Software Design Document

for DayGame

Group 7

Name	*ID*
Βασιλειάδης Ανέστης	dai19272
Καπουτσέλης Χρήστος	it14234
Μοσχόπουλος Αποστόλης	dai19104
Μπασιούκας Γεώργιος	dai19174
Ντατίδης Μητροφάνος	dai19011
Παπαδόπουλος Δημήτρης	dai17096
Παπαδοπούλου Αθανασία	dai19091
Σαράφογλου Μαρίνα	dai19080
Σπυριδόπουλος Κωνσταντίνος	dai19106
Τσιρπάνης Θοδωρής	dai19090

Instructor: Αμπατζόγλου Απόστολος, Χατζηγεωργίου Αλέξανδρος

Course: Software Engineering

Table of Contents

Revision

Name	Date	Reason	Version
Μπασιούκας Γιώργος	30/03/2020	Created Document	**0.1**
Μπασιούκας Γιώργος	30/03/2020	Added Table of Contents and Revision	**0.2**
Μπασιούκας Γιώργος Παπαδοπούλου Αθανασία Σπυριδόπουλος Κωνσταντίνος Τσιρπάνης Θοδωρής	03/04/2020	Added Introduction	**0.3**
Μπασιούκας Γιώργος	22/04/2020	Added Interface Explanations Added User Interfaces	**0.4**
Καπουτσέλης Χρήστος Μπασιούκας Γιώργος Παπαδοπούλου Αθανασία Σπυριδόπουλος Κωνσταντίνος Τσιρπάνης Θοδωρής	29/04/2020	Added Class Diagram Added Boss Battle Sequence Diagram Added Shop Sequence Diagram Added Update Task Sequence Diagram	**0.5**
Μπασιούκας Γιώργος Παπαδοπούλου Αθανασία	29/04/2020	Added System Static View	**0.6**
Σπυριδόπουλος Κωνσταντίνος Τσιρπάνης Θοδωρής	01/05/2020	Added Databases	**0.7**
Καπουτσέλης Χρήστος	01/05/2020	Added Dynamic System View Added Footprint Table	**0.8**
Καπουτσέλης Χρήστος Μπασιούκας Γιώργος Παπαδοπούλου Αθανασία Σπυριδόπουλος Κωνσταντίνος Σαράφογλου Μαρίνα Τσιρπάνης Θοδωρής	01/05/2020	Review Document	**0.9**
Μπασιούκας Γιώργος	02/05/2020	Document Fix	**0.9.1**
	02/05/2020	**Release**	**1.0**

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed description of the design of the desktop application DayGame. It will explain the detailed structure of the software, the classes of the software, what the software will do and the conditions under which it must operate. This document is intended for users of the software and also potential developers.

1.2 Document Conventions

This Document was created based on the Software Requirements Specification of DayGame as well as on the IEEE standard 1016 for Software Design Documents.

1.3 Intended Audience

This document is intended to be used by members of the project team that will implement and verify the correct functioning of the system.

1.4 Product Scope

DayGame is a desktop application that focuses on managing tasks in an interactive way using elements from role-playing games, while tracking everyday tasks and motivating users to complete them in exchange for virtual rewards.

1.5 Definitions, Acronyms and Abbreviations

This section remains the same with the previously submitted SRS.

1.6 References

DayGame SRS
https://github.com/teo-tsirpanis/DayGame/blob/master/Software%20Requirements%20Specification/Software_Requirements_Specification.pdf

IEEE Recommended Practice for Software Design Specifications:
http://cengproject.cankaya.edu.tr/wp-content/uploads/sites/10/2017/12/SDD-ieee-1016-2009.pdf

DayGame's GitHub page:
https://github.com/teo-tsirpanis/DayGame

MIT License :
https://opensource.org/licenses/MIT

SDD template provided by instructors

Software development with the use of ICONIX methodology (Greek)
http://users.uom.gr/~achat/AdvSoftEng/ICONIX_eBook.pdf

Software Design Document, Testing, and Deployment and Configuration Management
https://arxiv.org/ftp/arxiv/papers/1005/1005.0595.pdf

SOFTWARE DESIGN DESCRIPTION
https://senior.ceng.metu.edu.tr/2015/musinspro/MusIns-Pro_SDD.pdf

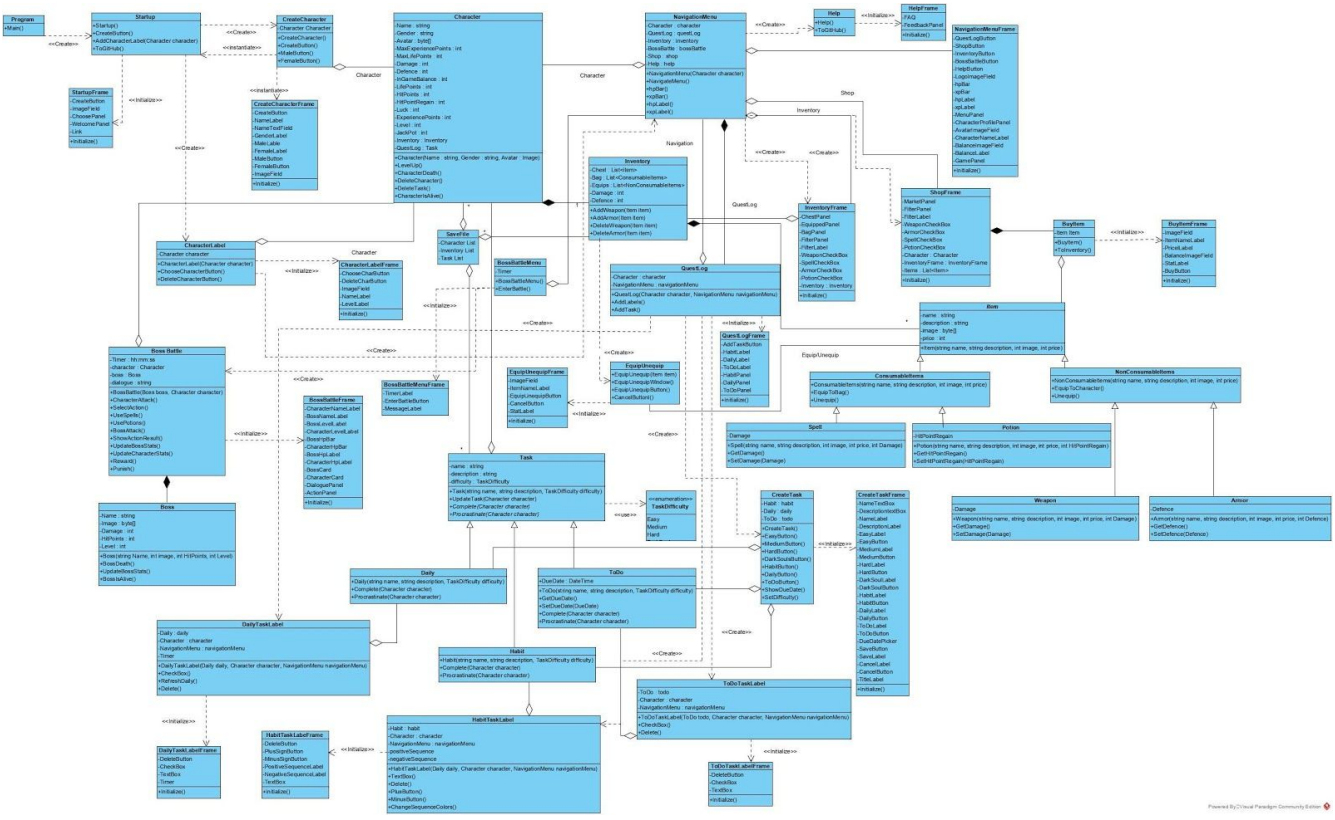
SDD Templates
https://sovannarith.files.wordpress.com/2012/07/sdd_template.pdf
https://ase.in.tum.de/stars.globalse.org/stars1/docs/SDD/SDDTemplate.html

2. System Design

2.1 Databases

DayGame will store the user data in a JSON file whose schema is described in the attached "DayGame-schema.json" file. The SaveFile contains information for each character, specifically it stores the current Characters data (HP, Experience, etc), information about the Tasks they are assigned to and their Inventory.

