Software Design Specification
For

Pokémon Go Back

Prepared by:

Name	ID
AmittBhardwj	40037899
Gaganjot Kaur	40037933
Gunpreet Ahuja	40041311
Gurpreet Kaur Lotey	40038315
Gurpreet Raju	40013682
Harinder Kaur	40041313
Mandeep Singh	40040964
Navjot Singh	40014477
Navrang Pal Kaur Dhaliwal	40047555
Simranjit Singh Bachher	40040601

Instructor Name -: Stuart Thiel

Table of Contents

1 Introduction

The purpose of this document is to define and communicate the software requirements of “Pokémon Go Back”. The requirements are documented as a means to provide a common understanding to stakeholders. The requirements will be verified through reviews. The structure of this software specification is inspired by IEEE standard 830-1998.

1.1 Project Overview

Pokémon Go Back is a standalone application which is based on Pokémon Go which is a trading card game. It is a game developed in single player mode and the game is played between a user and the AI. The players act as Pokémon trainers and they battle their Pokémons with special abilities and powers t o reach the final winning stage.

1.2 Project Scope

Pokémon Go Back is an interactive card game with a graphical user interface. The goal is to develop a card game using the software engineering processes. The game which is developed does not need any special expertise or skills. The developed game will be tested with various test cases and techniques. The basic purpose of the game is entertainment.

1.3 Document Preview

Pokémon go back is a standalone application which is developed in Java programming language by using Javafx platform. Pokémon go back is a windows based application which does not require any special environment. Pokémon go back is a single player game which can be played between user and AI. User and AI will play their turns to win the game.

1.4 References

- http://cscie12.dce.harvard.edu/lecture_notes/2010/20100421/slide43.html
- https://dzone.com/articles/composite-design-pattern-java
- http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/mvc.html

2 Design

2.1 Architecture Design

The architecture taken for Pokémon Go Back game is the Model View Controller model. The MVC architecture is consists of 3 separate components called model, view and controller.

The model is base of the game where the game's state and player data is managed. All of the computations that are performed during the game are done in the model component as well as all data that needs to be processed. In this Pokémon game model is a package and there are many classes comes under the model like: CardsGroup.java, CardsItem .java, AbilityParser.java, Pokemon.java,Shuffle.java, Turn.java

The view is the graphical user interface (GUI) of the game and displays the data from the model. View includes classes:DialogBoxHandler.java, GeneralCard.java, PokemonCard.java

The controller is the component that players use to interact with the game. All players data is captured through the controller, which it then passes to the model. The controller also translate all mouse clicks or game events and it decide which part of the model needs to be manipulated. Controller contains GameController.java.

The main purpose why we chose the MVC model for the Pokemon Go Back game is that it allows the GUI to be separated from the core application. This flexibility of architecture allows the game's core to be developed by the implementation team members and the GUI to be designed by the design team members . Both team members can work in parallel and easily cooperate on any changes that need to be done.

2.1.1 General Constraints

In Model View Controller architecture there are no specific constraints for game. Pokémon Go Back is a standalone application.

2.1.2 Software Architecture Diagram

This game uses MVC model and components of MVC model are shown below in the form of high-level class diagram. Player interacts directly with the View and the Controller, and Controller manipulates Model and Model updates.

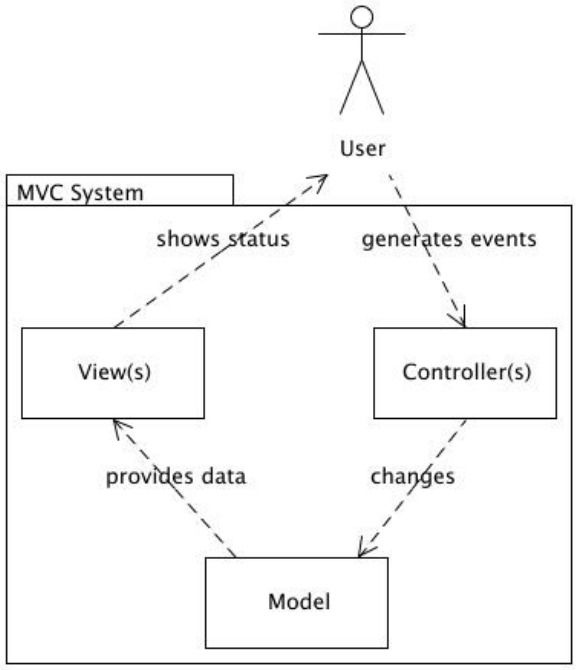


Fig:2.1 Model View Controller

2.2 Design Overview

2.2.1 Class Model

A class model is consist of one or more class diagrams and the supporting specifications that describe model elements including classes, attributes, methods and relationships between classes, and interfaces.

(For high-resolution view of diagrams please check **Documents/Images/**folder.)

