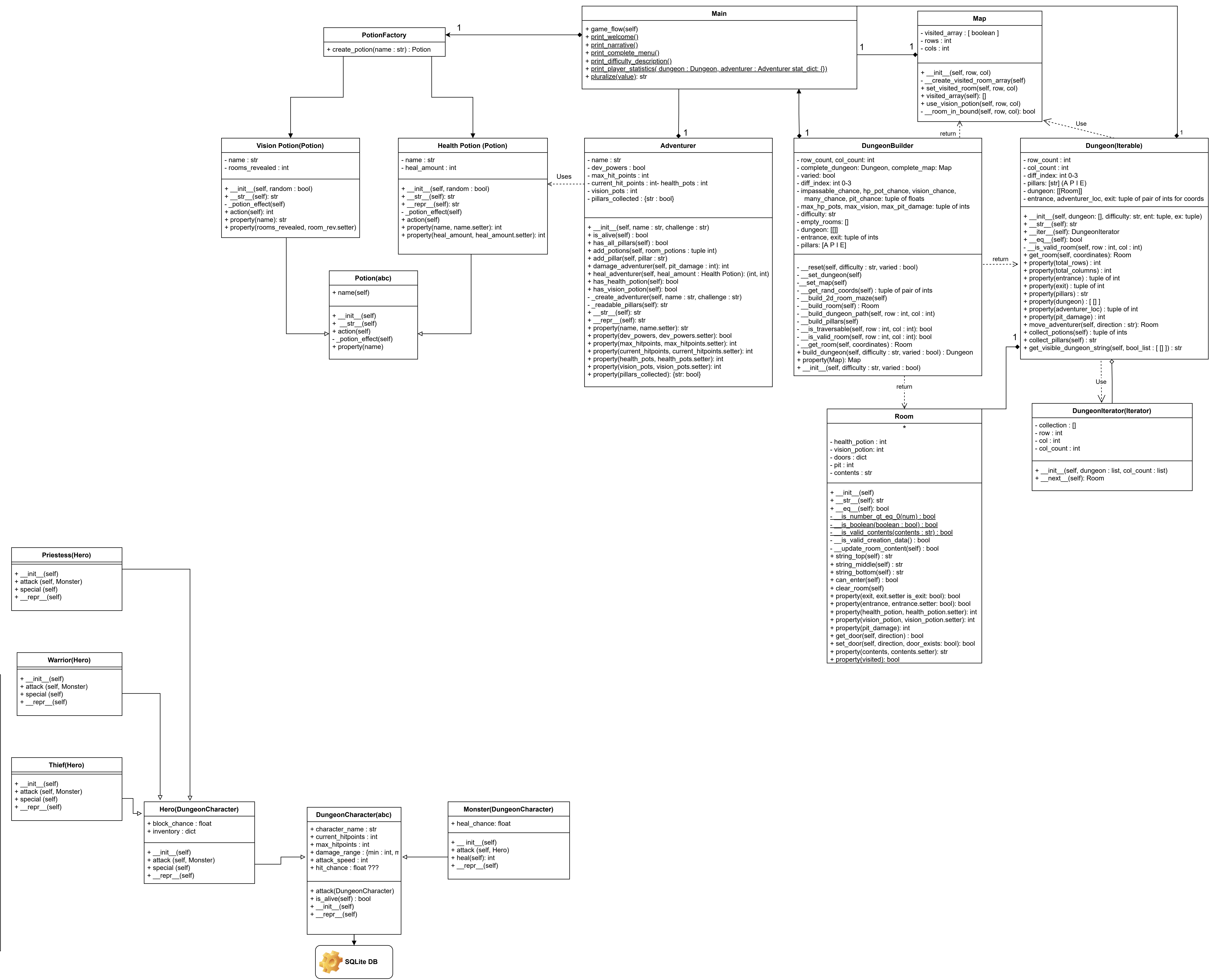


```
PotionFactory
+ create_potion(name : str) : Potion
```

```
Main
+ game_flow(self)
+ print_welcome()
+ print_narrative()
+ print_complete_menu()
+ print_difficulty_description()
+ print_player_statistics( dungeon : Dungeon, adventurer : Adventurer stat_dict: {})
+ pluralize(value): str
```

```
Map
- visited_array : [ boolean ]
- rows : int
- cols : int

+ __init__(self, row, col)
- __create_visited_room_array(self)
+ set_visited_room(self, row, col)
+ visited_array(self): []
+ use_vision_potion(self, row, col)
- __room_in_bound(self, row, col): bool
```

```
Vision Potion(Potion)
- name : str
- rooms_revealed : int

+ __init__(self, random : bool)
+ __str__(self): str
- _potion_effect(self)
+ action(self): int
+ property(name): str
+ property(rooms_revealed, room_rev.setter)
```

```
Health Potion (Potion)
- name : str
- heal_amount : int

+ __init__(self, random : bool)
+ __str__(self): str
+ __repr__(self): str
- _potion_effect(self)
+ action(self)
+ property(name, name.setter): int
+ property(heal_amount, heal_amount.setter): int
```

```
Potion(abc)
+ name(self)

+ __init__(self)
+  __str__(self)
+ action(self)
- _potion_effect(self)
+ property(name)
```

```
Adventurer
- name : str
- dev_powers : bool
- max_hit_points : int
- current_hit_points : int- health_pots : int
- vision_pots : int
- pillars_collected : {str : bool}

+ __init__(self, name : str, challenge : str)
+ is_alive(self) : bool
+ has_all_pillars(self) : bool
+ add_potions(self, room_potions : tuple int)
+ add_pillar(self, pillar : str)
+ damage_adventurer(self, pit_damage : int): int
+ heal_adventurer(self, heal_amount : Health Potion): (int, int)
+ has_health_potion(self): bool
+ has_vision_potion(self): bool
- _create_adventurer(self, name : str, challenge : str)
- _readable_pillars(self): str
+ __str__(self): str
+ __repr__(self): str
+ property(name, name.setter): str
+ property(dev_powers, dev_powers.setter): bool