2.2 System Static View



<Program>

Class Identity: <1>
This class executes the program.

- Method <Main>
  The main method of the program.

<StartUp>

Class Identity: <2>
This class develops the functionality of the Start up menu. From here you can choose a character, create a new character or delete a character.

- Method <CreateButton>
  This button pops up the create character window.

- Method <AddCharacterLabel>
  This method creates a label with the character. This is shown in the choose character list.

- Method <ToGitHub>
  This method contains an image with a link to the GitHub repository.

<StartUpFrame>

Class Identity: <2.1>

This class contains the GUI of the StartUp class.

- Method <Initialize>
  This method initializes the components of the StartUp class. It contains 2 panels. The first is the option to choose a character and the second is a welcome message. It also contains the buttons for the choose character option and the delete character option.

<CreateCharacter>

Class Identity: <3>

This class creates the character.

- Method <MaleButton>
  The user can press this button to choose the male gender. It also changes the character's avatar.

- Method <CreateButton>

  The user can press this button to choose the female gender. It also changes the character's avatar.

- Method <CreateButton>

  After the user has selected his gender and his name, he can press this button to create his character.

<CreateCharacterFrame>

Class Identity: <3.1>

This class contains the GUI of the Create character class.

- Method <Initialize>

  This method initializes the components of the Create character class. It contains an image with the game logo, a text field to choose his name and two buttons to choose his gender. After that he can create the character by pressing the create button.

<CharacterLabel>

Class Identity: <4>

This class contains the functionality of the character label.

- Method <ChooseCharacterButton>

  When the user presses this button, it loads the saved file of that selected character and pops up the Quest Log.

- Method <DeleteCharacterButton>

  When the user presses this button, it pops up a window confirming the user's decision and if it is affirmative, it deletes the character's saved file while removing him from the choose character list.

<CharacterLabelFrame>

Class Identity: <4.1>

This class contains the GUI of the character label class.

- Method <Initialize>

  This method initializes the components of the Character label class. It contains the character image, his name, his level and the two decision buttons.

<NavigationMenu>

Class Identity<5>

This class contains the functionality of the navigation menu.

- Method<NavigateMenu>
  This method changes the panels in the program.

- Method<lpBar>
  This method creates the functionality of the life points bar in the character profile.

- Method<xpBar>
  This method creates the functionality of the experience points bar in the character. profile

- Method<lpLabel>
  This method creates the functionality of the life points label in the character profile.

- Method<xpLabel>
  This method creates the functionality of the experience points label in the character profile.

<NavigationMenuFrame>

Class Identity<5.1>

This class contains the GUI of the navigation menu class.

- Method <Initialize>
  This method initializes the components of the Navigation Menu class. Two bar menus are generated. The first is the navigation menu with the buttons and the second is the character profile with his avatar, his name, his level, his experience points, his life points and his in game balance.

<Character>

Class Identity: <6>

Character is the main class of our program. It represents the character that is utilized and contains information about the user's name, gender and other stats.

- Method <LevelUp>

  This method is called when xp is at the max level. Level is incremented, as well as MaxExperiencePoints and MaxLifePoints.

- Method <CharacterDeath>

  This method is called when Life Points is zero. As a result, the level is decreased.

- Method <DeleteCharacter>

  This method deletes the current character. It also deletes the saved file.

- Method <CharacterIsAlive>

  This method gets the life points or hit points, when he is located in the boss battle, and returns true if the Hit Points/ LifePoints are above 1.

- Method<UpdateCharacterStats>
  Character Stats are updated in the card and the hp bar when he loses hit points or gains hit points. This method is used only in Boss Battle.

<Shop>

Class Identity<7>

This class develops the functionality of the Shop. It contains all the items that can be purchased during the game.

<ShopFrame>

Class Identity<7.1>

This class contains the GUI of the Shop Class.

- Method <Initialize>

  This method initializes the components of the ShopFrame class. Shop contains 2 panels. The first is the filter panel and the second is the Marketplace with all the listed items.

<BuyItem>

Class Identity<8>

This class is responsible for the purchase of an item. The class checks the balance of the character and the price of the item. If the balance is enough, the user can buy the item and it will be automatically be transported to Inventory.

- Method <BuyItem>
  This method purchases an item and updates the new balance of the character. Method ToInventory is called to place it in the inventory.

- Method <ToInventory>
  This method takes the item and places it in the inventory.

<BuyItemFrame>

Class Identity<8.1.>

This class contains the GUI of the Buy Item Class.

- Method <Initialize>
  This method initializes the components of the BuyItem class. A window is generated with the item image, the stat that increases, the price and the buy button.

<Inventory>

Class Identity<9>

This class contains all the items that the user has purchased.

- Method <AddWeapon>
  Adds a weapon type item in the Equip List.

- Method <AddArmor>
  Adds an armor type item in the Equip List.

- Method <AddSpell>
  Adds a spell type item in the Bag List.

- Method <AddPotion>
  Adds a potion type item in the Bag List.

- Method <DeleteWeapon>
  Deletes a weapon type item from the Equip List.

- Method <DeleteArmor>
  Deletes an armor type item from the Equip List.

- Method <DeleteSpell>
  Deletes a spell type item from the Bag List.

- Method <AddPotion>
  Deletes a potion type item from the Bag List.

<InventoryFrame>

Class Identity<9.1.>

This class contains the GUI of the Inventory Class.

- Method <Initialize>
  This method initializes the components of the Inventory class. Inventory is divided into 4 panels. The first is the item filter the second is the chest, the third is the equip panel and the fourth is the bag panel.

<EquipUnequip>

Class Identity<10>

This class contains the Equip and the Unequip Item functionality.

- Method <EquipUnequipWindow>
  Creates the window to equip and unequip items.

- Method <EquipUnequipButton>
  This method contains the functionality for the equip and unequip button.

- Method <CancelButton>
  This method contains the functionality for the cancel button.

<EquipUnequipFrame>

Class Identity<10.1.>

This class contains the GUI of the EquipUnequip Class.

- Method <Initialize>
  This method initializes the components of the EquipUnequip class. A window is generated with the item image, the stat that increases when equipped or decreases with unequipped and the appropriate button.

<Item>

Class Identity<11>

This class contains the characteristics of the item.

<ConsumableItems>

Class Identity<11.1>

This class inherits the Item class and contains the characteristics of the consumable items.

