be null.
- Post-condition: None.
- DEP: None.

14. ID: MB14
    - Title: PlayAudio
    - DESC: The mobile would play an audio file.
    - Input: Audio file and audio source component.
    - Action: Playing an audio file.
    - Output: Audio played.
    - Pre-condition: Audio file must exist and audio source must not be null.
    - Post-condition: None.
    - DEP: None.

15. ID: MB15
    - Title: GetGPSLatLong
    - DESC: The mobile would read the GPS sensor.
    - Input: GPS sensor.
    - Action: Reading the GPS sensor.
    - Output: Latitude and longitude.
    - Pre-condition: GPS sensor must be enabled.
    - Post-condition: None.
    - DEP: None.

16. ID: MB16
    - Title: GetCompassHeading
    - DESC: The mobile would read the compass sensor.
    - Input: Compass sensor.
    - Action: Reading the compass sensor.
    - Output: Compass heading.

- Pre-condition: Compass sensor must be enabled.
- Post-condition: None.
- DEP: None.

17. ID: MB17
    - Title: PollForReply
    - DESC: The mobile would poll the database continuously for a reply until received.
    - Input: Request ID.
    - Action: Polling for a reply.
    - Output: Reply received.
    - Pre-condition: Request ID must be correct.
    - Post-condition: None.
    - DEP: None.
18. ID: MB18
    - Title: StoreImageAsync
    - DESC: The mobile would store an image firebase storage asynchronously.
    - Input: Image byte data and image path in storage.
    - Action: Store image in firebase storage.
    - Output: Confirmation Sent.
    - Pre-condition: Image byte data and path must be correct.
    - Post-condition: None.
    - DEP: None.
19. ID: MB19
    - Title: ToJson
    - DESC: The mobile would convert a class object o JSON.
    - Input: Instance of class.
    - Action: Converting to JSON.
    - Output: JSON object.
    - Pre-condition: Instance must be initialized.
    - Post-condition: None.
    - DEP: None.
20. ID: MB21

- Title: StartCoroutine
- DESC: The mobile would start a coroutine.
- Input: Function.
- Action: Starting a coroutine.
- Output: Coroutine started.
- Pre-condition: None.
- Post-condition: None.
- DEP: None.

3.3 Admin

1. ID: DE1
    - Title: Place Object
    - DESC: The Admin would superimpose an object on real time objects.
    - Input: Location, SensorsReadings
    - Action: The Algorithms would work in the server and send the image into the mobile device.
    - Output: An Object
    - Pre-condition: Working Mobile Camera, Working Sensors, Internet connection.
    - Post-condition: Opening the application, connecting to internet.
    - DEP: SR1, SR2, SR3, SR4, SR5, SR6, Algo1, Algo2, Algo3, DE2, DE3, DE4, De5.
2. ID: DE2
    - Title: Get Information From the GPS.
    - DESC: The Admin would get the location of the mobile device from the sensor.
    - Input: Permissions.
    - Action: The GPS would detect the Location.
    - Output: Latitude, Longitude
    - Pre-condition: GPS must be working, The sensor should be outside.
    - Post-condition: Re-reading the location.
    - DEP: None.
3. ID: DE3
    - Title: Get Information From the Gyroscope.

- DESC: The Admin would get the orientation of the mobile device from the sensor.
- Input: Permissions.
- Action: The Gyroscope would detect the orientation.
- Output: Quaternion.
- Pre-condition: Gyroscope must be working.
- Post-condition: Re-reading the location.
- DEP: None.

4. ID: DE4

   - Title: Get Information From the compass.
   - DESC: The Admin would get the angle of the mobile device from the sensor.
   - Input: Permissions.
   - Action: The compass would detect the angle.
   - Output: Vector 3.
   - Pre-condition: compass must be working.
   - Post-condition: Re-reading the location.
   - DEP: None.

5. ID: DE5

   - Title: Get Information From the accelerometer.
   - DESC: The Admin would get the rate of change of the mobile device from the sensor.
   - Input: Permissions.
   - Action: The accelerometer would detect the rate of change.
   - Output: Vector 3.
   - Pre-condition: accelerometer must me working.
   - Post-condition: Re-reading the location.
   - DEP: None.

3.4 Player

1. ID: PL1

   - Title: Attack.
   - DESC: The player would attack an enemy, either by spells, hands or equipped items.

- Input: A Command.
- Action: the player would attack the enemy.
- Output: The enemy's health would decrease.
- Pre-condition: There must be an enemy to attack.
- Post-condition: The enemy's health decreases.
- DEP: PL4, PL5, PL6.

2. ID: PL2
    - Title: Defend.
    - DESC: The player can defend itself against the enemies attacks.
    - Input: A Command.
    - Action: The player would be in a defensive position.
    - Output: The player defended itself.
    - Pre-condition: The enemy attacking.
    - Post-condition: According to the equipped item or used item the player would decrease its health or not.
    - DEP: PL4, PL5, PL6.
3. ID: PL3
    - Title: Flee.
    - DESC: The player can flee from a combat.
    - Input: A Command.
    - Action: The player fleeing.
    - Output: The player no longer in combat.
    - Pre-condition: The enemy's health should be low.
    - Post-condition: The enemy not following the player.
    - DEP: PL1, PL2.
4. ID: PL4
    - Title: Use An Item.
    - DESC: The player can consume an item like a potion or a poison on an enemy.
    - Input: A Command.
    - Action: The player consuming an item.
    - Output: The player has a Buff on him, or the enemy has a DeBuff on him.