+ property(max_hitpoints, max_hitpoints.setter): int
+ property(current_hitpoints, current_hitpoints.setter): int
+ property(health_pots, health_pots.setter): int
+ property(vision_pots, vision_pots.setter): int
+ property(pillars_collected): {str: bool}
```

```
DungeonBuilder
- row_count, col_count: int
- complete_dungeon: Dungeon, complete_map: Map
- varied: bool
- diff_index: int 0-3
- impassable_chance, hp_pot_chance, vision_chance,
  many_chance, pit_chance: tuple of floats
- max_hp_pots, max_vision, max_pit_damage: tuple of ints
- difficulty: str
- empty_rooms: []
- dungeon: [[]]
- entrance, exit: tuple of ints
- pillars: [A P I E]

- __reset(self, difficulty : str, varied : bool)
- __set_dungeon(self)
-__set_map(self)
- __get_rand_coords(self) : tuple of pair of ints
- __build_2d_room_maze(self)
- __build_room(self) : Room
- __build_dungeon_path(self, row : int, col : int)
- __build_pillars(self)
- __is_traversable(self, row : int, col : int): bool
- __is_valid_room(self, row : int, col : int): bool
- __get_room(self, coordinates) : Room
+ build_dungeon(self, difficulty : str, varied : bool) : Dungeon
+ property(Map): Map
+ __init__(self, difficulty : str, varied : bool)
```

```
Dungeon(Iterable)
- row_count : int
- col_count : int
- diff_index: int 0-3
- pillars: [str] (A P I E)
- dungeon: [[Room]]
- entrance, adventurer_loc, exit: tuple of pair of ints for coords

+ __init__(self, dungeon: [], difficulty: str, ent: tuple, ex: tuple)
+ __str__(self): str
+ __iter__(self): DungeonIterator
+ __eq__(self): bool
- __is_valid_room(self, row : int, col : int)
+ get_room(self, coordinates): Room
+ property(total_rows) : int
+ property(total_columns) : int
+ property(entrance) : tuple of int
+ property(exit) : tuple of int
+ property(pillars) : str
+ property(dungeon) : [ [] ]
+ property(adventurer_loc) : tuple of int
+ property(pit_damage) : int
+ move_adventurer(self, direction : str): Room
+ collect_potions(self) : tuple of ints
