Team:
Dylan Ahearn
Leah Rodgers,
Nick Smith

Title: Notches

Project Summary: A turn based game, in which the user sends out an avatar to "adventure" The avatar returns with loot that can be sold or equipped. While on "adventure the player chooses each day to continue another day or return home. The more days the adventure lasts the higher the chance of good loot and the chance of death. The player avatar has levels called notches. Actors: Player (Video game players, poor college students, graders, our friends who we make try it, six year olds, etc.)

Project Requirements:

There are no business requirements for our project unless we implement the stretch goal of in app purchases.

User Requirements	
ID	requirement
UR-001	Start new game and create new character or continue last save
UR-002	Click adventure button resulting in one day of adventure
UR-003	Character can die while adventuring and amass loot (Higher chance of both as days progress)
UR-004	User can choose to continue adventure each day or return home
UR-005	View loot upon return from adventure
UR-006	See character stats, inventory, and equipped gear and equip gear
UR-007	Buy and sell items and see shop
UR-008	Can save game and exit while home

Non-Functional Requirements	
ID	Requirement
NFR-001	Uses a relational database
NFR-002	Current player progress stored between a database and file
NFR-003	User never notices loading or lag more than 10 seconds

Use Cases:
UC-01. As a player I want to be able to create a new game and start playing. (UR-001, UR-002)
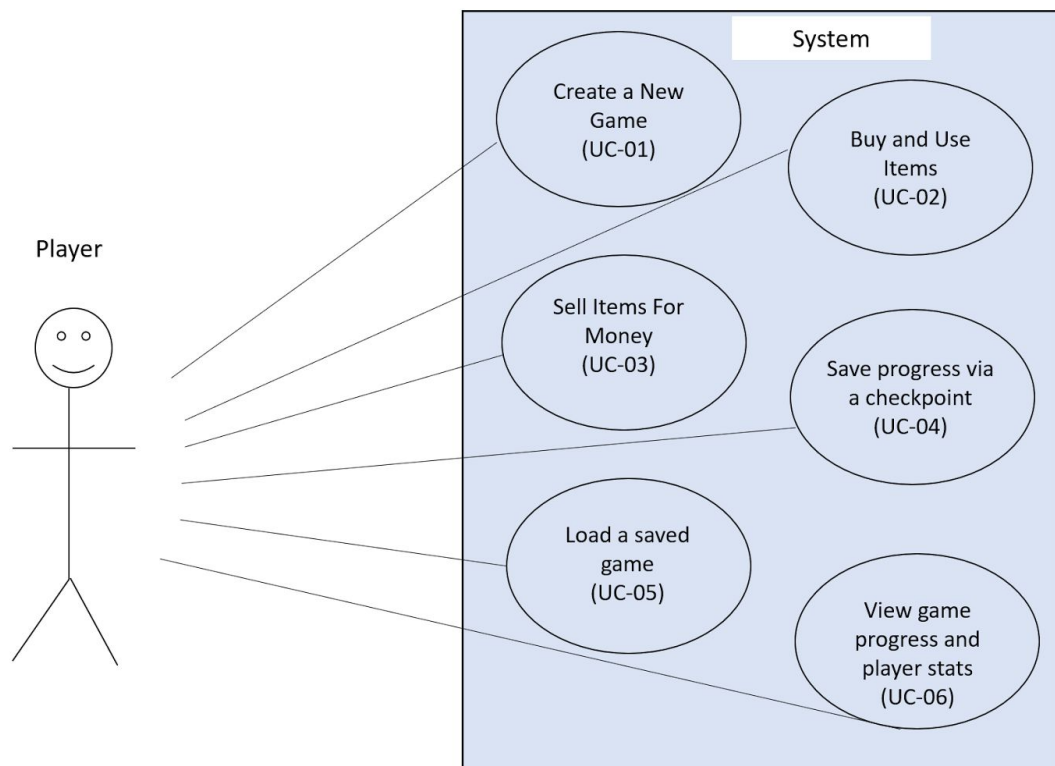UC-02. As a player I want to be able to buy and use items. (UR-007)
UC-03. As a player I want to be able to sell items for money. (UR-007)
UC-04. As a player I want to be able to save my progress with a checkpoint. (UR-004, UR-008)
UC-05. As a player I want to be able to load a saved game from previous play. (UR-001)
UC-06. As a player I want to be able to see my game progress and stats. (UR-005, UR-006)
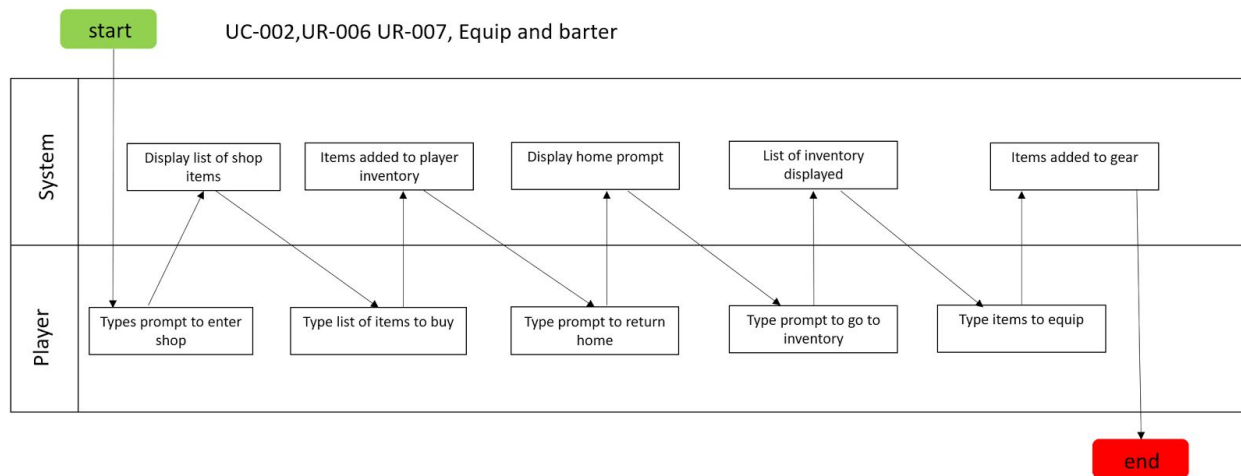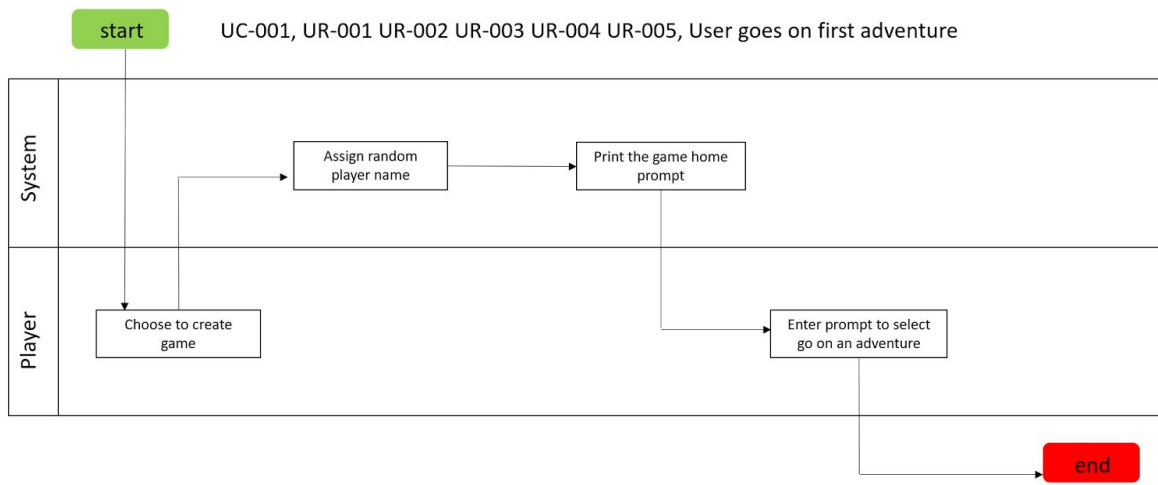
Use Case ID	UC-001
Use Case Name	User Goes on first adventure
Description	User needs to create a new game and get game started

Actors	player	
Pre-conditions	No data is saved, game is running, but no instance created yet	
Post-conditions	Player is playing the game and "going on an adventure"	
Frequency	However often a new game is created, probably on a regular basis for the player	
Flow of events	Actor Action:	System Response:
	1.Choose create game	displays game home screen
	2.Choose "go on an adventure"	displays result of days adventure
variations	none	