- Pre-condition: The player must have the specific item to use.
- Post-condition: None.
- DEP: PL7.

5. ID: PL5
    - Title: Equip An Item.
    - DESC: The player can equip an item to use like a weapon or an armor.
    - Input: The item to Equip.
    - Action: The player equipping the selected item.
    - Output: The player equipped the item.
    - Pre-condition: The player must have the item to equip it.
    - Post-condition: The player's rating increases.
    - DEP: PL7.

6. PL6
    - Title: UnEquip An Item.
    - DESC: The player can unequip an item to use like a weapon or an armor.
    - Input: The item to unEquip.
    - Action: The player unequipping the selected item.
    - Output: The player unequipped the item.
    - Pre-condition: The player must have an item equipped to unequip it.
    - Post-condition: The player's rating decreases.
    - DEP: PL7.

7. PL7
    - Title: Craft An Item.
    - DESC: The player can craft an item to use. Like a potion, a poison, weapon or an armor.
    - Input: Materials
    - Action: The player crafting the selected item.
    - Output: Crafted item in inventory.
    - Pre-condition: The player must have the necessary materials to craft the item.
    - Post-condition: There must be a space in the inventory.

- DEP: None.

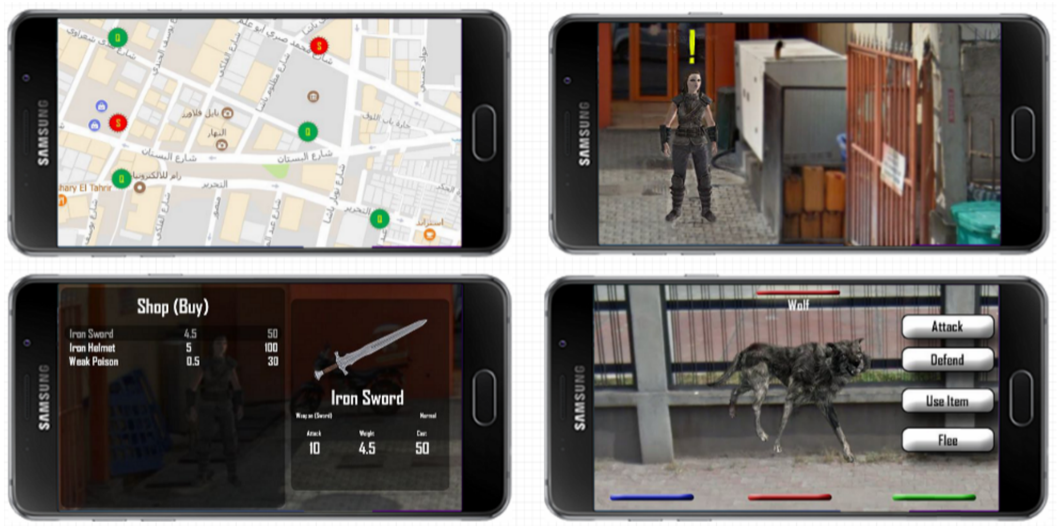
8. PL8

   - Title: Notify Mission Update.
   - DESC: If the player finishes or failed a mission, there will be an alarm for them.
   - Input: Quest, Quest Status.
   - Action: Checking the mission status.
   - Output: Notification for the quest status.
   - Pre-condition: The player must have taken a quest.
   - Post-condition: None.
   - DEP: None.

4 Interface Requirements

4.1 User Interfaces

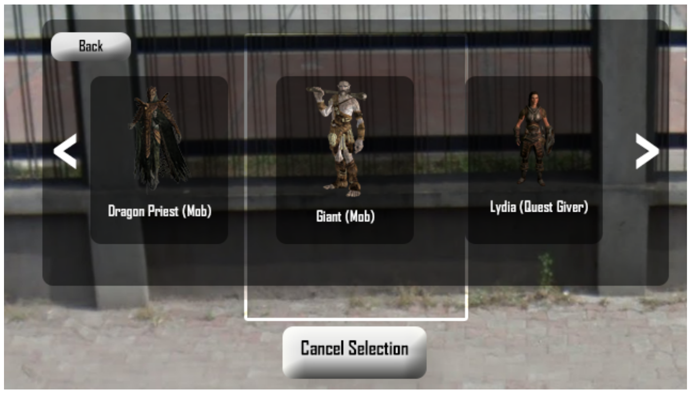
4.1.1 Player Primitive Graphical User Interface



Iron Sword	4.5	50
Iron Helmet	5	100
Weak Poison	0.5	30

Attack	Weight	Cost
10	4.5	50

4.1.2 Admin Primitive Graphical User Interface



4.1.3 API

1. Pyrebase: is a Firebase wrapper for python.

5 Performance Requirements

For the data to be uploaded from the server to the smartphone to be process will take about 2 seconds. for the information to be sent to the smartphone would take about 0.5 seconds.