<Spell>

Class Identity<11.1.1>

This class inherits the Consumable item class and contains the characteristics of the spells.

<Potion>

Class Identity<11.1.2>

This class inherits the Consumable item class and contains the characteristics of the potions.

<NonConsumableItems>

Class Identity<11.2>

This class inherits the Item class and contains the characteristics of the non-consumable items.

<Armor>

Class Identity<11.2.1>

This class inherits the Non-Consumable item class and contains the characteristics of the armor.

<Weapon>

Class Identity<11.2.2>

This class inherits the Non-Consumable item class and contains the characteristics of the weapons.

<QuestLog>

Class Identity<12>
This class contains the functionality of the Quest Log.

- Method <AddLabel>
  This method creates the task labels depending on the task type.

- Method <AddTask >
  This method pops up the create task window.

<QuestLogFrame>

Class Identity<12.1>
This class contains the GUI of the Quest Log class.

- Method <Initialize>
  This method initializes the components of the Quest log class. It has three panels where each task will be located with their labels and the add task button.

<Task>

Class Identity<13>

Abstract class which implements dailies, habits and to-dos.

- Method<UpdateTask>
  Abstract method that calls either Complete or Procrastinate and LevelUps if xp is at max level.

- Method<Complete>
  Abstract method that is implemented in each category of tasks.

- Method<Procrastinate>
  Abstract method that is implemented in each category of tasks.

<Habit>

Class Identity<13.1>

This class inherits the Task class and implements the Habits.

- Method<Complete>
  When habit is positive the character gains experience points and in game balance.

- Method<Procrastinate>
  When habit is negative the character loses life points, experience points and in game balance.

<Daily>

Class Identity<13.2>

This class inherits the Task class and implements the Dailies.

- Method<Complete>
  When the checkbox is checked the character gains experience points and in game balance.

- Method<Procrastinate>
  When the user unchecks the checkbox the character loses life points, experience points and in game balance.

<To-Dos>

Class Identity<13.3>

This class inherits the Task class and implements the To-Dos.

- Method<Complete>

  When the checkbox is checked the character gains experience points and in game balance.

- Method<Procrastinate>

  When the due time has passed the character loses life points, experience points and in game balance.

<TaskDifficulty>

Enumeration

- 1: Easy
- 2: Medium
- 3: Hard
- 4: DarkSouls

<TaskLabel>

Class Identity<19>

This class contains all the labels for the tasks. Habit has 3 buttons and a text field, Daily has a checkbox, a button and a text field and To-do has the same with Daily with the addition of the due date

<CreateTask>

Class Identity<14>

This class contains the functionality of creating a task.

- Method<DifficultyButtons>

  This method implements the functionality of the difficulty buttons in the create task window.

- Method<TaskType>

  This method implements the functionality of the task type buttons in the create task window.

- Method<DueDate>
  This method sets the due date when To-Do button is pressed.

<CreateTaskFrame>

Class Identity<14.1>
This class contains the GUI of create task class.

- Method <Initialize>
  This method initializes the components of the Create task class. It has texts box for the name and the description, the buttons for the task type and the task difficulty and showing the due date pop up.

<BossBattle>

Class Identity<15>
This class implements a digital fight between a *Boss* and a *Character.* It is called randomly between 2-7 days.

- Method<SelectAction>
  The user selects to Attack, to Use Spell or Use Potion.

- Method<CharacterAttack>
  The Character deals damage to the Boss. The amount of damage is based on the level.

- Method<UseSpells>
  Damage that is assigned to the spell that the user chooses is dealt to the Boss. The spell is deleted from the bag.

- Method<UsePotions>
  Health that is assigned to the potion is added to HitPoints. The potion is deleted from the bag.

- Method<BossAttack>
  After user's turn, the boss attacks and deals damage to the HitPoints of the Character.

- Method<ShowActionResult>
  A proper message is shown in the dialogue panel each time an action is taken place.

- Method<Reward>
  Character gains experience points and in game balance.

- Method<Punish>
  Character loses all the equipped items.

<BossBattleFrame>

Class Identity<15.1>
This class contains the GUI of the Boss Battle class.

- Method <Initialize>

  This method initializes the components of the Boss Battle class. It contains three panels. The first is where the fight takes place, the second is the dialogue, and the third is the action selection.

<Boss>

Class Identity<16>

The boss that takes part in the BossBattle.

- Method<BossIsAlive>

  If Boss's HitPoints are above 1 then he is alive. This method returns the Boolean true if he is alive or the Boolean false is he is not alive.

- Method<UpdateBossStats>
  Boss Stats are updated in the card and the hp bar when he loses hit points.

<BossBattleMenu>

Class Identity<17>

This is where the user will wait to enter the battle. It has a countdown timer and a button.

- Method<EnterBattle>

  When the timer is over the user has some time to prepare and then he can press the button to enter the button. The Boss Battle window will pop up.

<BossBattleMenuFrame>

Class Identity<17.1>

This class contains the GUI of the Boss battle menu.

- Method <Initialize>

  This method initializes the components of the Boss Battle Menu class. It has the countdown timer label and the enter battle button.

<Help>

Class Identity<18>
This class contains the functionality of the help tab. It has a FAQ and an option to give feedback.

- Method <ToGitHub>
  This method opens the repository in our github to give feedback.

<HelpFrame>

Class Identity<18.1>
This class contains the GUI of the help class.

- Method <Initialize>
  This method initializes the components of the Help class.

<FileSave>

Class Identity<20>
This class contains the functionality of the autosave. The class stores into JSON format the character class, the inventory and the task