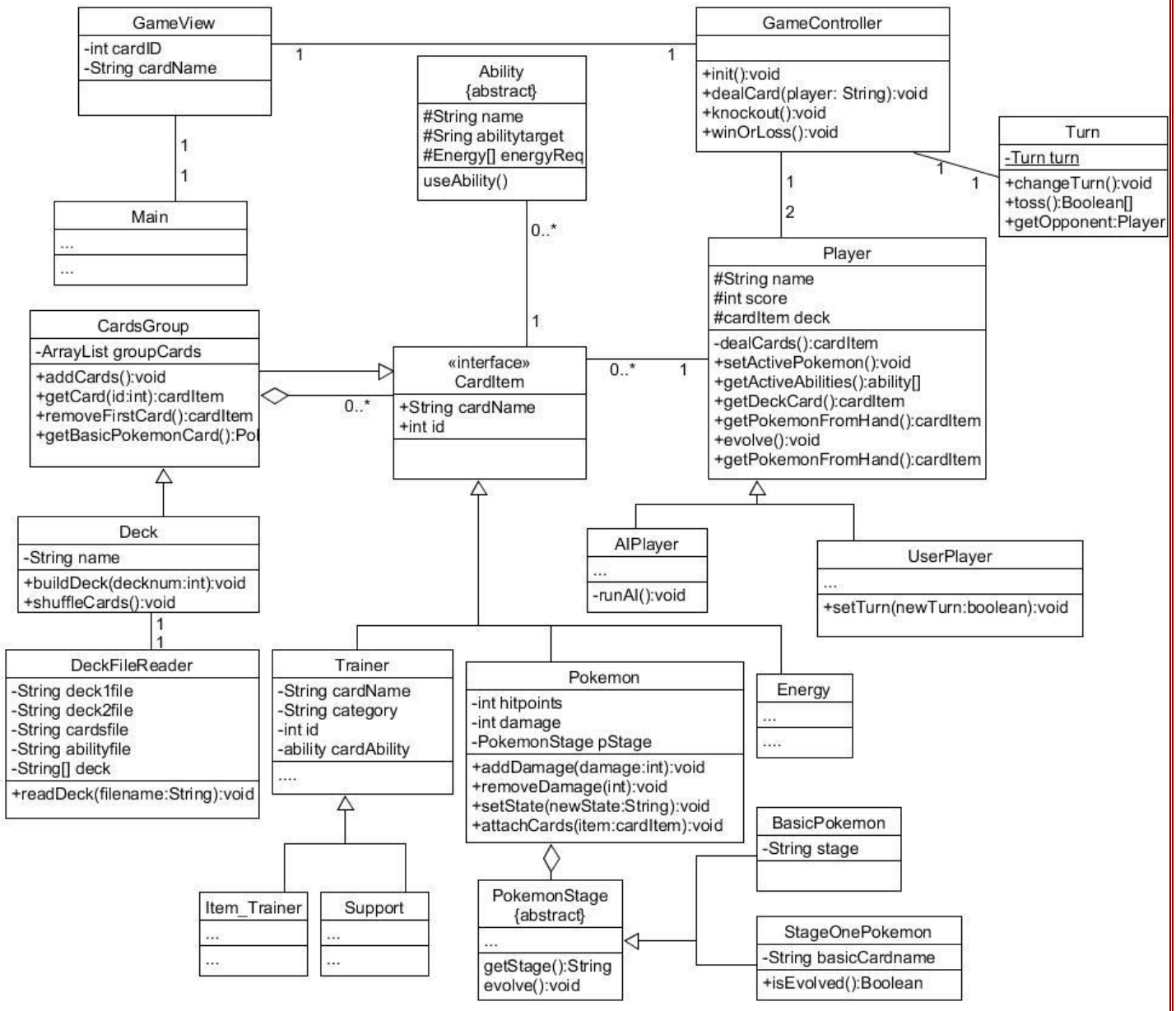


Fig: 2.2 Class Diagram

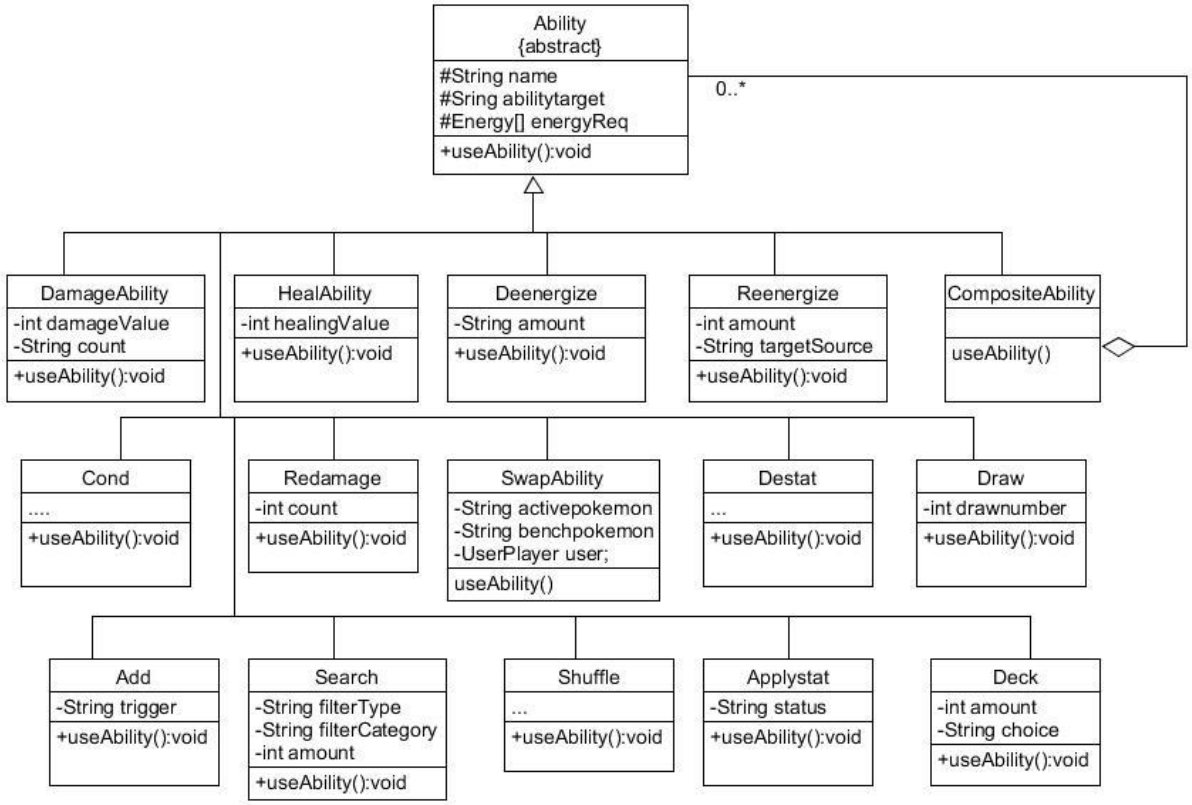


Fig 2.3 Class Diagram Of abilities

In Pokemon Go Back designing structure there are Composite pattern and Strategy patterns applied .Composite design pattern allows tree structure to represent all hierarchy . In class diagram CardItem class is component and Cardsgroup is composite. The Strategy pattern provides facility to dynamically swap out algorithms at runtime

2.2.2 Sequence Diagram

In order to depict design interaction between the system different modules and to show major functionalities in better way we used the sequence diagrams. We choose sequence diagrams because they can show interactions and calls between objects and classes appropriately. All of the following sequence diagram uses GRASP patterns. We focused that system should follow low coupling and high cohesion software practices.