6 Other non-functional attributes

6.1 Usability

As the concept of augmented is new, the game must be easy to use and intuitive. The wanderer would use the unity Graphical User Interface to help the player get around the game much easier. Unity GUI allows the developer to clarify to the user the use of their system, allowing to customize the GUI accordingly would make it for the user to easily grasp the logic of the system. (MB8, MB9, MB10, MB11, MB12, MB13, MB14)

6.2 Resource Utilization

Must be a priority as mobile devices resource are scarce and must be utilized with caution. That most of the computations would be on a separate server, rather than on the mobile phone. (Server Side in Functional Requirements)

6.3 Security

The game must have an encryption key for the passwords of the users, to protect their privacy. The system is equipped with the encryption and decryption of the user's password (SR20, SR21).

6.4 Scalability

The system can increase some features like spells, effects or quests without changing the structure of the project. The system can change the algorithms the system uses. (IAlgorithms, ICombat)

7 Preliminary Object-Oriented Domain Analysis

7.1 Class descriptions

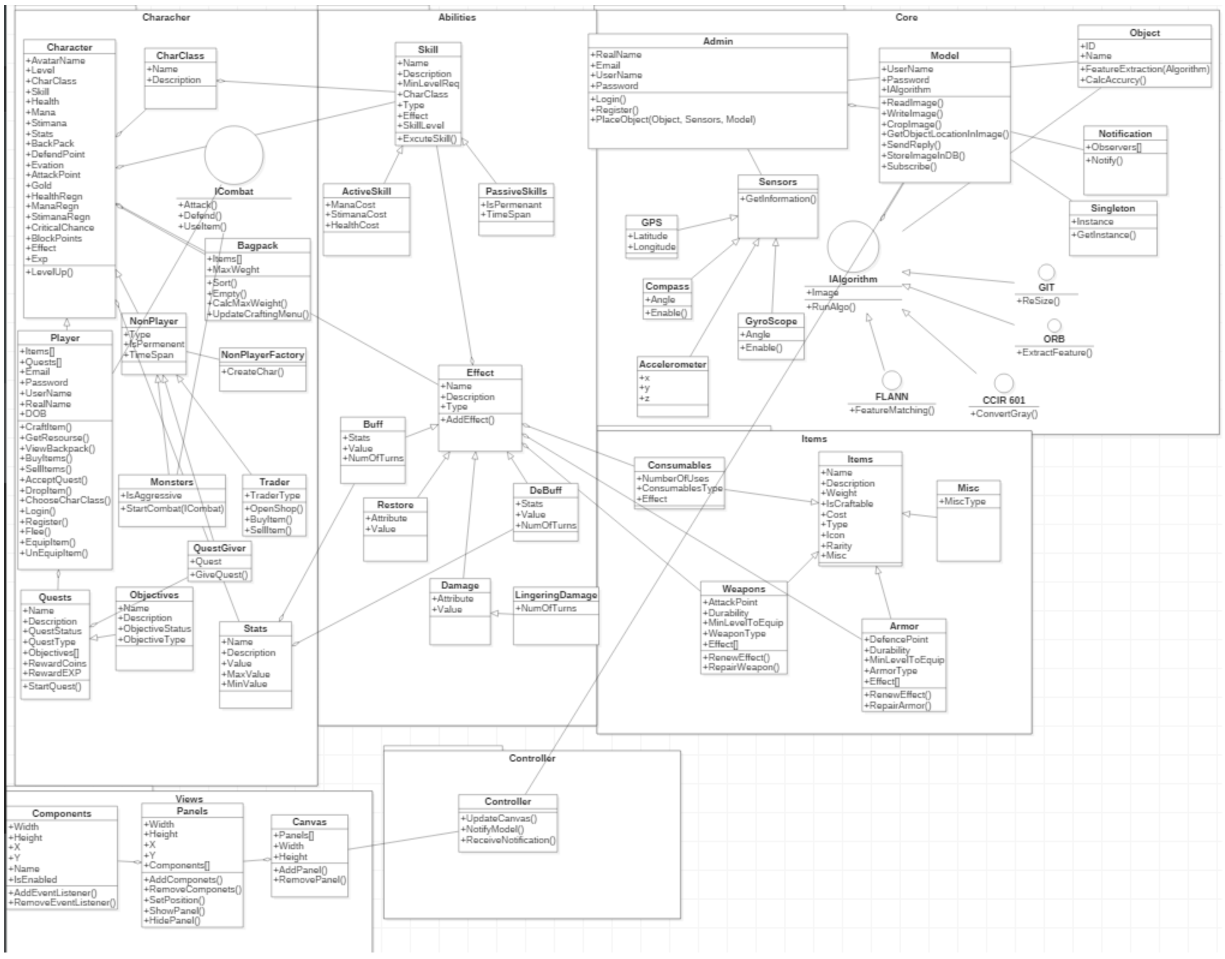


Figure 2: Primitive Class Diagram

7.2 Interface Classes

7.2.1 ICombat

Purpose: An interface that handles all the main attributes in the combat mechanism in that game. It includes attacking, defending and using item to Buff or DeBuff.