2.3 Dynamic System View

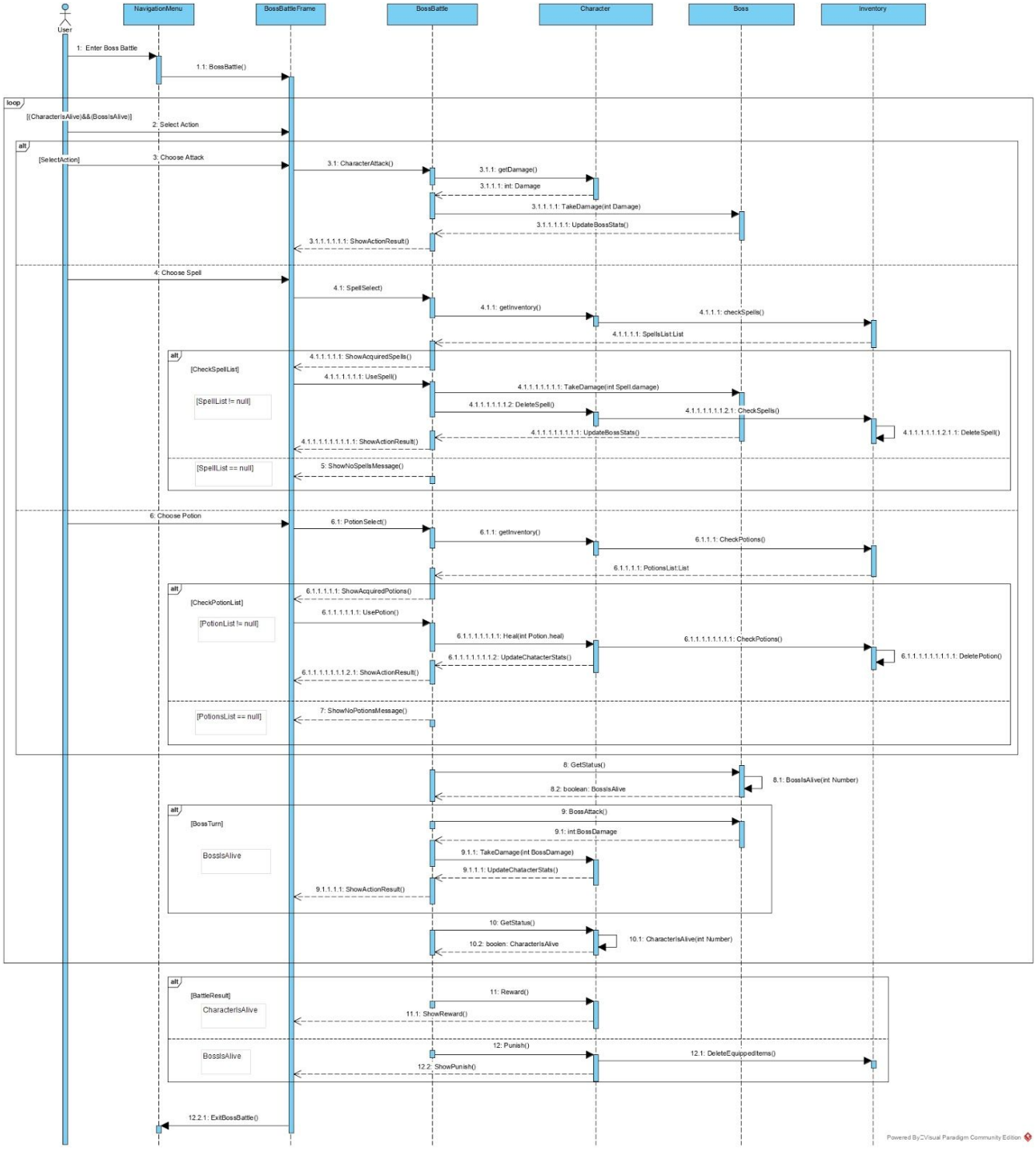
A traceability matrix containing major classes used in the UCs specified in the SRS is given below. Classes resulted from Abstraction, Inheritance , Implementation of class copies and/or interfaces are not included for readability purposes.

Requirement Identifiers	UC-01	UC-02	UC-03	UC-04	UC-05	UC-06	UC-07	UC-08	UC-09	UC-10	UC-11
StartUpFrame	X	X	X								
ChooseCharacterFrame	X	X	X								
MainFrame				X							
QuestLogFrame					X	X	X				
Task					X	X	X				
TaskDifficulty					X	X					
InventoryFrame									X		
Inventory									X	X	
ShopFrame								X			
Shop								X			
Character									X	X	
Item									X	X	
Boss										X	
BossBattleFrame										X	
BossBattle										X	
HelpFrame											X
SaveFile											

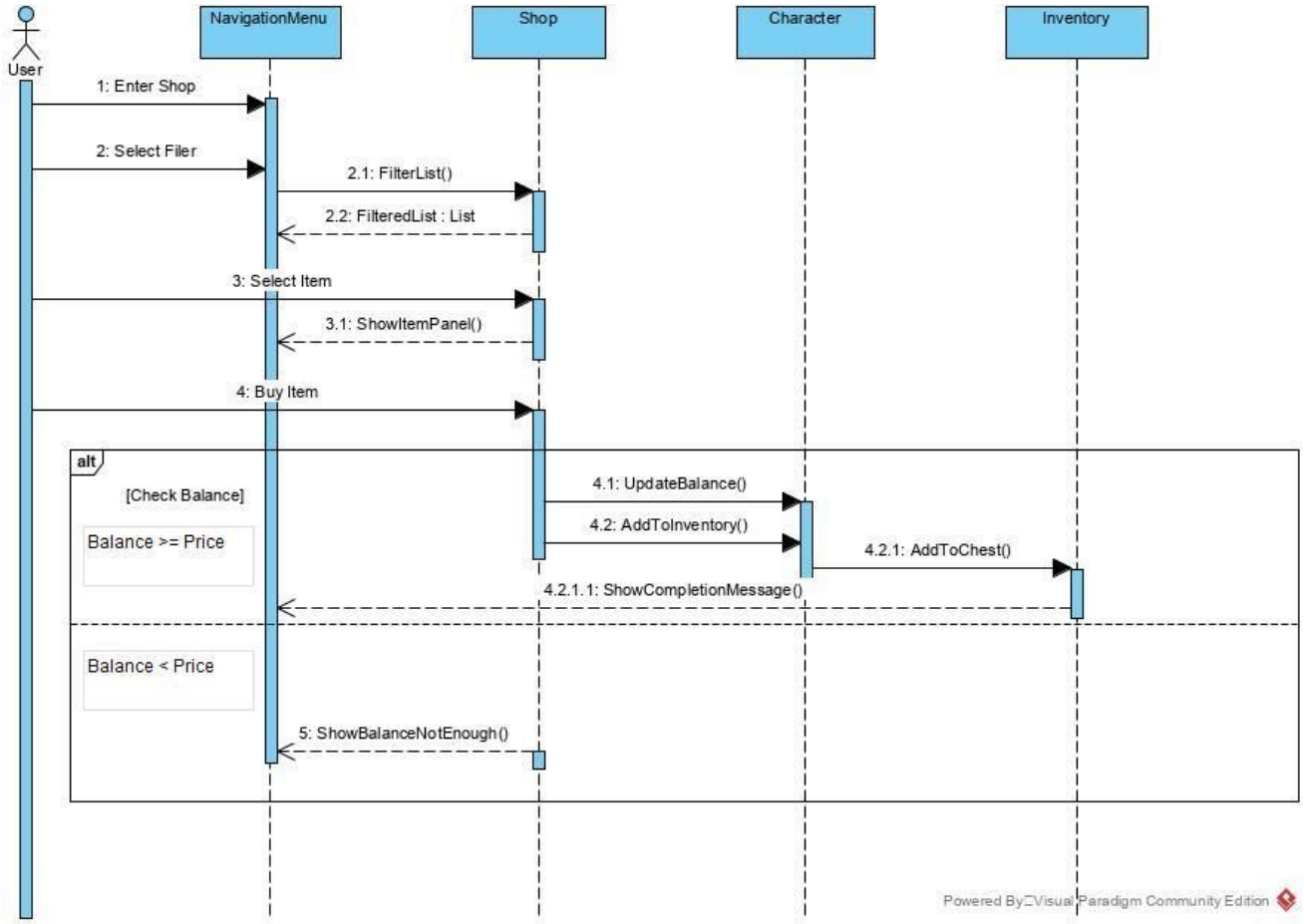
2.3.1 Sequence Diagrams

The following Sequence Diagrams showcase the flow events that take place in the use cases mentioned in the SRS