2.2.2.1 Game Setup (represents UC1,For more detail refer to SRS document)

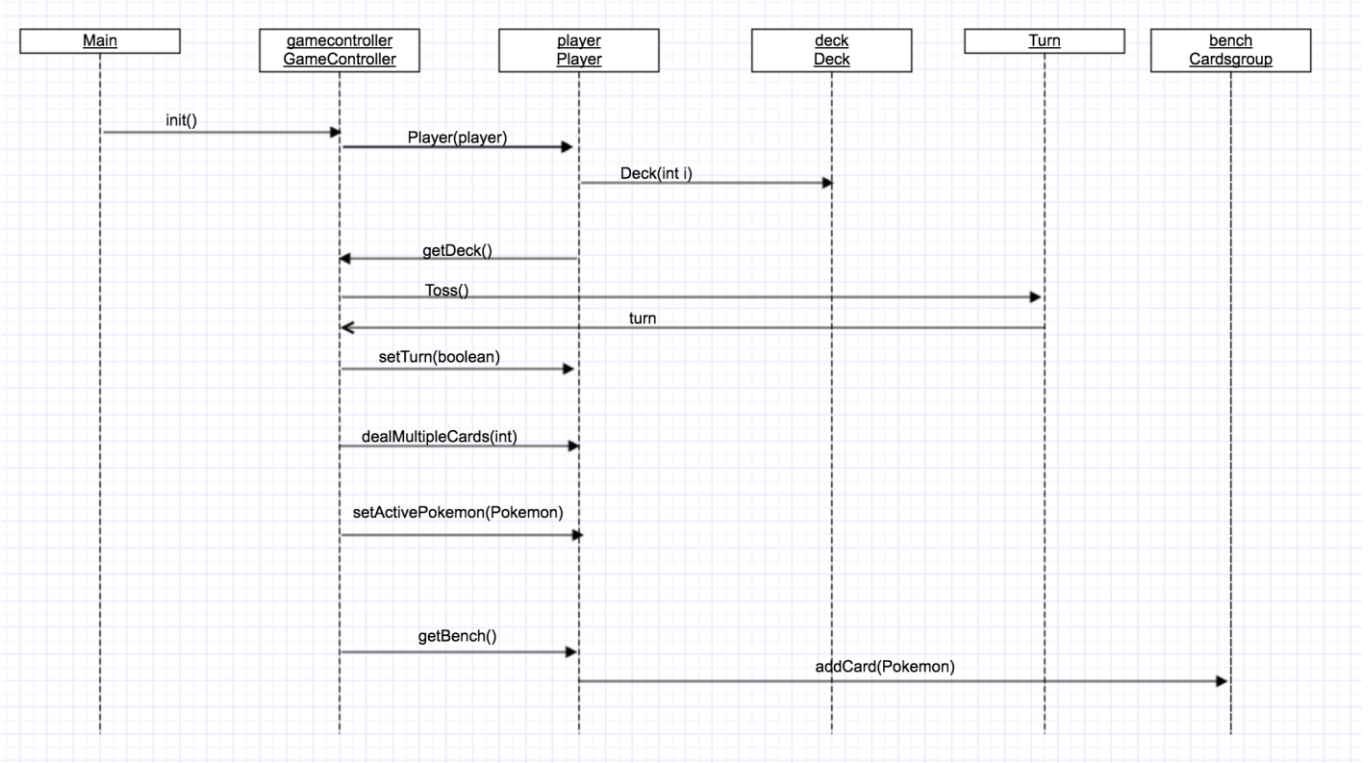


Fig 2.4: Game Setup

2.2.2.2 Play Turn (represents UC2,For more detail refer to SRS document)

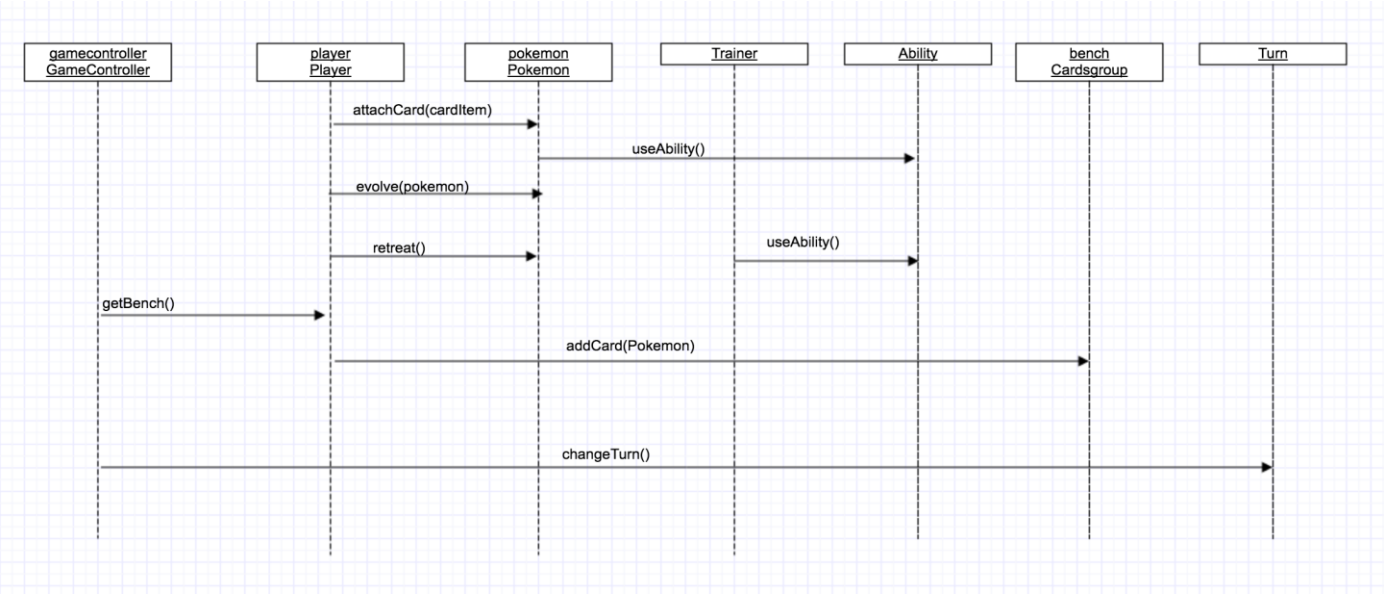


Fig: 2.5 Play Turn

2.2.2.3 Evolve Pokémon (represents UC6 ,For more detail refer to SRS document)

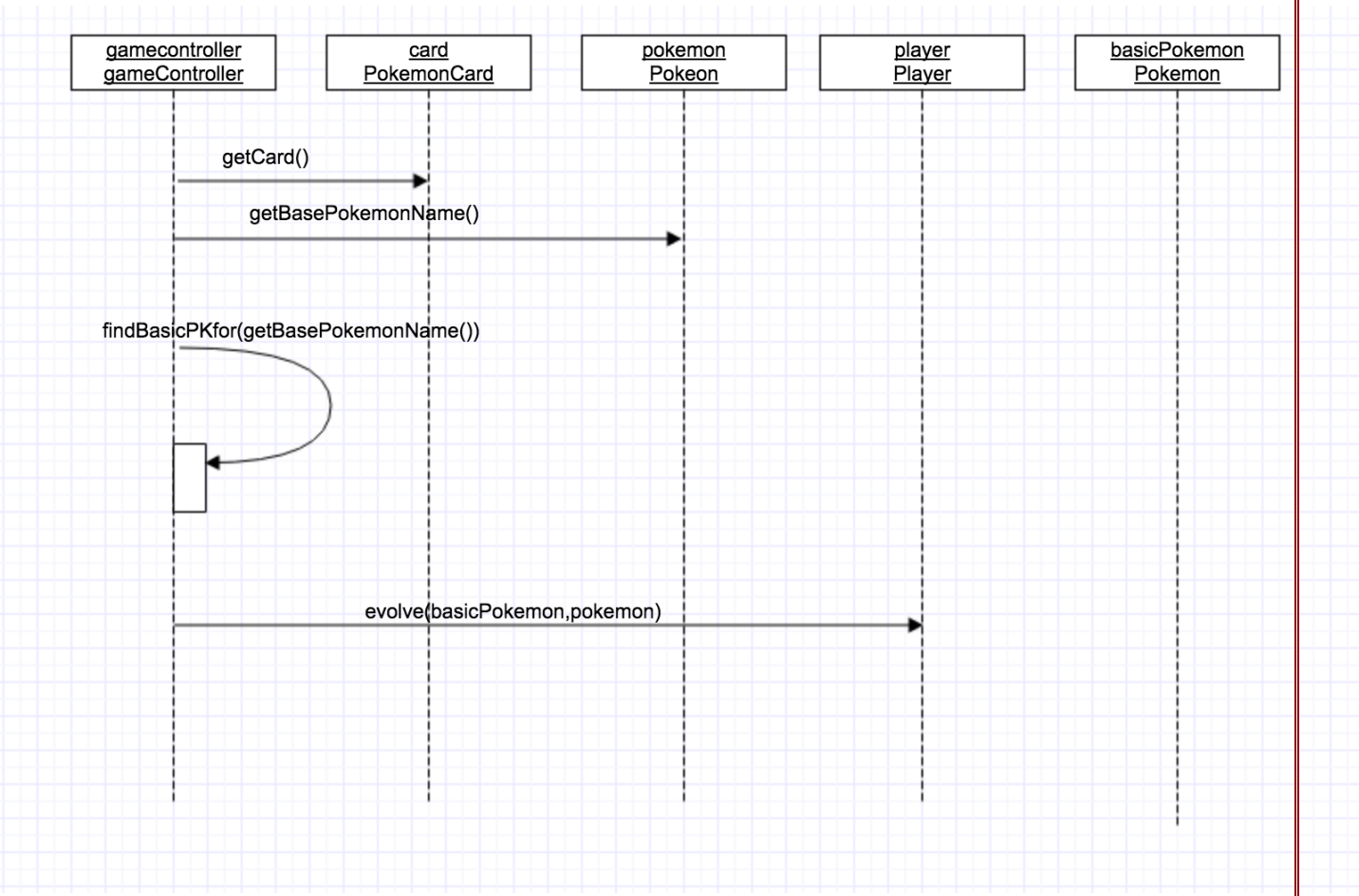


Fig: 2.6 Evolve Pokemon

2.2.2.4 Retreat Pokémon (represents UC7 ,For more detail refer to SRS document)

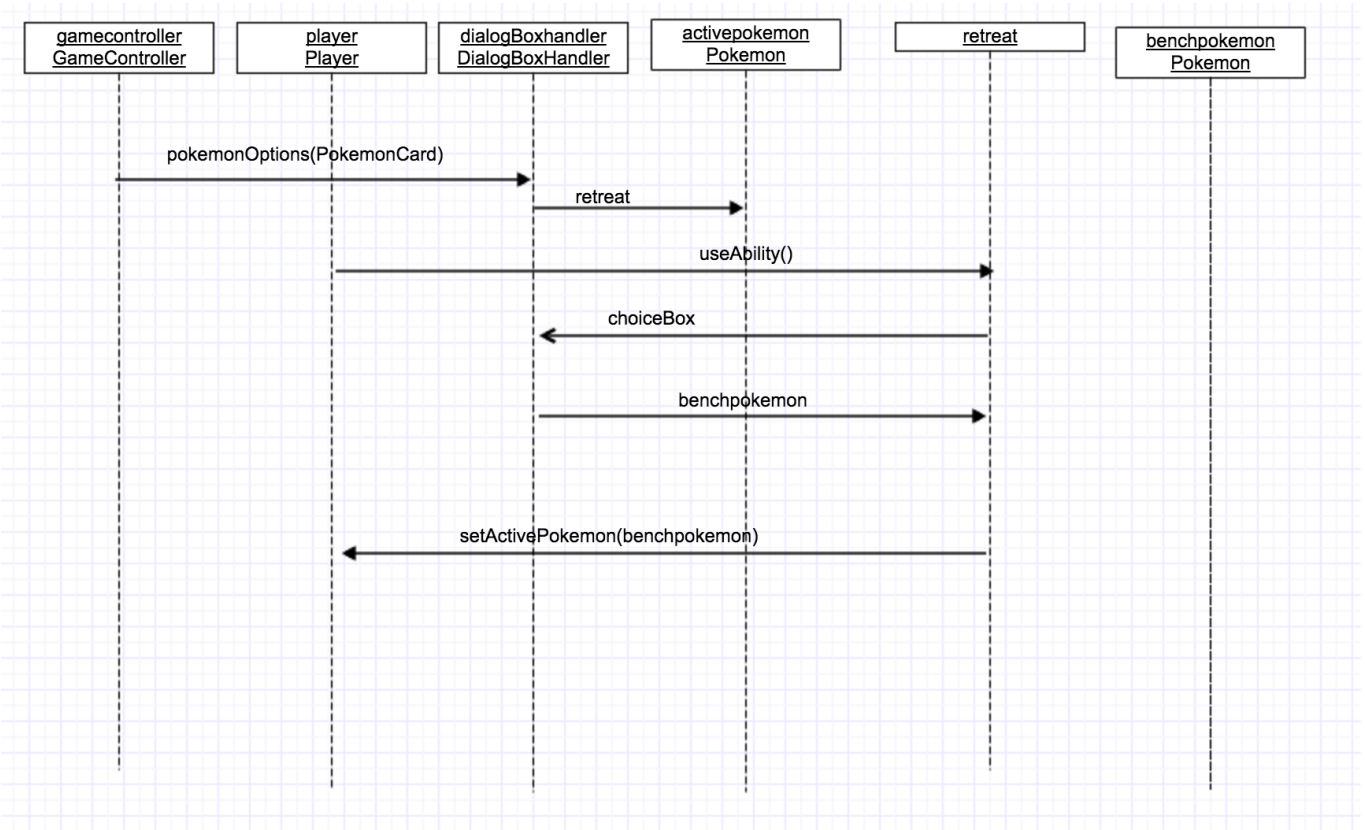


Fig: 2.6 Retreat Pokemon

2.2.2.5 Attack (represents UC8 ,For more detail refer to SRS document)

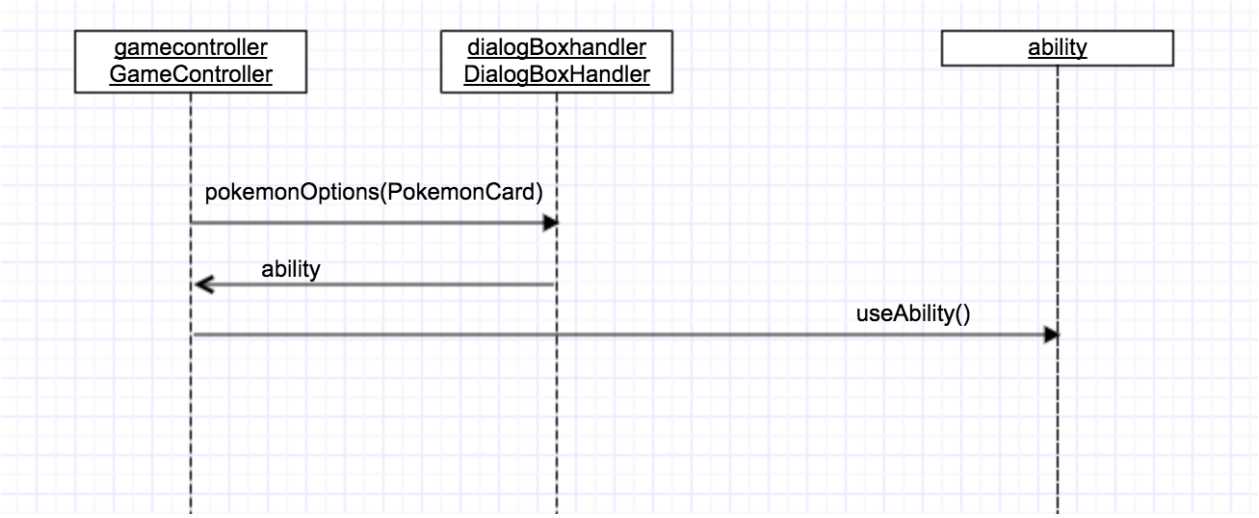


Fig: 2.7 Attack

2.2.3 Activity Diagram

For high-resolution view of an activity diagram please check Activity Diagram image in **Documents/Iages/**folder.

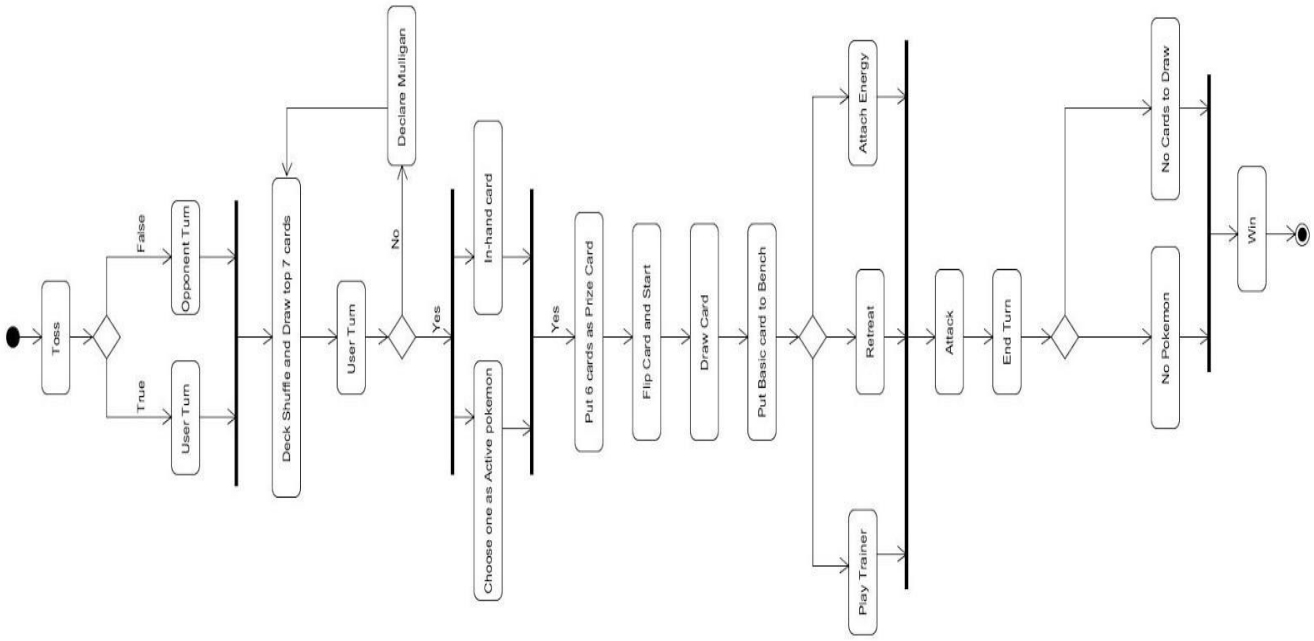


Fig:2.8 Activity Diagram

3 Software Interface Design

3.1System Interface Diagram

In Pokémon Go Back system level interface is only user interface as game does not use any Software and hardware interfaces. Here user interface is GUI, which allows user to interact with game. With use of GUI the players are able to see different states of the game.