+ collect_pillars(self) : str
+ get_visible_dungeon_string(self, bool_list : [ [] ]) : str
```

```
Room
*
- health_potion : int
- vision_potion: int
- doors : dict
- pit : int
- contents : str

+ __init__(self)
+ __str__(self): str
+ __eq__(self): bool
- __is_number_gt_eq_0(num) : bool
- __is_boolean(boolean : bool) : bool
- __is_valid_contents(contents : str) : bool
- __is_valid_creation_data() : bool
- __update_room_content(self) : bool
+ string_top(self) : str
+ string_middle(self) : str
+ string_bottom(self) : str
+ can_enter(self) : bool
+ clear_room(self)
+ property(exit, exit.setter is_exit: bool): bool
+ property(entrance, entrance.setter: bool): bool
+ property(health_potion, health_potion.setter): int
+ property(vision_potion, vision_potion.setter): int
+ property(pit_damage): int
+ get_door(self, direction) : bool
+ set_door(self, direction, door_exists: bool): bool
+ property(contents, contents.setter): str
+ property(visited): bool
```

```
DungeonIterator(Iterator)
- collection : []
- row : int
- col : int
- col_count : int

+ __init__(self, dungeon : list, col_count : list)
+ __next__(self): Room
```

```
Priestess(Hero)

+ __init__(self)
+ attack (self, Monster)
+ special (self)
+ __repr__(self)
```

```
Warrior(Hero)

+ __init__(self)
+ attack (self, Monster)
+ special (self)
+ __repr__(self)
```

```
Thief(Hero)

+ __init__(self)
+ attack (self, Monster)
+ special (self)
+ __repr__(self)
```

```
Hero(DungeonCharacter)
+ block_chance : float
+ inventory : dict

+ __init__(self)
+ attack (self, Monster)
+ special (self)
+ __repr__(self)
```

```
DungeonCharacter(abc)
+ character_name : str
+ current_hitpoints : int
+ max_hitpoints : int
+ damage_range : {min : int, m
+ attack_speed : int
+ hit_chance : float ???

+ attack(DungeonCharacter)
+ is_alive(self) : bool
+ __init__(self)
+ __repr__(self)
```

```
Monster(DungeonCharacter)
+ heal_chance: float

+ __ init__(self)
+ attack (self, Hero)
+ heal(self): int
+ __repr__(self)
```