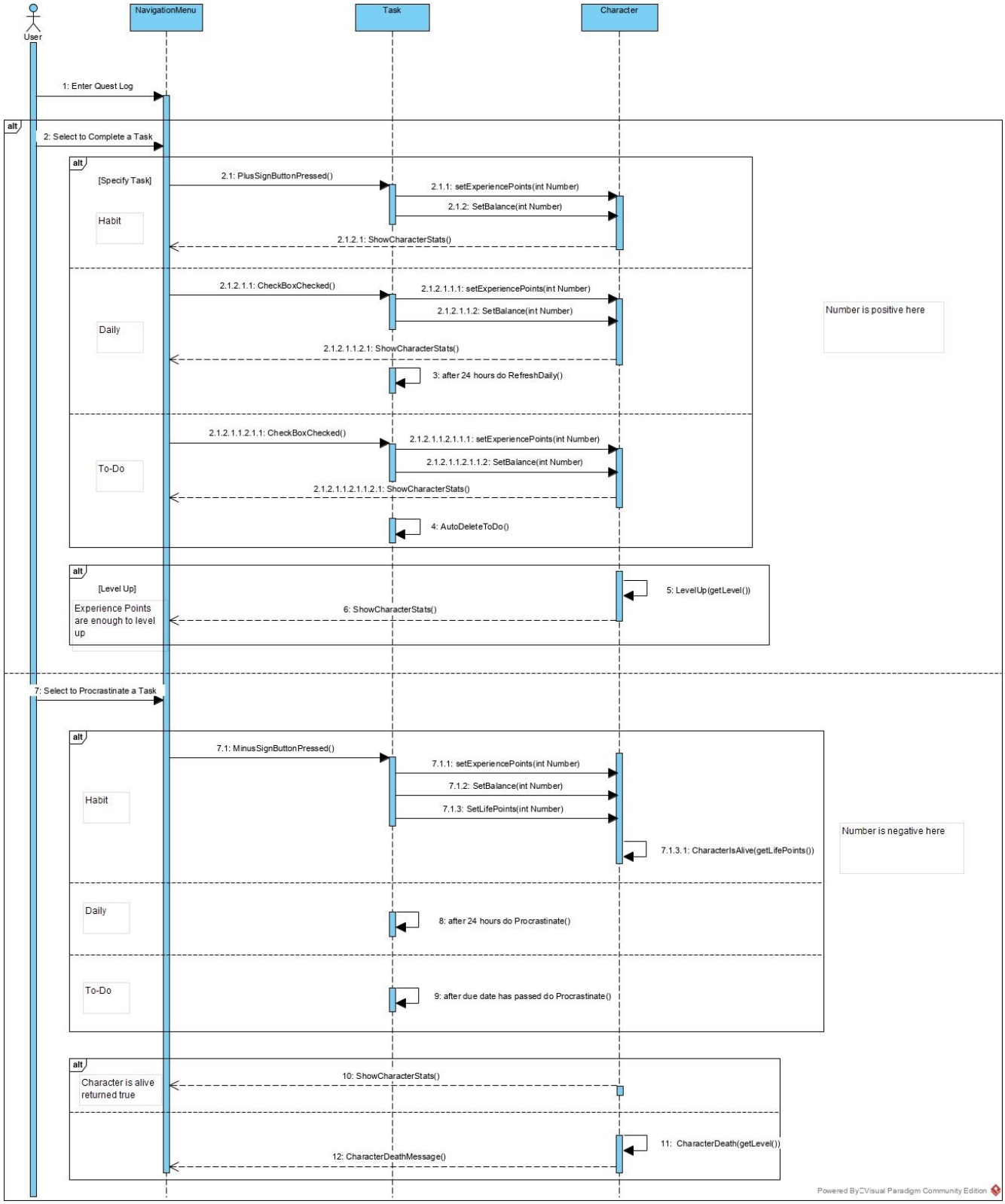
1. Boss Battle



2. Shop

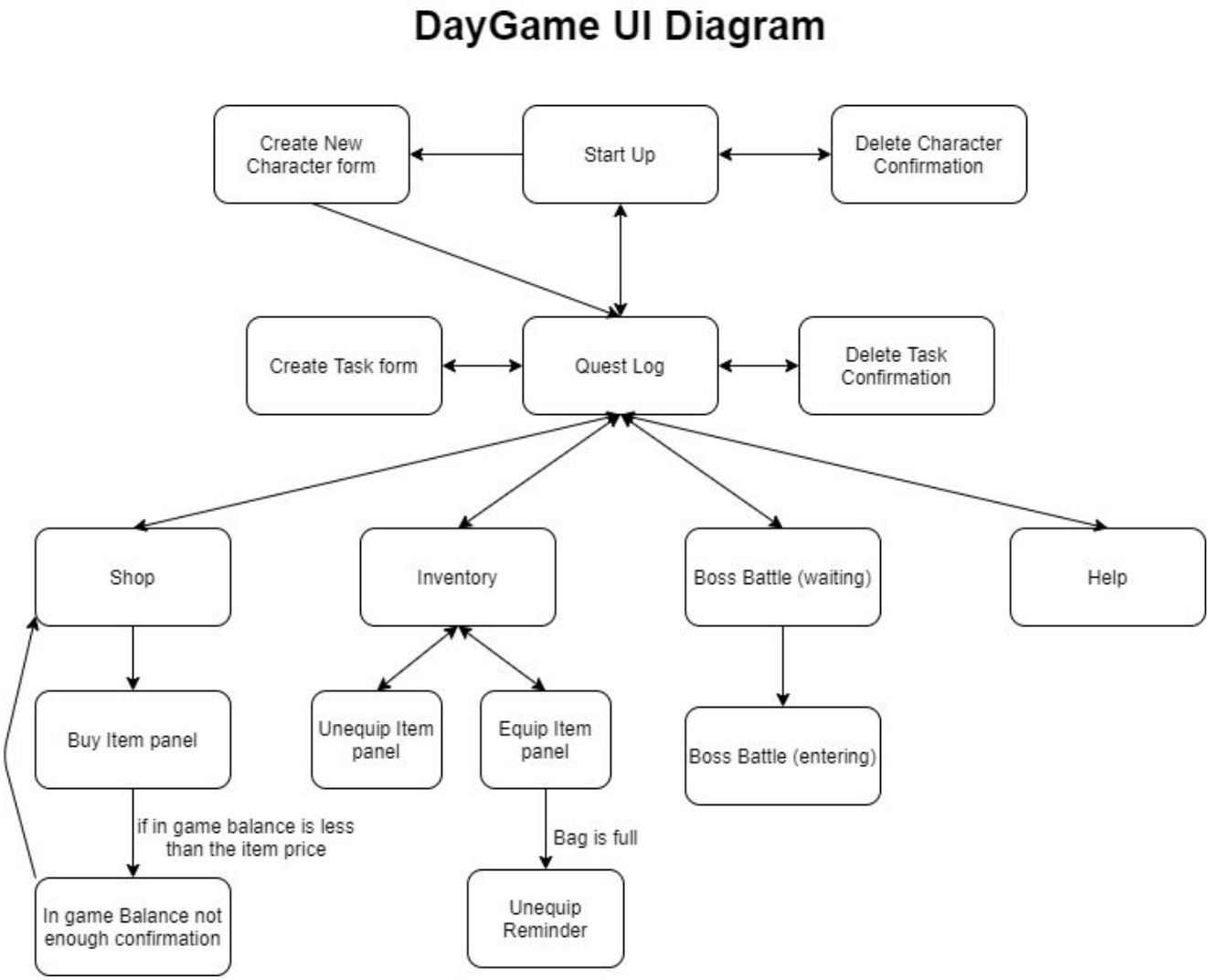

User
NavigationMenu
Shop
Character
Inventory
1: Enter Shop
2: Select Filer
2.1: FilterList()
2.2: FilteredList : List
3: Select Item
3.1: ShowItemPanel()
4: Buy Item
alt
[Check Balance]
Balance >= Price
4.1: UpdateBalance()
4.2: AddToInventory()
4.2.1: AddToChest()
4.2.1.1: ShowCompletionMessage()
Balance < Price
5: ShowBalanceNotEnough()
Powered By Visual Paradigm Community Edition

3. Update Task



3. Interface Explanations

A family friendly UI design was developed to enrich the visual aspect of DayGame. The main color of the app will be Green and the close shades offering relaxation and freshness to the way the human eyes perceive it, while also providing harmony and a sense of growth with the color of nature. Perfect match for the app. In terms of functionality, a user friendly navigation was created where the user has a quick and easy access to most portion of the game. Icons and shapes are easy to read and their functionality is easy to understand.