3.1.1User Interface

In Pokémon Go Back user interface is a intermediate between the users and computer. For ease of game use the efficient user interface is necessary. Following these elements should be taken into account:

- Complete information: Interface should provide all information to user which is required to interact with user.
- Error avoidance: Interface should be able to avoid ambiguity. It should provide clear error messages to user.
- Usability: Interface should be easy

3.1.2Interface Design Rules

In interface we follow JakobNielsen' s heuristics principles likewise design, usability principles.

- Easy to learn and use: easily understandable and learnable throughout the complete duration of use
- Engaging: User feels totally involved using interface.
- Effective: The task is completed or goals reached completely and accurately
- Efficient: Helps to complete the task within time quickly
- Error Tolerant: Prevents errors and helps the user recover from mistakes and gives instructions to user for occurring errors.

3.1.3GUI Components

Graphical User Interface is built using javafx scene builder. There is description of main components which are used to develop GUI.These are as:

- SrollPane
- FlowPane
- Button
- CheckBox
- Label
- Pane
- AnchorPane

4 Detailed Description

4.1 Detailed Description

Class Name	Game Controller			
Inherited From	None			
Description	Biggest container of the game.			
Attributes	Visibility	Data Type	Name	Description
	private	UserPlayer	user	UserPlayer object
	private	AIplayer	Ai	AIplayer object
	private	ScrollPane	userScrollPane	Scroll pane of UserPlayer
	private	HBox	userBench	Bench of UserPlayer
	private	HBox	userHand	hand of UserPlayer
	private	ScrollPane	aiScrollPane	Scroll pane of AIplayer
	private	HBox	AIBench	Bench of AIplayer
	private	HBox	AIHand	Hand of AIplayer
	private	HBox	aiActivePokemon	Active Pokemon of AIplayer
	private	HBox	userActivePokemon	Active Pokemon of UserPlayer
	private	Button	UserEndTurnBtn	Button used to end the user turn.
	private	Label	userDamage	Displays damage of UserPlayer
	private	Label	aiDamage	Displays damage of AIplayer
	private	Pane	gameStage	Consists active cards of both the AIplayer and UserPlayer.
	private	BorderPane	gameBoard	consists active cards, inhand cards and bench cards of AIplayer and UserPlayer.
	private	VBox	btndn_rew	Vertical box consists done button and reward cards.

	private	VBox	aiDisc_deck	Vertical box consists deck and discard pile of AIplayer
	private	VBox	AIReward	Shows reward cards of AIplayer
	private	VBox	UIDisc_deck	Vertical box consists deck and discard pile of UserPlayer
Methods	Visibility	Return type	Name	Description
	private	NA	GameController()	Constructor.
	public	GameController	getInstance()	Returns object of GameController class.
	public	void	init()	Creates new players: AIplayer and UserPlayer. Then, sets turns for both players and deals multiple cards in hands of both players.
	public	void	addCardsToPanel(cardItem[] card,HBox panel)	Adds one or more cards in the User panel.
	public	void	removeCard(String id ,HBox panel)	Removes card from the panel
	public	void	addCardToPanel(cardItem card, HBox panel)	Adds a card in the panel.
	private	PokemonCard	createPokemonCard(Pokemonpokemon, HBox panel)	Creates card of pokemon type in the panel.
	private	Void	pokemonOptions(PokemonCardpokemonCard)	Gives available options to the desired pokemon.

	Private	GeneralCard	createCard(cardItem card, HBox panel)	Creates a cards in the panel.
	Private	Void	EnergyOptions(Gene ralCardnewcard)	Gives available Energy options to the desired card.
	Private	Void	trainerOptions(GeneralCardnewcard)	Gives available trainer options to the desired card.

public	HBox	getBench(Player player)	Returns bench object of UserPlayer or AIplayer.
public	HBox	getactivepokemon(Pl ayer player)	Returns active pokemon of UserPlayer or AIplayer.
public	HBox	getHand(Player player)	Returns bench object of UserPlayer or AIplayer.
Public	void	refreshCards(Player player)	Refreshes the active, inhand and bench cards
public	void	addCardsToAIPanel(cardItem[] cards, HBox panel)	Add cards in the panel of AIplayer.
Private	FlowPane	CreatePokemonCard (Pokemonpokemon)	Create the visible pokemon card, which consists name, Id, stage and hitpoints.
Public	void	dealCard(String player)	Give new card from deck to UserPlayer or AIplayer.
private	PokemonCard	evolveoptions(PokemonCard card)	Gives available evolve options to the pokemon card.
Public	Void	knockout()	Removes the active pokemon when the damage will be more than or equal to the hitpoints of pokemon.
public	void	winOrLoss()	Ask the player after winning or losing the game, if player want to play again or quit game.
public	void	makeUIResponsive()	Manage the height and width of all the display boxes.
public	Pokemon	getHandandBenchPokemonsDialog(Pla yer player)	Returns pokemon from the bench or hand of the UserPlayer or AIplayer.

	public	Pokemon	getPanelPokemonDialog(Player player,StringpanelOpt)	Returns pokemon from the panel of the UserPlayer or AIplayer.
	public	void	ulabelUpdate()	Updates the labels after changes
	public	boolean	getAbilityChoice()	Asks the player if player want to play the ability or not.

Class Name	AIplayer			
Inherited From	Player			
Description	Controls the functionality related to AIplayer			
Attributes	Visibility	Data Type	Name	Description
	private	string	name	name of the AIplayer.
	private	int	score	Current score of the AIplayer.
	private	cardItem	deck	Deck of AIplayer.
Methods	Visibility	Return type	Name	Description
	public	NA	AIplayer(String newName)	Constructor.
	public	String	getName()	Returns the name of the AIplayer.
	public	int	getScore()	Returns the current score of the AIplayer
	public	void	setTurn(booleannewTurn)	Sets the turn of the AIplayer
	private	void	runAI()	Run the AIplayer

	private	boolean	checkAndPlayEnergy(ArrayList<Energy>energyCards)	Check and add energy to the pokemon if conditions are true.
	public	void	activePokemonMove()	Makes pokemon active, choose card from bench or hand of AIplayer.
	public	void	updateGUI()	Updates the GUI after every turn.

Class Name	BasicPokemon			
Inherited From	PokemonStage			
Description	Set the stage of pokemon to Basic.			
Attributes	Visibility	Data Type	Name	Description
	private	string	stage	Pokemon stage.
Methods	Visibility	Return type	Name	Description
	public	NA	basicPokemon()	Constructor.
	public	String	getStage()	Get the pokemon stage.
	public	void	evolve(PokemonbasicCard)	Basic pokemon can not evolve with another Basic Pokemon.

Class Name	CardsGroup
Inherited From	CardItem
Description	Groups the cards to improve the accessibility.

Attributes	Visibility	Data Type	Name	Description
	default	int	uDeck	Deck of UserPlayer.
	default	int	aiDeck	Deck of AIPlayer
	private	ArrayList<cardItem>	groupCards	ArrayList of group of cards of any type.
Methods	Visibility	Return type	Name	Description
	public	void	addCard(cardItemnewCard)	Add single card to GroupCards
	public	void	addCards(cardItem[] newCards)	Add multiple card to GroupCards
	public	cardItem	getCard(int id)	Returns a card from the GroupCards based on given Id.
	public	cardItem	removeFirstCard()	Remove first card from the GroupCards.
	public	cardItem[]	getCard()	Returns card from the GroupCards.
	public	void	removeCard(cardItemnewCard)	Removes the given card from the GroupCards
	public	Pokemon	getBasicPokemonCard()	Returns a Basic Pokemon card from GroupCards.
	public	ArrayList<Pokemon	getAllBasicPokemonCar	Returns all the Basic
		>	d()	Pokemon card from GroupCards.
	public	ArrayList<Pokemon>	getStageOneCards()	Returns all the Stage one Pokemon card from GroupCards.
	public	ArrayList<Pokemon>	getAllPokemonCard(Stri ng type)	Returns all the Pokemon card from GroupCards.
	public	ArrayList<Energy>	getAllEnergyCards()	Returns all the Energy card from GroupCards.
	Public	ArrayList<Trainer>	getAllTrainerCards()	Returns all the Trainer card from GroupCards.
	public	String	getName()	Returns name of the card.
	public	int	getId()	Returns Id of the card.

	public	Arraylist<cardItem>	getGroupCards()	Returns the GroupCards
	public	void	setGroupCards(ArrayLis t <cardItem>groupCards)	Give the GroupCards.

Class Name	Deck			
Inherited From	CardsGroup			
Description	Manages the functionality related to Deck.			
Attributes	Visibility	Data Type	Name	Description
	private	string	name	Name of the Deck.
	private	int	deckNumber	Number of the Deck.
	private	DeckFile Reader	Db	Object of the DeckFileReader class.
Methods	Visibility	Return type	Name	Description
	public	N/A	Deck(int n)	Constructor.
	public	void	buildDeck(intdeckNumber)	Make a Deck of 60 cards using Either deck1 or deck2 values.
	public	void	display()	Shows the cards in the Deck.
	public	void	readFile()	Reads cards from the file for given deck number.
	public	void	shuffleCards()	Change the order of the cards in the Deck.
	public	String	getName()	Returns the name of the Deck.