7.2.2 IAlgorithm

Purpose: An interface that handles all the algorithms that the server would use to convert to gray scale, resize an image, running feature extraction and comparing the feature extracted image.

7.3 Design Patterns

7.3.1 Singleton Design Pattern

A software design pattern that restricts the instantiate of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system. (Singleton Class)

7.3.2 Factory Design Pattern

A software design pattern that is used when a method returns one of several possible classes that share a common super class. Our project uses the factory pattern to return one of the non player characters. (NonplayerFactory Class)

7.3.3 Observer Design Pattern

A software design pattern in which an object maintains a list of its dependents and notifies them automatically of any state changes. (Notification Class)

7.3.4 Strategy Design Pattern

A software design pattern that enables selecting an algorithm at runtime. Our project uses the strategy pattern to select from one algorithm to another. (IAlgorithm Interface, ICombat Interface)

7.4 Class Descriptions

7.4.1 Character

1. Class Name: Character
2. List Of Superclasses: None.
3. List of Subclasses: Player, NonPlayer

4. Purpose: This class's purpose is to handle any character in the game's information.
5. Collaborations: Charclass, Skill, Player, NonPlayer, Effect, Stats.
6. Attributes: AvatarName, Level, CharClass, Skill, Health, Mana, stamina, Stats, Backpack, DefendPoint, Evation, AttackPoink, Gold, HealthRegn, ManaRegn, staminaRegn, CriticalChance, BlockPoints, Effect, Exp.
7. Operations: LevelUp(Skill)

7.4.2 Player

1. Class Name: Player
2. List Of Superclasses: Character
3. List of Subclasses: None.
4. Purpose: This class contains the user's progress in the game.
5. Collaborations: ICombat, Quests.
6. Attributes: Items, Quests, Email, Password, Username, Realname, DOB.
7. Operations: CraftItem(Item), GetResources(), ViewBackpack(BagPack), BuyItems(Trader), SellItems(Trader), AcceptQuest(Quest), DropItem(Item), ChooseCharClass(Charclass), Login(Username, password), Register(Email, Password, Username, Realname, DOB), Flee(), EquipItem(Item), UnEquipItem(Item).

7.4.3 NonPlayer

1. Class Name: NonPlayer.
2. List Of Superclasses: Character.
3. List of Subclasses: Trader, Monsters, QuestGiver.
4. Purpose:This class contains all the NPCs information and their functionality.
5. Collaborations: NonPlayerFactory.
6. Attributes: Type, Ispermenent, Timespan.
7. Operations: None.

7.4.4 NonPlayerFactory

1. Class Name: NonPlayerFactory.
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: This class is the design pattern factory that allows the admin to create more instants of the nonplayer class.
5. Collaborations: NonPlayer.
6. Attributes: None.
7. Operations: CreateChar.

7.4.5 Monster

1. Class Name: Monster
2. List Of Superclasses: Nonplayer.
3. List of Subclasses: None.
4. Purpose: This class contains the monster's information and type.
5. Collaborations: ICombat.
6. Attributes: IsAggresive.
7. Operations: StartCombat(ICombat).

7.4.6 Trader

1. Class Name: Trader
2. List Of Superclasses: NonPlayer
3. List of Subclasses: None.
4. Purpose: This class can initiate trading system with the user.
5. Collaborations: Items
6. Attributes: TraderType.
7. Operations: OpenShop(), BuyItems(Items), SellItems(Items).

7.4.7 QuestGiver

1. Class Name: QuestGiver
2. List Of Superclasses: NonPlayer
3. List of Subclasses: None.
4. Purpose: The NPC gives quests to the player to finish them.
5. Collaborations: Quests.
6. Attributes: Quest.
7. Operations: GiveQuest(Quest).

7.4.8 Quest

1. Class Name: Quest
2. List Of Superclasses: None.
3. List of Subclasses: Objectives.
4. Purpose: This class contains the Quests Information.
5. Collaborations: QuestGiver.
6. Attributes: Name, Description, QuestStatus, QuestType, Objectives, RewardCoins, RewardExp.
7. Operations: StartQuest(QuestGiver).

7.4.9 Objectives

1. Class Name: Objectives
2. List Of Superclasses: Quests.
3. List of Subclasses: None.
4. Purpose: This class contains the missions inside the quests given.
5. Collaborations: None.
6. Attributes: Name, Description, ObjectiveStatus, ObjectiveType.
7. Operations: None.

7.4.10 Stats

1. Class Name: Stats
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: This class contains all the special abilities that the player gains. Like crafting new items or increasing the backpack's limit.
5. Collaborations: Buff, DeBuff.
6. Attributes: Name, Description, Value, MaxValue, MinValue.
7. Operations: None.

7.4.11 Bagpack

1. Class Name: Bagpack
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: This class allows the user to carry items and equipment.
5. Collaborations: Character.
6. Attributes: Items, MaxWeight.
7. Operations: Sort(), Empty(), CalcMaxWeight(), UpdateCraftingMenu(Items).