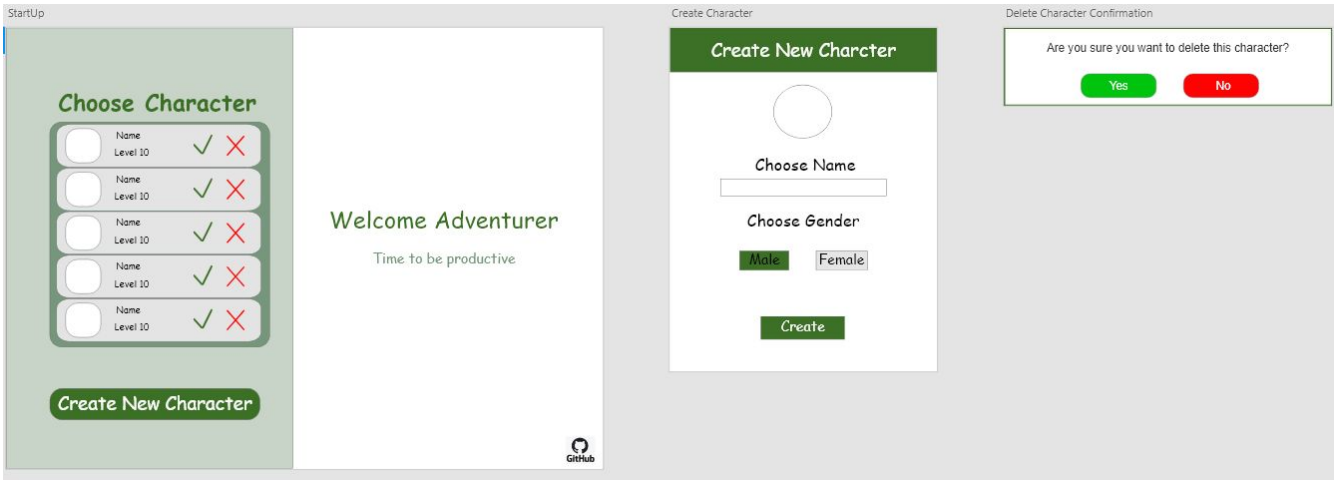


3.1 User Interfaces

Below screenshots of the UI are referenced. For the creation Adobe XD was used and we have provided a quick demo of how the app will function. When the .xd file is opened, you can press the start button and navigate like it is a real app. All the buttons have some kind of a functionality.

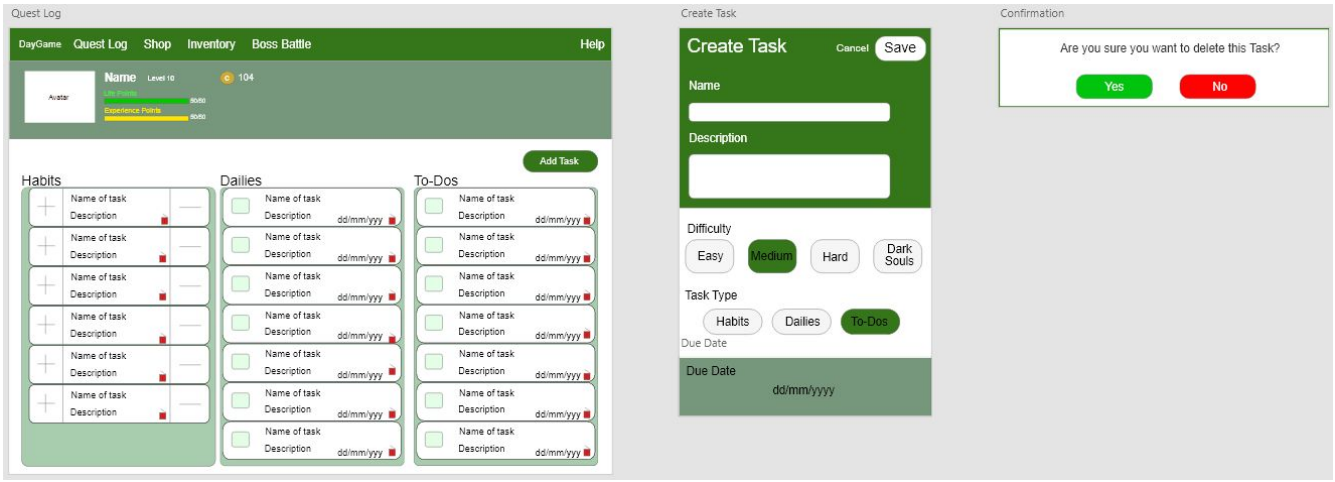
Start Up

When the user opens the app, he is greeted with the start up window. There he can select a character to enter the game or he can create a new one, where a form pops up, or he can delete an existing character where a confirmation window pops up. Start up window will also feature a link to our repository.