Class Name	Energy			
Inherited From	cardItem			
Description	manages the tasks related to Energy Cards.			
Attributes	Visibility	Data Type	Name	Description
	protected	string	energy	Name of the Energy card.
	protected	int	id	Id of the Energy Card.

Methods	Visibility	Return type	Name	Description
	public	NA	Energy(String newEnergy, intnewID)	Constructor
	public	string	getName()	Returns the name of the Energy Card.
	public	int	getID()	Returns the id of the Energy Card.

Class Name	Player			
Inherited From	NA			
Description	Consist most of the objects used by the P layer.			
Attributes	Visibility	Data Type	Name	Description
	protected	string	name	Name of the Player. Player can be UserPlayer or AIplayer.
	protected	int	score	Score of the Player.Player can be UserPlayer or AIplayer.
	protected	cardItem	deck	Deck cards of the Player.Player can be UserPlayer or AIplayer.
	protected	cardItem	inhand	Inhand cards of the Player.Player can be UserPlayer or AIplayer
	protected	Pokemon	activePokemon	Active Pokemon of the Player.Player can be

				UserPlayer or AIplayer
	protected	boolean	turn	Turn of the Player.Player can be UserPlayer or AIplayer
	protected	CardsGroup	bench	Bench cards of the Player.Player can be UserPlayer or AIplayer.

	protected	CardsGroup	userDiscardPile	Discard pile of the UserPlayer.
	protected	CardsGroup	rewardCards	Reward cards of the Player.
Methods	Visibility	Return type	Name	Description
	public	NA	Player(String name)	Constructor.
	public	cardItem	getDeckCard()	Returns the first card from the deck.
	public	cardItem[]	getDeckCards(inti)	Returns the given cards from the deck.
	public	void	addRewardCrads(inti)	Give Reward cards as given by the player
	private	cardItem	dealCard()	Transfer a card from the Deck to the inhand of the player.
	private	cardItem[]	getInhandCards()	Returns inhand cards of the player.
	public	cardItem	getInhand()	Returns a single inhand card.
	public	void	setactivePokemon(Pok e mon newPokemon)	Sets a pokemon card as active pokemon card.
	public	Pokemon	getActivepokemon()	Returns the active pokemon card.
	public	ability[]	getActiveAbilities()	Returns the Active abilities of the pokemon.
	public	Deck	getDeck()	Returns the control of the Deck.
	public	boolean	getTurn()	Returns the turn of the Player.
	public	cardItem	getPokemonFromHand( )	Selects the pokemon card from the hand of the Player.
	public	void	evolve(PokemonstageO nePokemon, Pokemon basic Pokemon)	Replace the active basic pokemon card with the available stage one pokemon card.
	public	cardsGroup	getBench()	Returns the control of the bench of the Player.
	public	cardsGroup	getDiscardPile()	Returns the control of the Discard Pile of the Player
	public	ArrayList<Po kemon>	getPokemonFromBenc hAndActive(	Returns all the Pokemon cards from bench and active
				area.

Class Name	Pokemon			
Inherited From	cardItem			
Description	Contains important methods and objects used by player. All the functionality of Pokemon card is defined.			
Attributes	Visibility	Data Type	Name	Description
	private	int	id	Id of the pokemon card
	private	String	cardName	Name of the pokemon card
	private	int	hitpoints	Hitpoints is the value upto which a pokemon card can take damage.
	private	pokemonStage	pStage	Stage of the pokemon card.
	private	int	Damage	Initial damage for the pokemon card is 0.
	private	String	state	State of the pokemon card.
	private	ArrayList<ability>	abilities	Abilities attached to the pokemon card.
	private	String	status	Status of the pokemon card.
	private	ArrayList<cardItem>	attachedCards	Cards attached to the pokemon.
	private	ArrayList<ability>	activeAbilities;	Abilities which are active for the pokemon card.
	private	PokemonCard	uiCard	Pokemon card of the UserPlayer
	private	boolean	healed	Default value =False

	private	Retreat	retreat	Active pokemon can be replaced by any bench pokemon card.
Methods	Visibility	Return type	Name	Description
	public	NA	Pokemon(intnewId, String name, pokemonStagenewPokemonStage, int	Constructor.

			newHp, ArrayList<ability>newAbilities,RetreatnewRetreat)	
	public	void	addDamage(intnewDamage)	Adds damage to the Pokemon card.
	public	void	removeDamage(intnewHealing)	Removes damage from the Pokemon card.
	public	boolean	isHealed()	Checks whether pokemon card is healed or not.
	public	void	resetHealStatus()	Resets the Healed status to False.
	public	void	setState(String newState)	Gives new state to the pokemon card.
	public	String	getName()	Returns the name of the pokemon card.
	public	Int	getID	Returns the Id of the pokemon card.
	public	String	getStage()	Returns the stage of the pokemon card.

public	String	getState()	Returns the state of the pokemon card.
public	Int	getDamage()	returns the damage of the pokemon card.
public	ability[]	getAbilities()	returns the abilties attached to the pokemon card.
public	string	getStatus()	Returns the status of the pokemon card.
public	void	setStatus()	Sets status of the pokemon card.
public	void	attachCard(cardItemnewCard)	Attach a card to the pokemon card.
public	void	dettachCardType(Class< ? Extends cardItem>newtype, inti)	Remove a card from the pokemon card.
public	void	useAbility(ability uAbility)	Player uses the ability and turn is given to the opponent.
public	Int	getAttachCardsCount()	Returns the number of cards attached to the pokemon card.
public	cardItem[]	getAttachedcards()	Returns array of the cards attached to the
			pokemon card.
public	void	evolve(PokemonbasicCard)	Replace the active basic pokemon card with the available stage one pokemon card.

public	Int	getHP()	Returns the hitpoints of the pokemon card.
public	String	getBasePokemonName()	Returns the name of the Basic pokemon card.
public	void	attachCard(cardItem[] cards)	Adds the given number of cards to the pokemon card.
public	void	addObserver(PokemonCardnewCard)	Adds an Observer to the pokemon card.
public	void	removeAbility(ability newAbility)	removes a n ability from the pokemon card.
public	void	addActiveAbility(ability newAbility)	Adds an ability to the pokemon and set is active.
public	ArrayList<ability>	getActiveAbilities()	Returns active abilities of the pokemon card.
public	string	getAbilityIndex(ability newAbility	Returns the value of the index at which the abilities is added in the arraylist.
public	Int	getAttachedCardsCount(Class<?>classtype)	Returns the number of the cards attached to the pokemon card
public static	void	main(String[] arg)	Do the parsing and creates a pokemon card.

Class Name	pokemonStage			
Inherited From	NA			
Description	Abstract class consists two abstract methods			
Methods	Visibility	Return type	Name	Description
	public	String	getStage()	Give current stage of pokemon.
	public	void	evolve(PokemonbasicCard)	Active pokemon card is replaced with a bench pokemon card if conditions are true.

Class Name	stageOnePokemon			
Inherited From	Pokemonstage			
Description	Set the stage of pokemon to Stage one.			
Attributes	Visibility	Data Type	Name	Description
	private	string	stage	Stage of pokemon
	private	String	basicCardname	Name of basic pokemon.
	private	Pokemon	basicPokemon	Object of basic pokemon.
Methods	Visibility	Return type	Name	Description
	public	NA	stageOnePokemon(String newEvolvePokemon)	Constructor.
	public	String	getStage()	Gives current stage of pokemon.
	public	String	getBasicPokemonName()	Gives name of basic pokemon.
	public	void	evolve(PokemonbasicCard)	Active pokemon card is replaced with a bench pokemon card if conditions are true.
	public	boolean	isEvolved()	Check if stage one pokemon card is replaced with basic pokemon card.
	public	Pokemon	getBasicPokemonCard()	Gives the basic pokemon.

Class Name	Trainer			
Inherited From	cardItem			
Description	Manages the Trainer cards.			
Attributes	Visibility	Data Type	Name	Description
	private	string	cardName	Name of the trainer card.
	private	int	id	Id of the trainer card.
	private	ability	cardAbility	Ability of the trainer card.
Methods	Visibility	Return type	Name	Description
	public	NA	Trainer(String name, int value, ability newAbility)	Constructor.
	public	String	getName()	Gives name of the trainer card.
	public	int	getID()	Gives Id of the trainer card.
	public	ability	getAbility()	Gives ability of the trainer card.

Class Name	Turn			
Inherited From	NA			
Description	Decides the turn of the players.			
Attributes	Visibility	Data Type	Name	Description
	private	Turn	turn	Turn of the player.
	private	AIplayer	Ai	Object of AIplayer
	private	UserPlayer	user	Object of UserPlayer.
Methods	Visibility	Return type	Name	Description
	private	NA	Turn()	Constructor.
	public	Turn	getInstance()	Returns object of Turn class.
	public	void	setPlayer(AIplayernewAi, UserPlayernewuser)	Set the Players as newAI and newuser.
	public	Player	getCurrentPayer()	Give the current Player.