7.4.12 Charclass

1. Class Name: Charclass
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: This class specify the player's class. For example, A mage class, A Warrior Class or A Hunter Class.
5. Collaborations: Character.
6. Attributes: Name, Description.
7. Operations: None.

7.4.13 ICombat

1. Class Name: ICombat.
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: This interface initiates the player's comabt mechanism, like attacking.
5. Collaborations: Player.
6. Attributes: None.
7. Operations: Attack(), Defend(), UseItem().

7.4.14 Skill

1. Class Name:Skill
2. List Of Superclasses: None.
3. List of Subclasses: ActiveSkill, PassiveSkill.
4. Purpose: This class's purpose is to map the player's skills, which will help them with surviving the game.
5. Collaborations: Character, Charclass, Effect.
6. Attributes: Name, Description, MinLevelReq, Charclass, Type, Effect, SkillLevel.
7. Operations: ExcuteSkill().

7.4.15 ActiveSkill

1. Class Name: ActiveSkill.
2. List Of Superclasses: Skill.
3. List of Subclasses: None.
4. Purpose: This Class affect the player's and enemy's health, mana and Stamina. Either by buffing or debuffing.
5. Collaborations: Charclass, Effect.
6. Attributes: ManaCost, StaminaCost and HealthCost.
7. Operations: None.

7.4.16 PassiveSkill

1. Class Name: PassiveSkill.
2. List Of Superclasses: Skill.
3. List of Subclasses: None.
4. Purpose: This class will help the player gain more abilities to help advance the player.
5. Collaborations: None.
6. Attributes: IsPermenant, TimeSpan.
7. Operations: None.

7.4.17 Effect

1. Class Name: Effect.
2. List Of Superclasses: None.
3. List of Subclasses: Buff, Debuff, Restore, Damage.
4. Purpose: This class puts special effect on weapons and armors that either will help the player gain special status or will put the enemy at a temporary disadvantage.
5. Collaborations: Consumables, Weapons, Armor, Character.
6. Attributes: Name, Description, Type.
7. Operations:AddEffect(Item).

subsubsectionBuff

1. Class Name: Buff.
2. List Of Superclasses: Effect.
3. List of Subclasses: None.
4. Purpose: This class will have the specific effects which buffs the character's Stat.
5. Collaborations: Stat.
6. Attributes: Stats, Value, NumberOfTurns.
7. Operations: None.

7.4.18 Restore

1. Class Name: Restore.
2. List Of Superclasses: Effect.
3. List of Subclasses: None.
4. Purpose: This class will have the specific effects which restores the character's attributes.
5. Collaborations: None.
6. Attributes: Attribute, Value.
7. Operations: None.

7.4.19 DeBuff

1. Class Name: DeBuff.
2. List Of Superclasses: Effect.
3. List of Subclasses: None.
4. Purpose: This class will have the specific effects which debuffs the character's Stat.
5. Collaborations: Stat.
6. Attributes: Stats, Value, NumberOfTurns.
7. Operations: None.

7.4.20 Damage

1. Class Name: Damage.
2. List Of Superclasses: Effect.
3. List of Subclasses: LingeringDamage.
4. Purpose: This class will have the specific effects which damages the character's attributes.
5. Collaborations: None.
6. Attributes: Attribute, Value.
7. Operations: None.

7.4.21 LingeringDamage

1. Class Name: LingeringDamage.
2. List Of Superclasses: Damage.
3. List of Subclasses: None.
4. Purpose: This class will have the specific effects which damages the character's attributes over time (turns).
5. Collaborations: None.
6. Attributes: NumOfTurns.
7. Operations: None.

7.4.22 Items

1. Class Name: Items.
2. List Of Superclasses: None.
3. List of Subclasses: Consumables, Weapons, Armor, Misc.
4. Purpose: This class will contain all the information needed for the items in the game.
5. Collaborations: None.
6. Attributes: Name, Description, Weight, IsCraftable, Cost, Type, Icon, Rarity, Misc.
7. Operations: None.

7.4.23 Consumables

1. Class Name: Consumables.
2. List Of Superclasses: Items.
3. List of Subclasses: None.
4. Purpose: This class will contain all the specific information needed for the consumables.
5. Collaborations: Effect.
6. Attributes: NumberOfUses, ConsumablesType, Effect.
7. Operations: None.

7.4.24 Weapons

1. Class Name: Weapons.
2. List Of Superclasses: Items.
3. List of Subclasses: None.
4. Purpose: This class will contain all the specific information needed for the weapons.
5. Collaborations: Effect.
6. Attributes: AttackPoint, Durability, MinLevelToEquip, WeaponType, Effect.
7. Operations: RenewEffect(), RepairWeapon().

7.4.25 Armor

1. Class Name: Armor.
2. List Of Superclasses: Items.
3. List of Subclasses: None.
4. Purpose: This class will contain all the specific information needed for the armors.
5. Collaborations: Effect.
6. Attributes: DefensePoint, Durability, MinLevelToEquip, ArmorType, Effect.
7. Operations: RenewEffect(), RepairArmor().