Dungeon Adventure
The Spoony Bard
Team: Halt Catch Fire
12/16/2021

Steph:
Room
Dungeon
DungeonBuilder
Kevin:
Adventurer
Potion
Potion Factory
Health Potion
Vision Potion
Xingguo:
Main
Map

Static methods are underlined.

Project Details: Dungeon Adventure 2.0

Expand your Dungeon Adventure program from 502 to incorporate the following features:

- Add an inheritance hierarchy of dungeon characters. These classes will be used to represent the hero (player) and monsters
    - DungeonCharacter is the parent/super class for the hierarchy. It should contain the following
        - is abstract
        - character name
        - health points/hit points
        - damage range (min and max)
        - attack speed (1 is slowest)
            - when battling attack speeds of the two opponents will be compared
            - a character can get multiple attacks per round of battle based on speed: a character that is twice as fast gets two attacks per round, a character that is three times as fast gets three attacks, etc.
            - you have freedom to adjust how this works
        - chance to hit (when attacking opponent)
        - constructor for initializing all fields provided by the class
        - properties as necessary for accessing/changing fields
        - an attack behavior (method)
            - this method passed the opponent to attack
            - if a character can attack (based on chance to hit), damage is generated in the min to max range for the character and applied to the opponent
            - provide a means to report success of attack or failure as necessary (this might be done inside the method or elsewhere depending on your design)
        - anything else you deem necessary (be creative and have fun :-)
    - Hero
        - Inherits from DungeonCharacter
        - Is abstract
        - A hero never gets fewer attacks than a monster (you can change this if you wish)
        - A hero has a chance to block an attack. This can be an integer or float.
        - Has a constructor that initializes all fields specific to Hero and calls the DungeonCharacter constructor
        - Heroes have a regular attack and also a special skill (skills for specific heroes will be defined below)
        - Any other fields or methods you deem necessary
    - Warrior
        - Inherits from Hero
        - Special skill is Crushing Blow that does 75 to 175 points of damage but only has a 40% chance of succeeding (you can adjust all numbers)
        - gets, sets, and any other methods you deem necessary (you may want to override the **attack** method to fit your Warrior – or not)
        - <u>suggested</u> statistics for Warrior (should be set up in constructor(s))
            - hit points: 125
            - attack speed: 4
            - chance to hit: 0.8 (80 percent)
            - minimum damage: 35
            - maximum damage: 60
            - chance to block: 0.2 (20 percent)
    - Priestess
        - Inherits from Hero
        - special skill is **heal** (choose a range of hit points that will be healed)
        - <u>suggested</u> statistics for Priestess (should be set up in constructor(s))
            - hit points: 75
            - attack speed: 5
            - chance to hit: 0.7 (70 percent)
            - minimum damage: 25
            - maximum damage: 45
            - chance to block: 0.3 (30 percent)
        - any other fields and methods you deem necessary
    - Thief
        - Inherits from Hero
        - Special skill is **surprise attack** -- 40 percent chance it is successful. If it is successful, Thief gets an attack and another turn (extra attack) in the current round. There is a 20 percent chance the Thief is caught in which case no attack at all is rendered. The other 40 percent is just a normal attack.
        - <u>suggested</u> statistics for Thief (should be set up in constructor(s))
            - hit points: 75
            - attack speed: 6
            - chance to hit: 0.8 (80 percent)
            - minimum damage: 20
            - maximum damage: 40
            - chance to block: 0.4 (40 percent)
        - any other fields and methods you deem necessary
    - Monster
        - Inherits from DungeonCharacter
        - Is abstract
        - constructor - should call base/super constructor
        - get, set, and any other methods (this includes overridden ones) you deem necessary
        - a **heal** method that is based on chance to heal and then range of heal points for monster
        - chance to heal (a Monster has a chance to heal after any attack that causes a loss of hit points -- this should be checked after the Monster has been attacked and hit points have been lost -- note that if the hit points lost cause the Monster to faint, it cannot heal itself!)
    - Ogre
        - Inherits from Monster
        - instance variables as you deem necessary (none may be necessary!)
        - gets, sets, and any other methods you deem necessary (you may want to override the **attack** method to fit your Ogre – or not)
        - <u>suggested</u> statistics for Ogre (should be set up in constructor(s) -- choose a name for your Ogre)
            - hit points: 200
            - attack speed: 2
            - chance to hit: 0.6 (60 percent)
            - minimum damage: 30
            - maximum damage: 60
            - chance to heal: 0.1 (10 percent)
            - minimum heal points: 30
            - maximum heal points: 60
    - Gremlin
        - Inherits from Monster
        - instance variables as you deem necessary (none may be necessary!)
        - gets, sets, and any other methods you deem necessary (you may want to override the **attack** method to fit your Gremlin)
        - suggested statistics for Gremlin (should be set up in constructor(s) -- choose a name for your Gremlin)

        - suggested statistics for Gremlin (should be set up in constructor(s)-- choose a name for your Gremlin)
            - hit points: 70
            - attack speed: 5
            - chance to hit: 0.8 (80 percent)
            - minimum damage: 15
            - maximum damage: 30
            - chance to heal: 0.4 (40 percent)
            - minimum heal points: 20
            - maximum heal points: 40