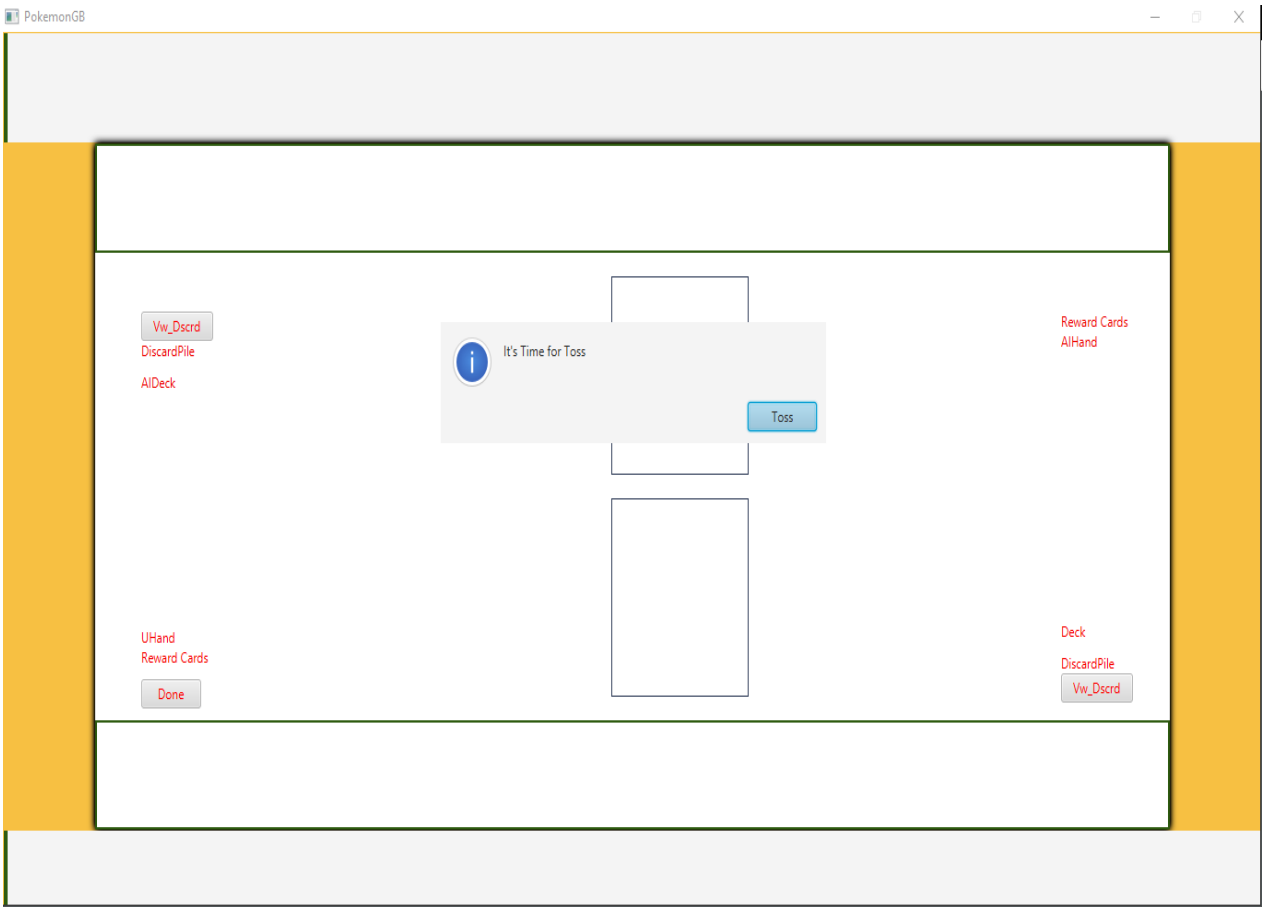
	public	void	changeTurn()	Change the turn.
	public	Player	getOpponent()	Returns the opponent player.
	public	boolean[]	toss()	UserPlayer and AIplayer wins the toss randomly and whoever wins , gets first turn.

Class Name	UserPlayer			
Inherited From	Player			
Description	Manages both player's functionality.			
Attributes	Visibility	Data Type	Name	Description
	protected	String	name	Name of the UserPlayer
	protected	Int	score	Score of the UserPlayer
	protected	cardItem	deck	Deck cards of the UserPlayer
	protected	cardItem	inhand	Inhand Cards of the UserPlayer
	protected	Pokemon	activePokemon	Active Pokemon of the UserPlayer
	protected	Boolean	turn	Turn of the UserPlayer
	protected	CardsGroup	bench	Bench of the UserPlayer
	protected	CardsGroup	userDiscardPile	Discard pile of the UserPlayer
	protected	CardsGroup	rewardCards	Reward cards of the UserPlayer
Methods	Visibility	Return type	Name	Description
	public	NA	UserPlayer(String newName)	Constructor.
	public	String	getName()	Gives name of the UserPlayer
	public	Int	getScore()	Gives score of the UserPlayer.
	public	void	setTurn(booleannewTurn)	Set the turn of UserPlayer.

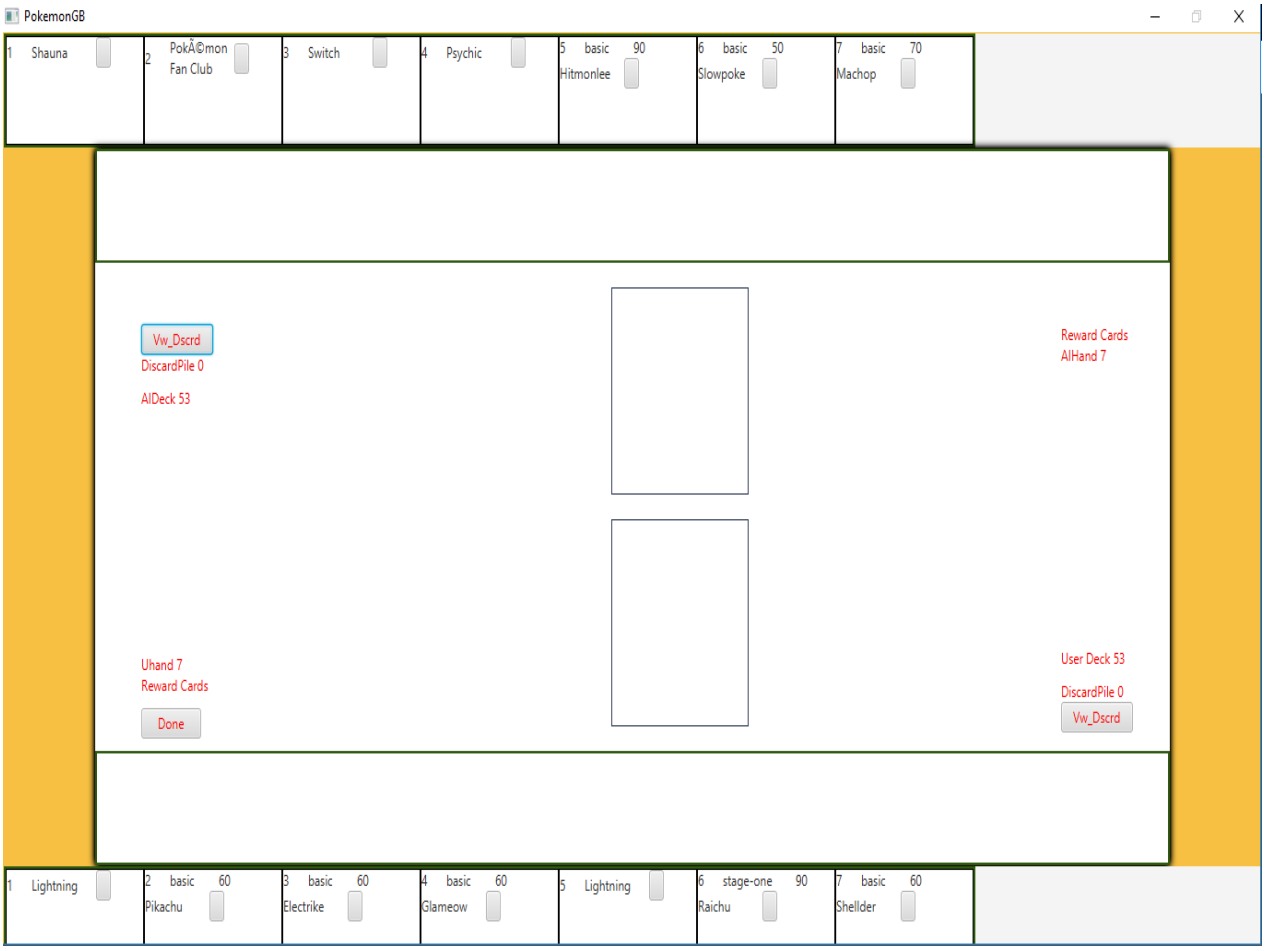
5 Screenshot

Game Start

At start of the game the system will ask for toss and select one of the coin head or tail.