7.4.26 Misc

1. Class Name: Misc.
2. List Of Superclasses: Items.
3. List of Subclasses: None.
4. Purpose: This class will contain all the specific information needed for the misc items.
5. Collaborations: None.
6. Attributes: MiscType.
7. Operations: None.

7.4.27 Admin

1. Class Name: Admin
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: The Admin has the ability to Place An object in a certain place. This object can be a quest or a monster.
5. Collaborations: Sensors, Model, Object.
6. Attributes: RealName, Email, Username, Password.
7. Operations:Login(Username, Password), Register(Username, Password, RealName, Email), PlaceObject(Sensors, ImageClassificationSystem, Object)

7.4.28 Sensors

1. Class Name: Sensors
2. List Of Superclasses: None.
3. List of Subclasses: GPS, Compass, GyroScope, Acclermeter.
4. Purpose: The Sensors Class is aimed to collect the information of the sensors on the mobile Phone
5. Collaborations: Admin.
6. Attributes: None.
7. Operations: GetInformation().

7.4.29 GPS

1. Class Name: GPS
2. List Of Superclasses: Sensors.
3. List of Subclasses: None.
4. Purpose: The GPS detects the location of the mobile phone.
5. Collaborations: None.
6. Attributes: Latitude, Longitude.
7. Operations: None.

7.4.30 Compass

1. Class Name: Compass
2. List Of Superclasses: Sensors.
3. List of Subclasses: None.
4. Purpose: The compass captures the orientation of the mobile phone. Like the North, South, East or West.
5. Collaborations: None.
6. Attributes: Angle.
7. Operations: Enable().

7.4.31 Accelerometer

1. Class Name: Accelerometer
2. List Of Superclasses: Sensors.
3. List of Subclasses: None.
4. Purpose: The Accelerometer detects the change of movement of the mobile phone.
5. Collaborations: None.
6. Attributes: x, y, z.
7. Operations: None.

7.4.32 GyroScope

1. Class Name: GyroScope
2. List Of Superclasses: Sensors.
3. List of Subclasses: None.
4. Purpose: The GyroScope detects the orientation and the angle of the mobile phone according to the x, y and z axis.
5. Collaborations: None.
6. Attributes: Angle.
7. Operations: Enable().

7.4.33 Model

1. Class Name: Model
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: This class focuses on the conversion of the sent image from the mobile phone to be handled and preform the algorithms. This class is the intermediate between the Admin and the mobile phone.
5. Collaborations: Admin, IAlgorithm, Object, Notification, Singleton.
6. Attributes: Username, Password, IAlgorithm.
7. Operations: ReadImage(Image), WriteImage(Image), CropImage(Image), GetObjectLocationInImage(), SendReply(), StoreImageInDB(), Subscribe(Notification).

7.4.34 Object

1. Class Name: Object.
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: The object is the augmented object. For example the quests, shops or monsters.
5. Collaborations: Admin, IAlgorithms.
6. Attributes: ID, Name.
7. Operations: FeatureExtraction(IAlgorithms), CalcAccurcy().

7.4.35 Notification

1. Class Name: Notification.
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: The notification notifies the ImageClassificationSystem that there is a change in the database.
5. Collaborations: Model.
6. Attributes: Observers.
7. Operations: Notify().

7.4.36 SingleTon

1. Class Name: SingleTon
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: The SingleTon is a software design pattern that restricts the instantiate of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system.
5. Collaborations: Model.
6. Attributes: Instance.
7. Operations: GetInstance().

7.4.37 Components

1. Class Name: Components
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: The Components provide a unified class for creating UI Elements.
5. Collaborations: Panels.
6. Attributes: Width, Height, X, Y, Name, IsEnabled.
7. Operations: AddEventListener(), RemoveEventListener().

7.4.38 Panels

1. Class Name: Panels
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: The Panels are containers of group of components that forms a single window.
5. Collaborations: Canvas.
6. Attributes: Width, Height, X, Y, Components.
7. Operations: AddComponents(Components), RemoveComponents(Components), SetPosition(X, Y), ShowPanel(Panels), HidePanel(Panels).

7.4.39 Canvas

1. Class Name: Canvas.
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: The canvas contains the panels and collects them into one window.
5. Collaborations: Controller.
6. Attributes: Panels, Width, Height.
7. Operations: AddPanels(Panels), HidePanels(Panels).

7.4.40 Controller

1. Class Name: Controller.
2. List Of Superclasses: None.
3. List of Subclasses: None.
4. Purpose: This class handles the information that has been updated from the package View and sends them to the Model class. The updates from the Model class get sent to the package View to view them to the player.
5. Collaborations: Canvas, Model.
6. Attributes: None.
7. Operations: UpdateCanvas(Canvas), NotifyModel(Model), ReciveNotification(Model).