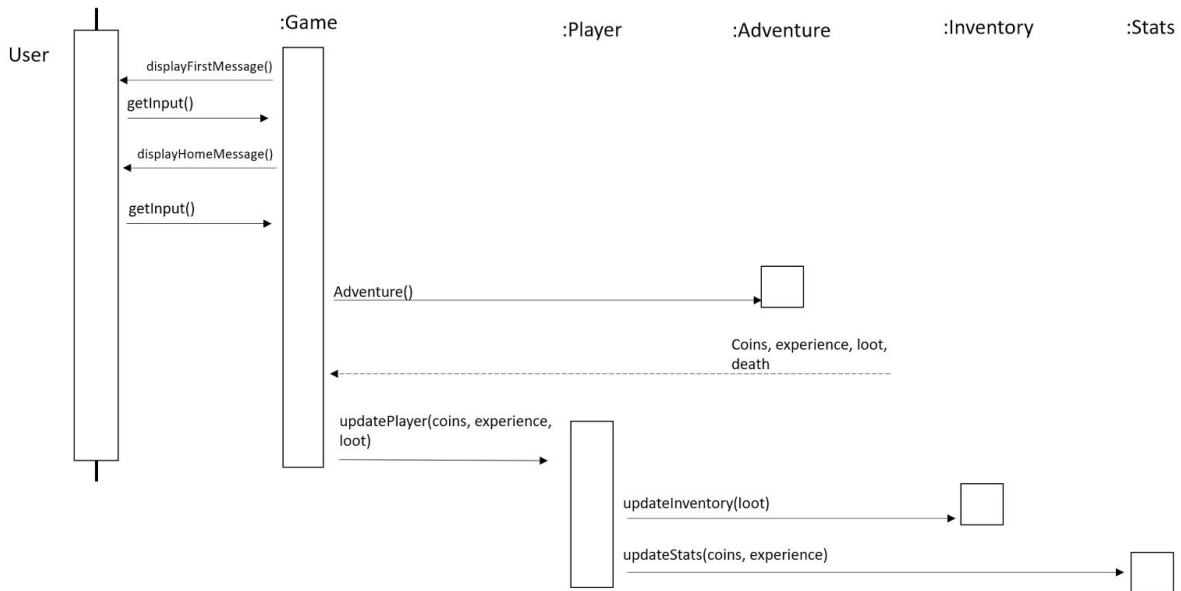| Use Case ID	UC-002
Use Case Name	Equip and Barter
Description	User participates in buying objects, and then equips one of the objects

Actors	player	
Pre-conditions	The player is "at home"	
Post-conditions	The player is still at home, but with new items in the inventory and new items in the gear, those items have been removed from the shop	
Frequency	Regularly during game play	
Flow of events	Actor Action:	System Response:
	1.Types prompt to go to shop	prints list of items in the shop
	2.Type items for purchase	items are added to inventory
	3.Type prompt to return home	shows home prompt
	4.Types prompt to go to inventory	prints list of items in shop
	5.Types item to be equipped	item is added to gear
	6.Typs prompt to return home	shows home prompt
variations	none	

Activity Diagrams:

UC-001, UR-001 UR-002 UR-003 UR-004 UR-005, User goes on first adventure

start

System		Assign random player name	→	Print the game home prompt	
Player	Choose to create game				Enter prompt to select go on an adventure

end

UC-002,UR-006 UR-007, Equip and barter

start

System	Display list of shop items	Items added to player inventory	Display home prompt	List of inventory displayed	Items added to gear
Player	Types prompt to enter shop	Type list of items to buy	Type prompt to return home	Type prompt to go to inventory	Type items to equip

end

Sequence Diagrams:

UC-001, UR-001 UR-002 UR-003 UR-004 UR-005, User goes on first adventure

User — :Game — :Player — :Adventure — :Inventory — :Stats

- displayFirstMessage()
- getInput()
- displayHomeMessage()
- getInput()
- Adventure()
- Coins, experience, loot, death
- updatePlayer(coins, experience, loot)
- updateInventory(loot)
- updateStats(coins, experience)

UC-002, UR-006 UR-007, Equip and barter

User — :Game — :Player — :Inventory — :Shop — :Stats — :Gear

- displayHomeMessage()
- getInput()
- goToShop()
- printItems(shopItems)
- shopItems
- getInput()
- removeItems(items)
- updatePlayer(items, price)
- displayShopMessage()
- updateInventory(items)
- getInput()
- updateStats(price)
- displayHomeMessage()
- exit()
- getInput()
- goToIventory()
- printItems(inventoryItems)
- inventoryItems
- getInput()
- updatePlayer(items)
- updateGear(items)