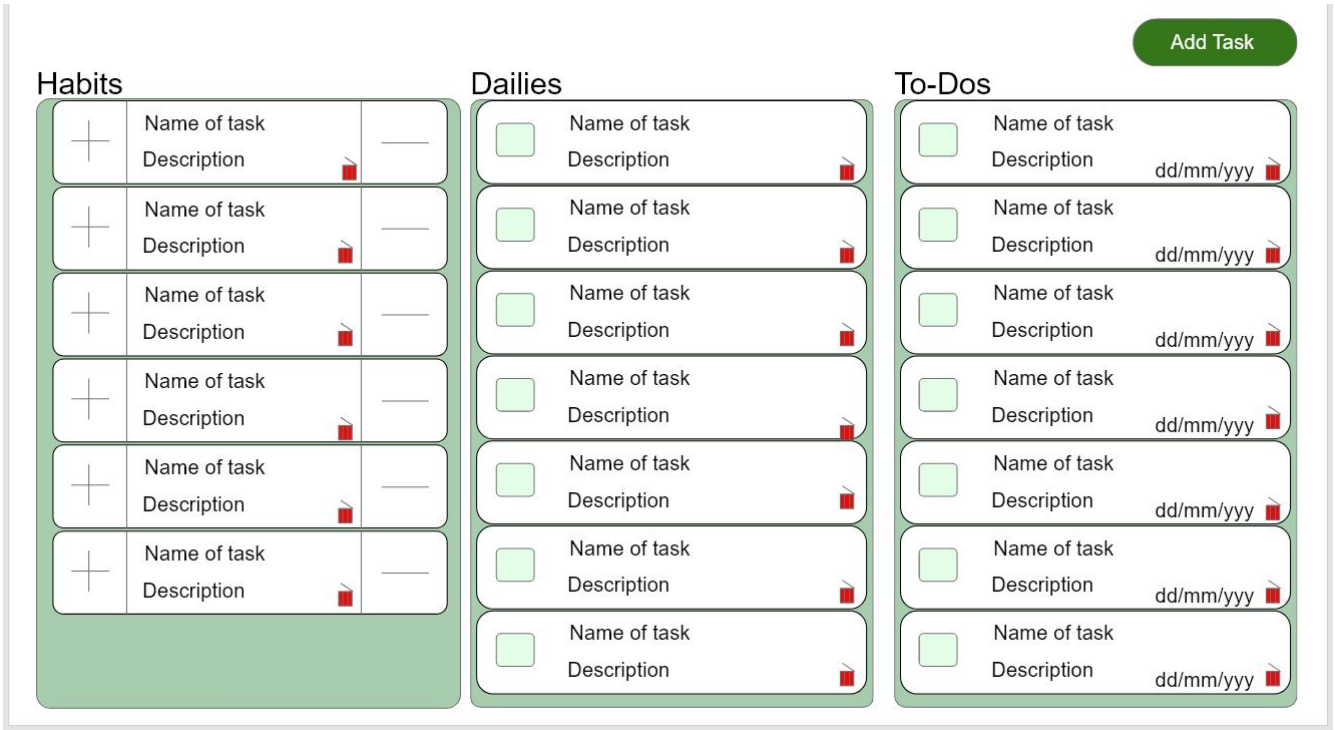


Quest Log

This is the main page of the program where the user will spend most of him time. Here Tasks are located and divided into specific sections. Each task will have the option to get deleted by the user where a confirmation window will pop up. In addition he can create a new task via a button which opens up a form to enter the necessary data. When To-Do task type is selected a Due Date window be visible from the bottom of the form, for the user to enter the task's deadline.

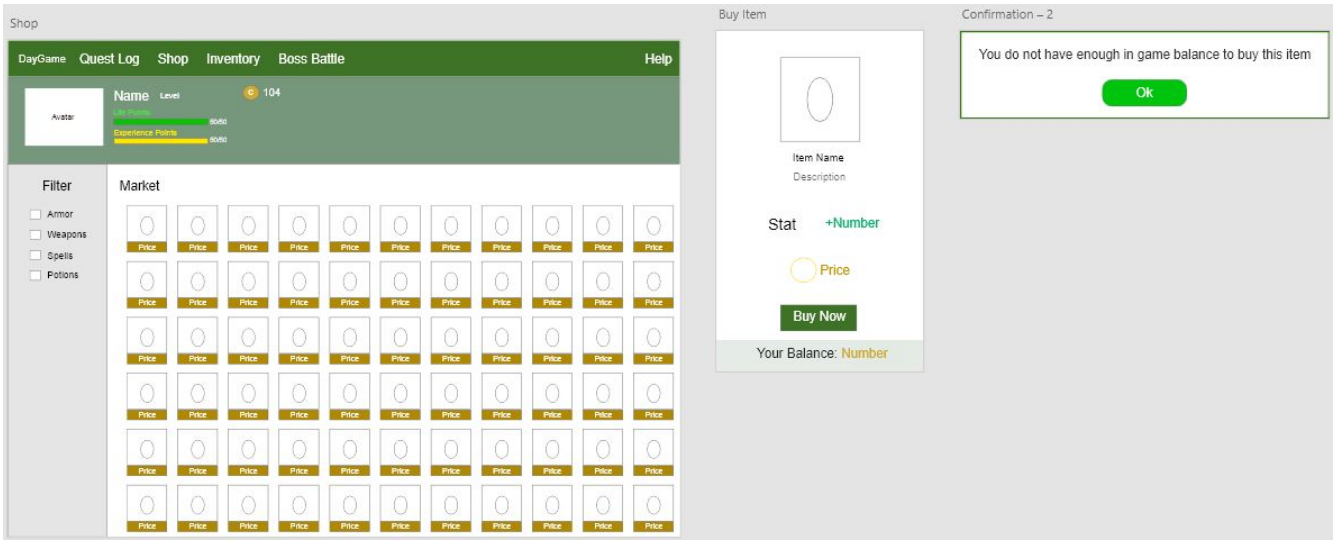


A closer look to Quest Log.



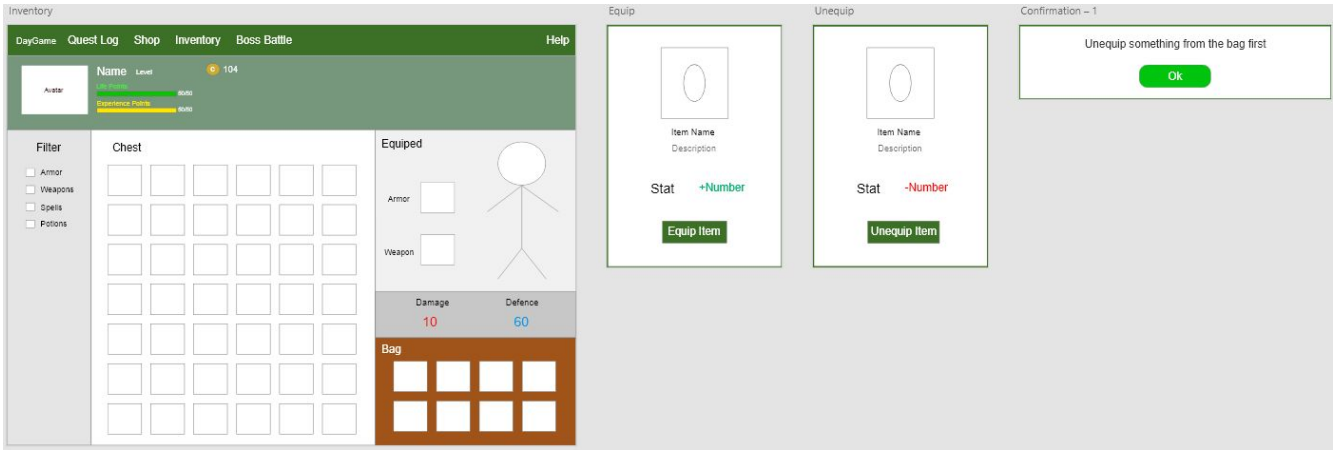
Shop

Shop is where all the items will be located with the option to add an item filter to make it easier for the user to buy an item. Each time user wishes to buy an item, he will have to click it first in order for the Buy Item panel to be shown. Insufficient amount of in game balance will result in an error message to pop up.