8 Operational Scenarios

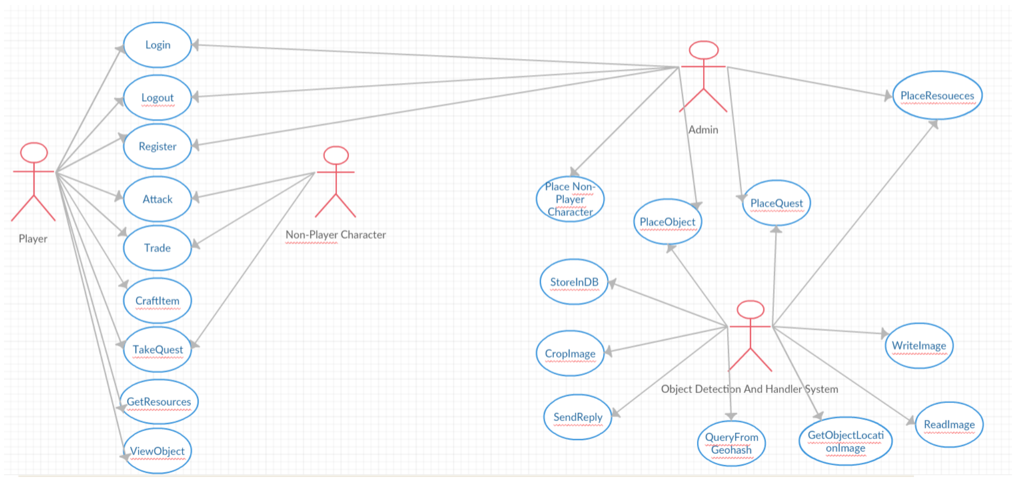
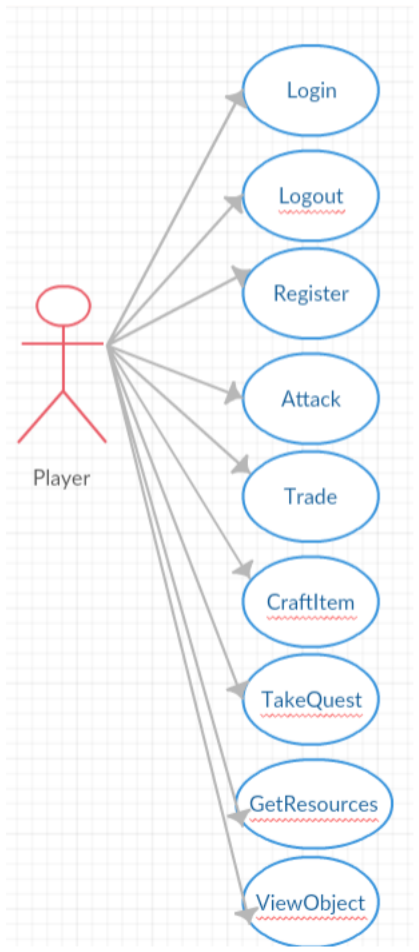


Figure 3: Use Case Diagram

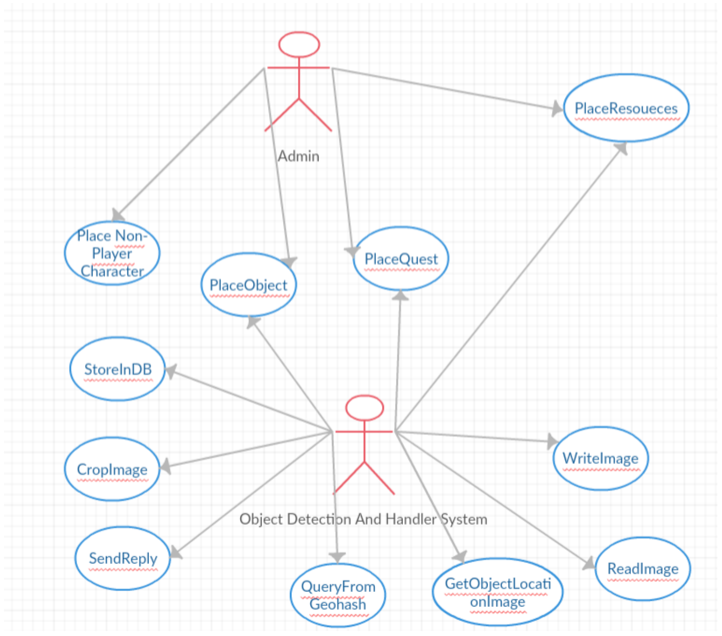
8.1 Scenario 1: Player's Scenarios



In this scenario, the player is able to preform several actions that will allow them to play the game at it's maximum abilities.