UI Mockups:

```
Welcome to notches, would you like to create or load?
Type "create" or "load"

Create

Your character is named Kenny. What would you like to do?
Type "shop", "inventory", "stats", or "ADVENTURE"

ADVENTURE

Day 1: You found a bag, and 2 coins. What would you like to
do?
Type "go home" or "ADVENTURE"
```

UC-01

```
You are at home. What would you like to do?
Type "shop", "inventory", "stats", or "ADVENTURE"

shop

STORE:
Hammer: 10 coins
Sword: 35 coins
Armor: 100 coins
_____
Type what you would like to buy separated by spaces or X to exit

Hammer Sword

You now own Hammer Sword
What would you like to do?
Type "go home", or "buy items" or "sell items"

go home

You are at home. What would you like to do?
Type "shop", "inventory", "stats", or "ADVENTURE"

inventory

INVENTORY:
Hammer
Sword
_____
Type what you would like to equip separated by spaces, or X to exit

Hammer

Hammer is equipped
```
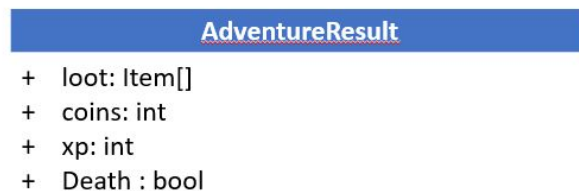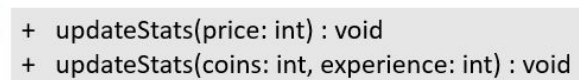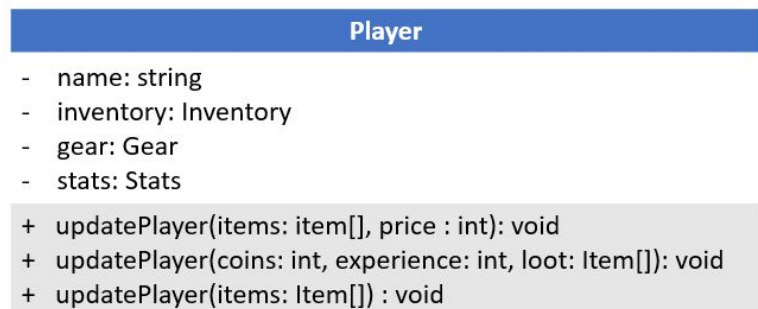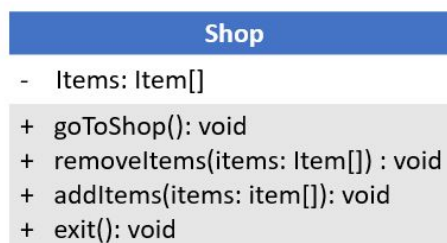
UC-02

Class diagrams:

Overview:

Game

- player : Player
- adventure : Adventure
- shop : Shop

- displayFirstmessage() : void
- getInput(): String
- displayHomeMessage(): void
- print Items(items :Item[]) : void
- displayShopMessage() : void
- displayInventoryMessage(): void

Shop

- Items: Item[]

+ goToShop(): void
+ removeItems(items: Item[]) : void
+ addItems(items: item[]): void
+ exit(): void

Player

- name: string
- inventory: Inventory
- gear: Gear
- stats: Stats

+ updatePlayer(items: item[], price : int): void
+ updatePlayer(coins: int, experience: int, loot: Item[]): void
+ updatePlayer(items: Item[]) : void

Stats

- notch : int
- coins: int
- xp: int

+ updateStats(price: int) : void
+ updateStats(coins: int, experience: int) : void

AdventureResult

+ loot: Item[]
+ coins: int
+ xp: int
+ Death : bool

Inventory

- items: Item[]
- maxItems: int

+ updateInventory(items: Item[]): void
+ goToInventory(): Item[]

Adventure

- day: int
- adventureResult: AdventureResult

+ adventure(): AdventureResult
- doAdventure(xp: int): AdventureResult

Item

+ name: String
+ worth: int
+ rarity: int
+ isEquipped: bool

Gear

- items: Item[]

+ updateGear(items: Item[]): void