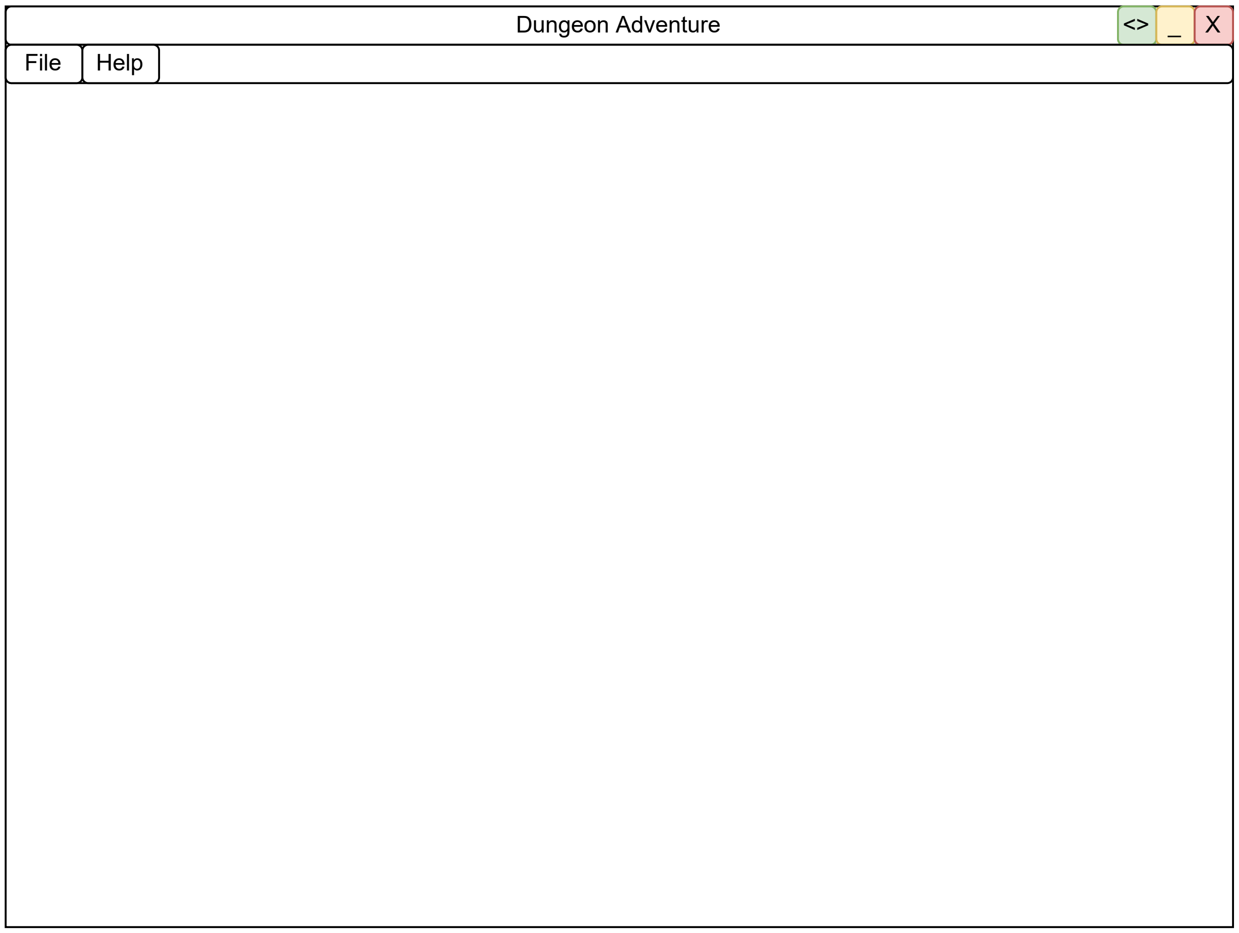
    - Skeleton
        - Inherits from Monster
        - instance variables as you deem necessary (none may be necessary!)
        - gets, sets, and any other methods you deem necessary (you may want to override the **attack** method to fit your Skeleton)
        - suggested statistics for Skeleton (should be set up in constructor(s)-- choose a name for your Skeleton)
            - hit points: 100
            - attack speed: 3
            - chance to hit: 0.8 (80 percent)
            - minimum damage: 30
            - maximum damage: 50
            - chance to heal: 0.3 (30 percent)
            - minimum heal points: 30
            - maximum heal points: 50