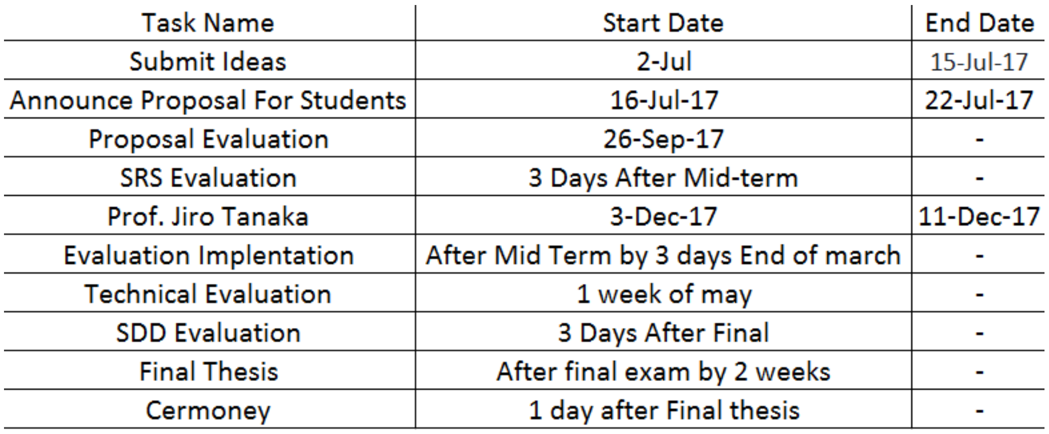
1. The player can Login into our game.
2. The player Register into our game.
3. The player can attack a NPCs.
4. The player can Trade Items With NPCs.
5. The player can Craft items to use later on.
6. The player can Accept Quests from NPCs and go on adventure.
7. The player can Get resources to craft items.

8.2 Scenario 2: Server's Scenarios



The Image Classification System's main function is to process images and apply the object recognition algorithm to superimpose objects into real time objects. The server first read the image from the database, apply resize algorithm on the image, and then compare the featured extracted image to the images in the database. If there was a match, that means that there are objects in here, and then return array of objects to the database to serve to the mobile device. If there wasn't a match, then the server asks the mobile device to capture the environment and save it in the database.

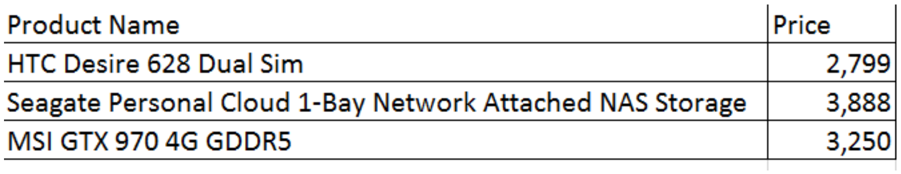
9 Preliminary Schedule Adjusted



Task Name	Start Date	End Date
Submit Ideas	2-Jul	15-Jul-17
Announce Proposal For Students	16-Jul-17	22-Jul-17
Proposal Evaluation	26-Sep-17	-
SRS Evaluation	3 Days After Mid-term	-
Prof. Jiro Tanaka	3-Dec-17	11-Dec-17
Evaluation Implentation	After Mid Term by 3 days End of march	-
Technical Evaluation	1 week of may	-
SDD Evaluation	3 Days After Final	-
Final Thesis	After final exam by 2 weeks	-
Cermoney	1 day after Final thesis	-

Figure 4: Preliminary Schedule Table.

10 Preliminary Budget Adjusted



Product Name	Price
HTC Desire 628 Dual Sim	2,799
Seagate Personal Cloud 1-Bay Network Attached NAS Storage	3,888
MSI GTX 970 4G GDDR5	3,250

Figure 5: Preliminary Budget Table.

11 Appendices

11.1 Definitions, Acronyms, Abbreviations

1. Firebase: A mobile and web application platform.
2. OpenCv: a library of programming functions mainly aimed at real-time computer vision.
3. Unity Game Engine: a cross-platform game engine, which is primarily used to develop video games and simulations for computers, consoles and mobile devices.
4. Geolocation: estimation of the real-world geographic location of an object.
5. RPG: role-playing game is a game in which players assume the roles of characters in a fictional setting.
6. NPC: Non player character.

7. Buff: a temporary beneficial status effect in some video games on a character or an enemy.

8. DeBuff: a temporary detrimental status effect in some video games on a character or an enemy.

9. Lure Modules: items players can use at a Pokestop to increase Pokemon Spawn Rates.

12 References

[1] Tateno, Masaru, et al. "New game software (Pokmon Go) may help youth with severe social withdrawal, hikikomori." Psychiatry research 246 (2016): 848-849.

[2] Althoff, Tim, Ryen W. White, and Eric Horvitz. "Influence of Pokmon Go on physical activity: study and implications." Journal of medical Internet research 18.12 (2016).

[3] Denham, Jess (July 12, 2016). "Pokmon Go has won the praise of gender fluid gamers". Archived from the original on July 15, 2016. Retrieved July 19, 2016

[4]Cosimano, Mike (July 12, 2016). "Review: Pokemon Go". Destructoid. Archived from the original on July 17, 2016. Retrieved July 12, 2016.

[5] Concepcion, Miguel (July 12, 2016). "Pokemon GO Review". GameSpot. Archived from the original on July 17, 2016. Retrieved July 12, 2016.

[6] Shunichi Kasahara , Valentin Heun , Austin S. Lee , Hiroshi Ishii, Second surface: multi-user spatial collaboration system based on augmented reality, SIGGRAPH Asia 2012 Emerging Technologies, p.1-4, November 28-December 01, 2012, Singapore, Singapore