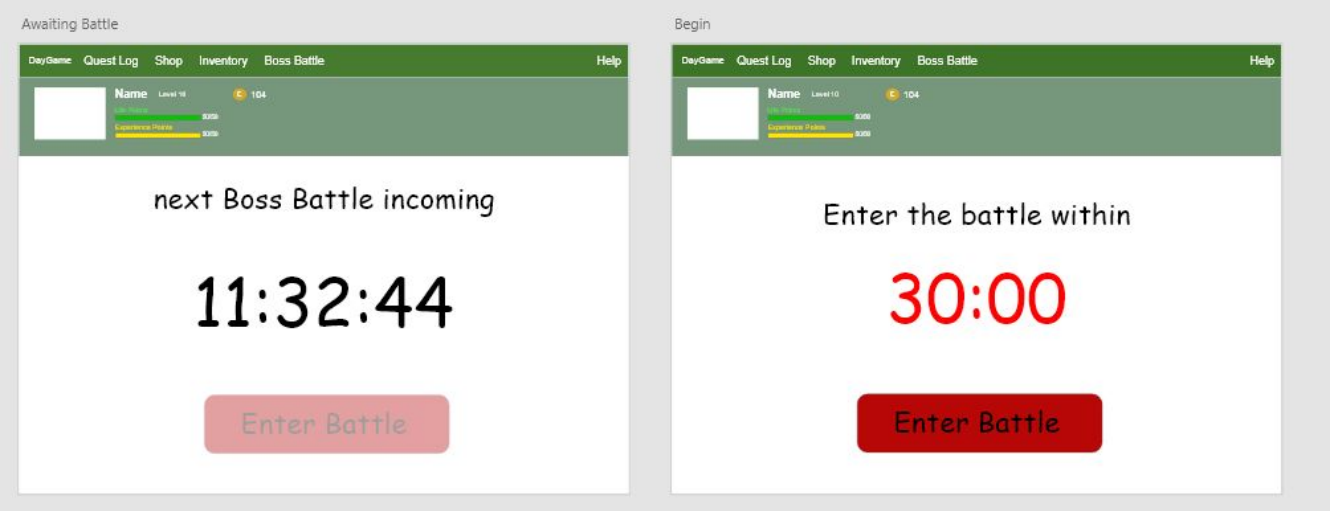
Inventory

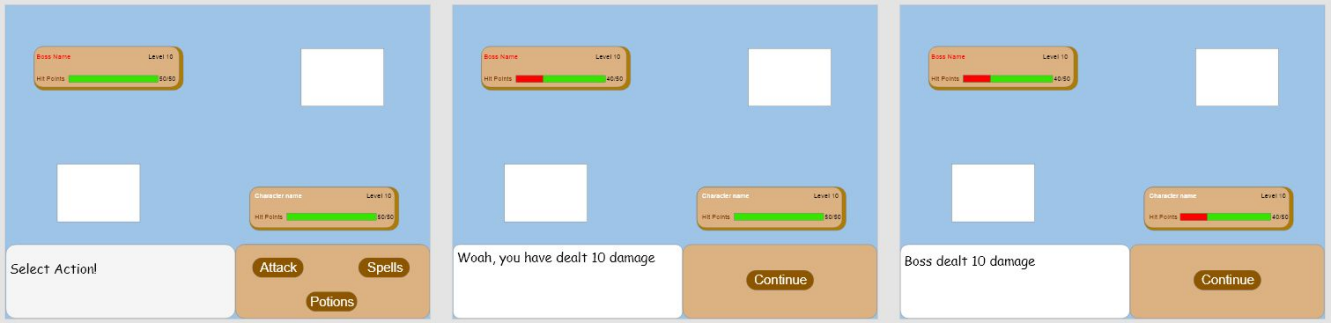
This is where all the bought items are located within the chest region. In addition this is where the character can equip items in order to increase his stats and survivability. By clicking the items in the chest, the Equip Item panel shows up. If although, the item about to be equipped has no free space and is a Consumable then a message error will pop up.



Boss Battle

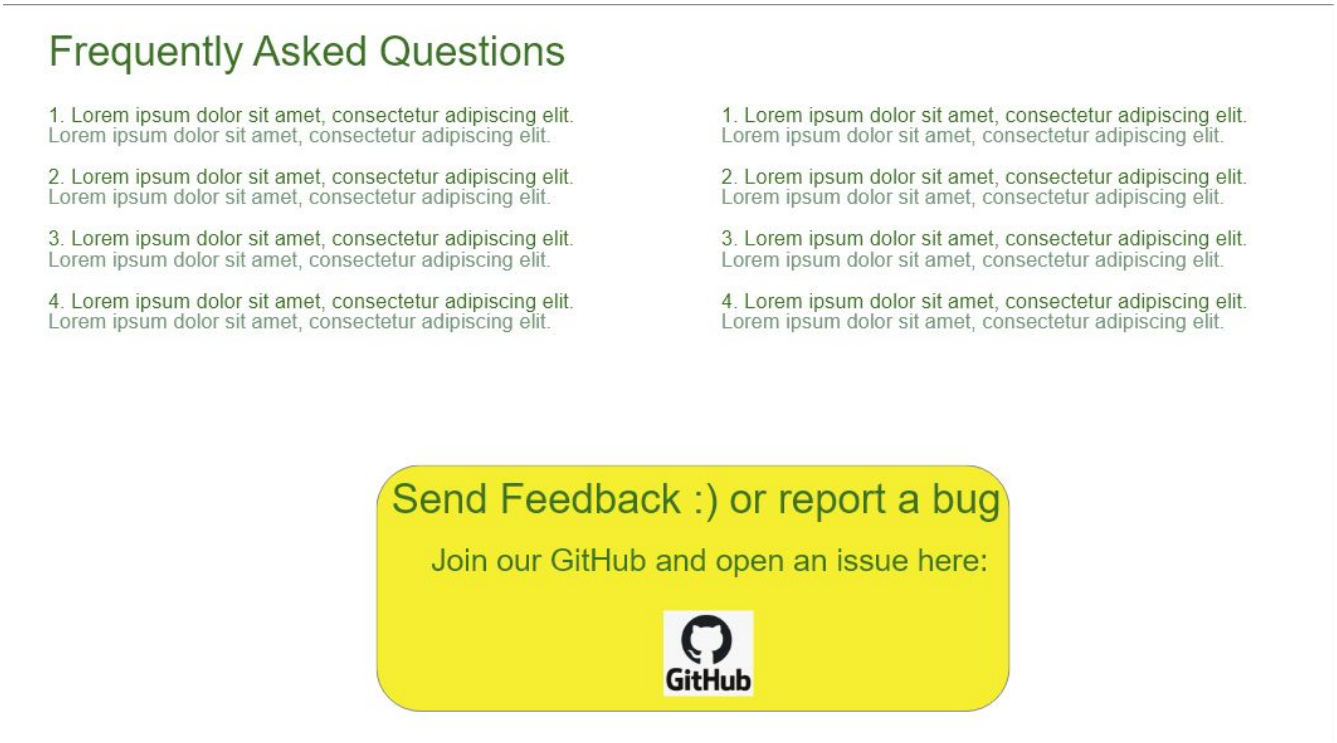
The Boss Battle has two states. The first is when the user is waiting to enter the battle with a countdown timer until the next boss shows up and the second is when he is entering the battle. While in the battle, there are two elevated cards with wood-like feel showing the stats for each entity. In the bottom the user selects his actions with a proper message shown.





Help

A FAQ will be written for the user to read in case of problems with the ability to provide feedback if he is still having trouble, prompting him to the issues page on the repository.



Navigation Bar and Character Profile

Navigation Bar and Character Profile will be always locked and shown when the user navigates through the app except when he is entering the battle.