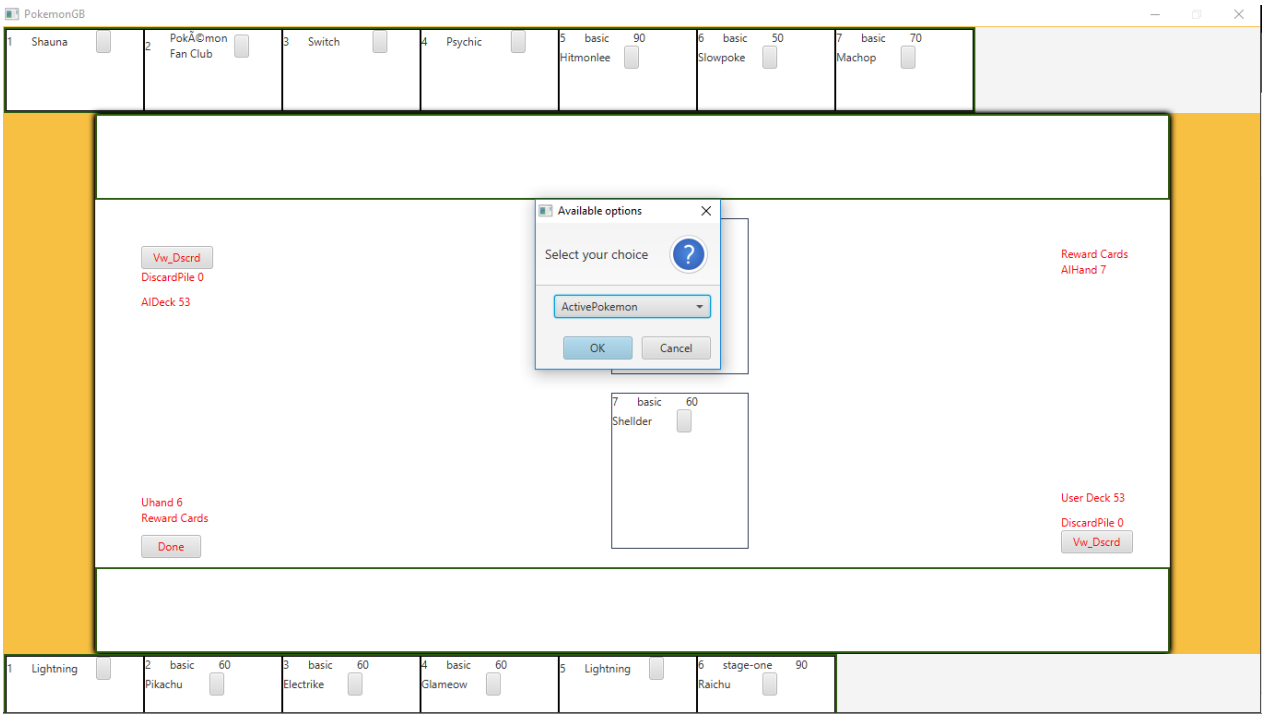


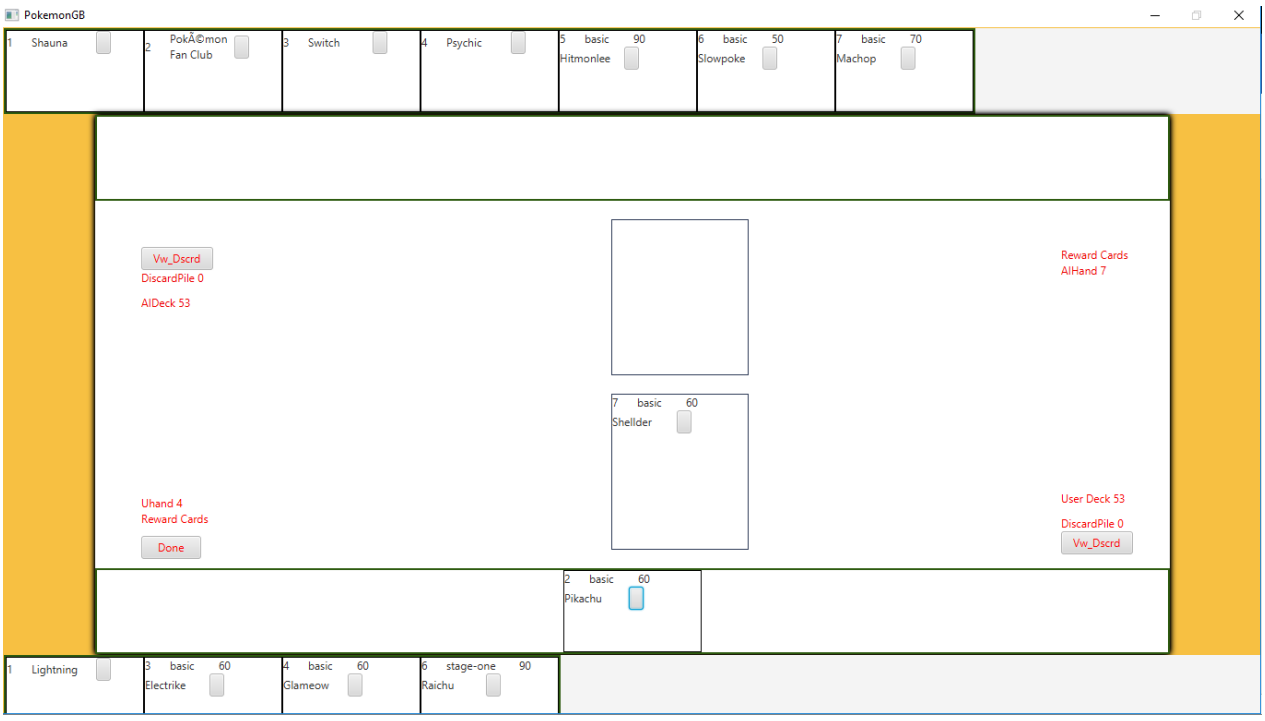
Main View:
System Design Specifications - Pokémon Go Back

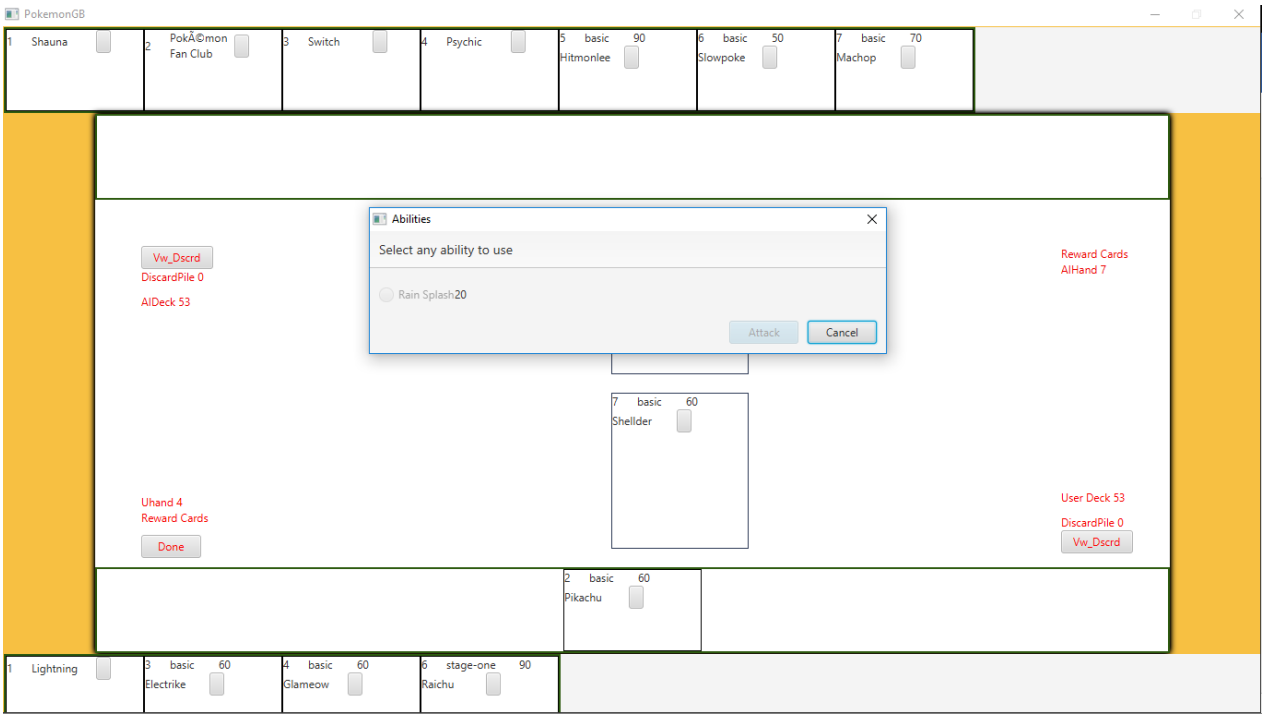


Other Views:

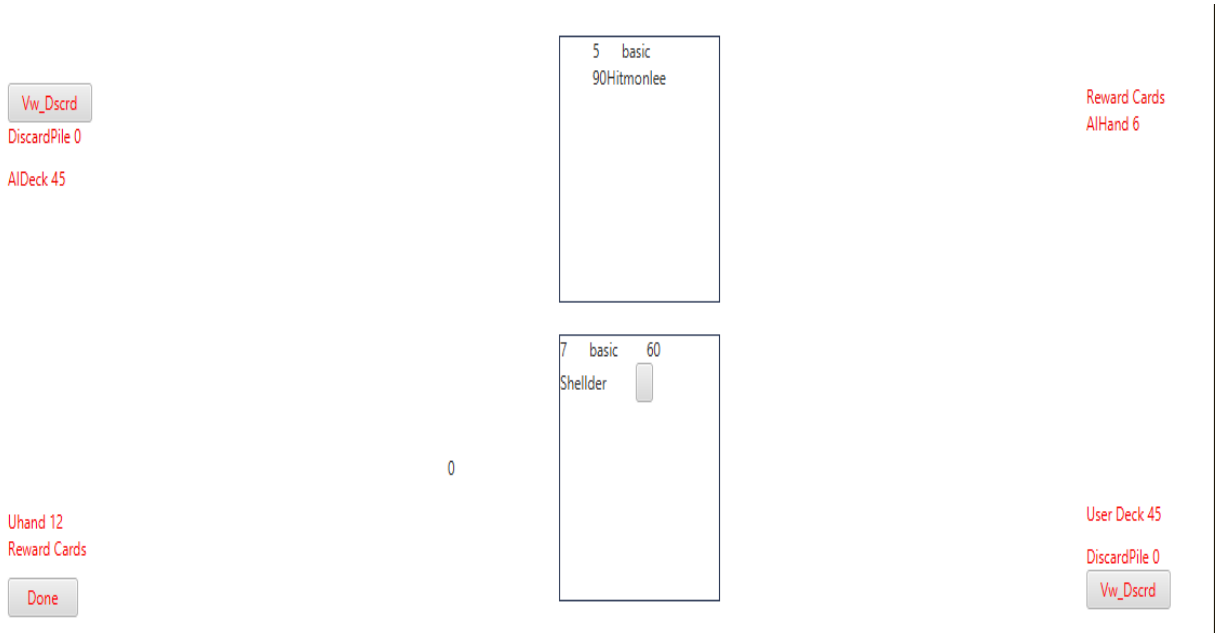
System Design Specifications - Pokémon Go Back

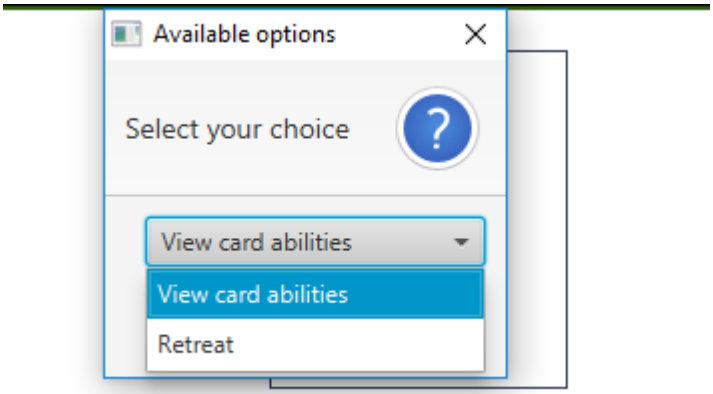

PokemonGB
1 Shauna
2 PokÃ©mon Fan Club
3 Switch
4 Psychic
5 basic 90 Hitmonlee
6 basic 50 Slowpoke
7 basic 70 Machop
Available options
Select your choice
ActivePokemon
OK
Cancel
Vw_Dscrd
DiscardPile 0
AIDeck 53
Reward Cards
AIHand 7
7 basic 60 Shellder
Uhand 6
Reward Cards
Done
User Deck 53
DiscardPile 0
Vw_Dscrd
1 Lightning
2 basic 60 Pikachu
3 basic 60 Electrike
4 basic 60 Glameow
5 Lightning
6 stage-one 90 Raichu