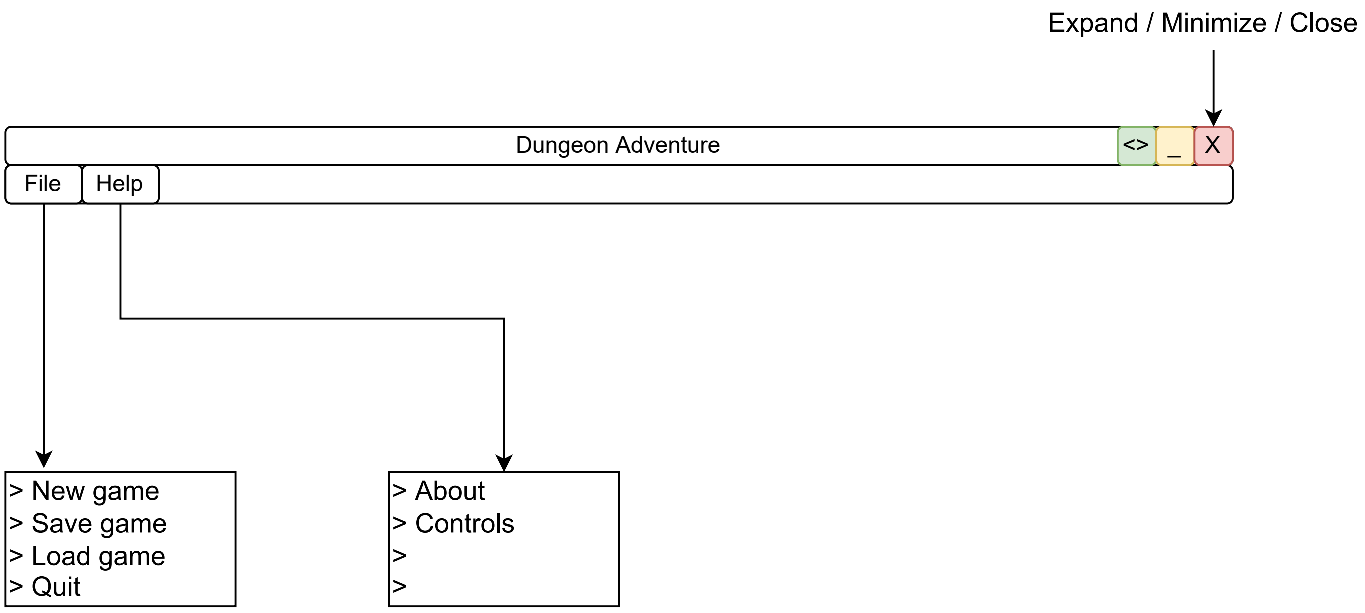
- Game Play
    - Player chooses a Hero (ask user for name of hero)
    - Monsters are randomly placed in rooms of dungeon
    - Stronger/special/more monsters should be placed with pillars and exit
    - Previous rules for Dungeon Adventure are still in place but you can modify things based on your team's vision for the game
    - Provide the ability to save and load a game
        - you can provide a single save, multiple saves, let the user choose the names of the save files, whatever you deem best :-)