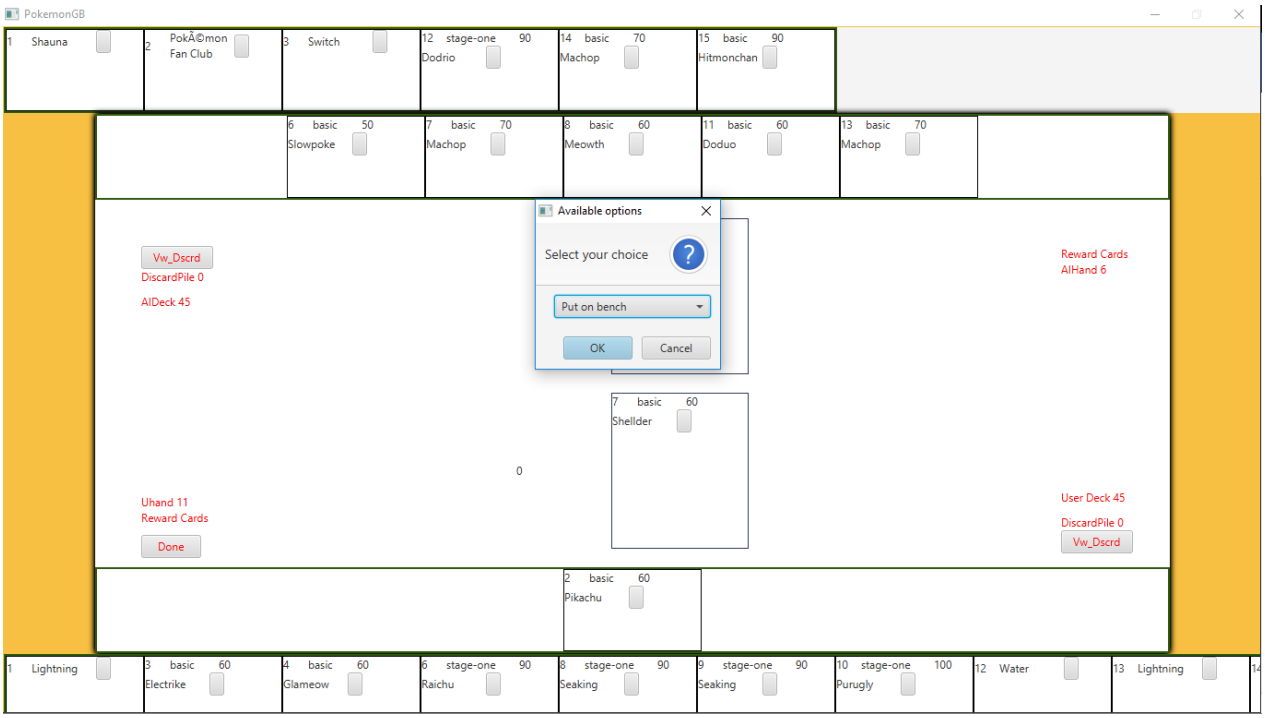

PokemonGB
1 Shauna
2 PokÃ©mon Fan Club
3 Switch
4 Psychic
5 basic 90 Hitmonlee
6 basic 50 Slowpoke
7 basic 70 Machop
Vw_Dscrd
DiscardPile 0
AIDeck 53
Reward Cards
AIHand 7
7 basic 60 Shellder
Uhand 4
Reward Cards
Done
User Deck 53
DiscardPile 0
Vw_Dscrd
2 basic 60 Pikachu
1 Lightning
3 basic 60 Electrike
4 basic 60 Glameow
6 stage-one 90 Raichu


PokemonGB
1 Shauna
2 PokÃ©mon Fan Club
3 Switch
4 Psychic
5 basic 90 Hitmonlee
6 basic 50 Slowpoke
7 basic 70 Machop
Abilities
Select any ability to use
Rain Splash20
Attack
Cancel
Vw_Dscrd
DiscardPile 0
AIDeck 53
Reward Cards
AIHand 7
7 basic 60 Shellder
Uhand 4
Reward Cards
Done
User Deck 53
DiscardPile 0
Vw_Dscrd
2 basic 60 Pikachu
1 Lightning
3 basic 60 Electrike
4 basic 60 Glameow
6 stage-one 90 Raichu


PokemonGB
1 Shauna
2 PokÃ©mon Fan Club
3 Switch
12 stage-one 90 Dodrio
6 basic 50 Slowpoke
7 basic 70 Machop
8 basic 60 Meowth
11 basic 60 Doduo
5 basic 90Hitmonlee
Vw_Dscrd
DiscardPile 0
AIDeck 48
Reward Cards
AIHand 4
7 basic 60 Shellder
0
Uhand 9
Reward Cards
Done
User Deck 48
DiscardPile 0
Vw_Dscrd
2 basic 60 Pikachu
1 Lightning
3 basic 60 Electrike
4 basic 60 Glameow
6 stage-one 90 Raichu
8 stage-one 90 Seaking
9 stage-one 90 Seaking
10 stage-one 100 Purugly
11 Lightning
12 Water


Vw_Dscrd
DiscardPile 0
AIDeck 45
5 basic
90Hitmonlee
Reward Cards
AIHand 6
7 basic 60
Shellder
0
Uhand 12
Reward Cards
Done
User Deck 45
DiscardPile 0
Vw_Dscrd


Available options
Select your choice
View card abilities
View card abilities
Retreat


PokemonGB
1 Shauna
2 PokÃ©mon Fan Club
3 Switch
12 stage-one 90 Dodrio
14 basic 70 Machop
15 basic 90 Hitmonchan
6 basic 50 Slowpoke
7 basic 70 Machop
8 basic 60 Meowth
11 basic 60 Doduo
13 basic 70 Machop
Available options
Select your choice
Put on bench
OK
Cancel
Vw_Dscrd
DiscardPile 0
AIDeck 45
Reward Cards
AIHand 6
7 basic 60 Shellder
0
Uhand 11
Reward Cards
Done
User Deck 45
DiscardPile 0
Vw_Dscrd
2 basic 60 Pikachu
1 Lightning
3 basic 60 Electrike
4 basic 60 Glameow
6 stage-one 90 Raichu
8 stage-one 90 Seaking
9 stage-one 90 Seaking
10 stage-one 100 Purugly
12 Water
13 Lightning

6 Artificial Intelligence (AI)

This section will deal with the artificial intelligence component of the game. This is a feature of the system that needs to be dealt with separately, we chose to describe it here. The AI is required in the cases where a decision for a system player needs to be made. All the actions of opponent System Design Specifications - Pokémon Go Back

Player will be equal to actual player. AI will perform all the tasks as user player except game setup. Game setup can be done by user player only.

7 Roles and Contributions

Document	Section	Contribution
Software requirement specification		
	Introduction and Overall Description	Wrote by Simranjitsinghbachher and reviewed by Mandeep Singh
	Budget	Wrote by Mandeep Singh and reviewed By Simranjit Singh Bachher
	Domain Model	Designed by Gaganjot Kaur and reviewed by GurpreetkaurLotey
	Functional Requirements	Wrote by Mandeepsingh and reviewed by Gurpreet Raju and Navjot Singh
	Use Cases	Use Case Diagram – Designed by Gaganjot Kaur and reviewed by Gurpreet Kaur Lotey Simranjitsinghbachher and Mandeep Singh Wrote **use cases (1-4)** reviewed by Gaganjot Kaur and GurpreetkaurLotey Use case (5-4) written by Gurpreet Kaur Lotey and Reviewed by Gaganjot Kaur Gurpreet Raju , Amit Bhardwj and Gunpreet Ahuja reviewed all Use Cases to ensure consistency between Code, Acceptance tests and Use Cases.
	Conclusion	Witten by Simranjit Singh Bachher reviewed by Gagnjot Kaur

Supplementary Specification Document		
	Introduction	Written by Harinder Kaur and reviewed by Navjot Singh
	Functionality	Written by Harinder Kaur and reviewed by Gurpreet Raju and Navjot Singh for GUI designing and code.
	Usability	Written by Harinderkaur and reviewed by Navjot Singh
	Reliability	
	Performance	
	Design Constraints	
	Purchased Components	
	Interfaces	
	Licensing Requirements	
	Legal, Copyright and other notices	
	Applicable Standards	
Vision Document		Written by Navrang Pal Kaur Dhaliwal and reviewed by Navjot Singh
Software Design Specification		
	Introduction	Written by Navrang Pal Kaur Dhaliwal and reviewed by Mandeep Singh
	Architectural Design	Written by Navrang Pal Kaur Dhaliwal reviewed by Gunpreet Ahuja
	Class Model	Written by Navjotsingh and reviewed by Gurpreet Raju and Amit Bhardwj to ensure consistency in Code
	Sequence diagram	Designed by Gaganjotkaur and Mandeep Singh and reviewed by Gunpreet Ahuja for Coding.

	Activity Diagram	Designed by Harinderkaur and Simranjit Singh and reviewed by Gurpreet Kaur Lotey
	Software Interface Design	Written by Mandeep Singh and reviewed by Navjot Singh to ensure consistency between user requirements and GUI
	Detailed Description	Written by Gunpreet Ahuja and reviewd by AmittBhardwj and Gurpreetraju for coding
Test Document		Witten by Harinder Kaur and Navrang Pal Kaur Dhaliwal and reviewed by Gurpreet Kaur Lotey and Gaganjot Kaur (Testing is also done by these four team members)

8 Conclusion

The Pokémon Go Back game is standalone system. In designing part we followed Rational Unified Process model. Design Document consists of class diagram, sequence diagram and detailed description of classes, all these aspects describe design of game e effectively.