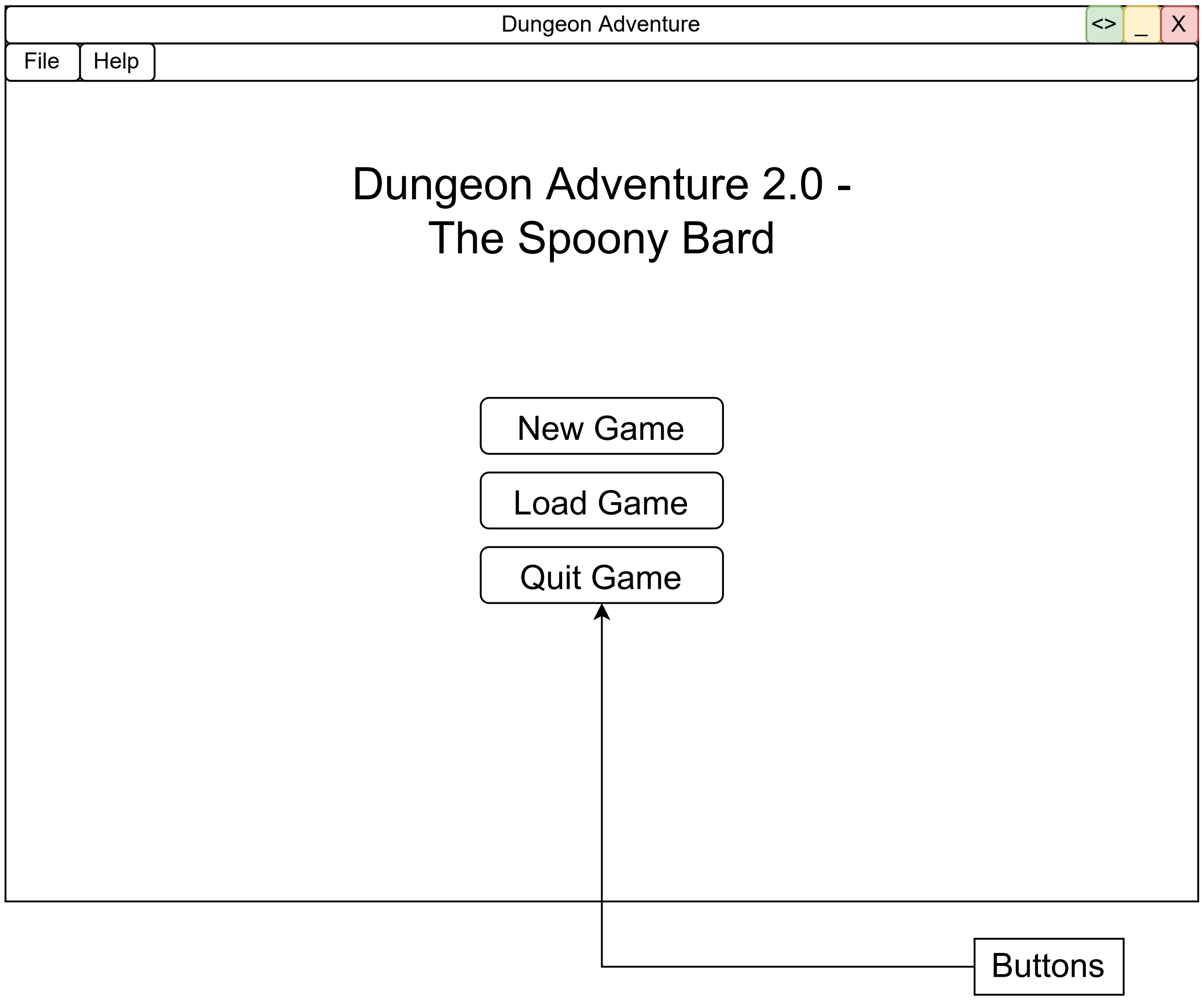
    - Once game is over provide the ability for the player to start a new game

- Store data for your monsters in a SQLite database (Monster name and statistics that go with that monster)
    - Retrieve this data at the start of your program and use it to generate and place monsters in the dungeon

- Extra Credit possibilities
    - creativity
    - difficulty levels
    - audio/video
    - custom assets (images, etc.)
    - 3D maze
    - Multiple heroes (a party)
    - Additional potion types (perhaps a bomb that can be used for massive damage against a monster)
    - ???


Dungeon Adventure
<>
_
X
File
Help


Expand / Minimize / Close
Dungeon Adventure
<>
_
X
File
Help
> New game
> Save game
> Load game
> Quit
> About
> Controls
>
>

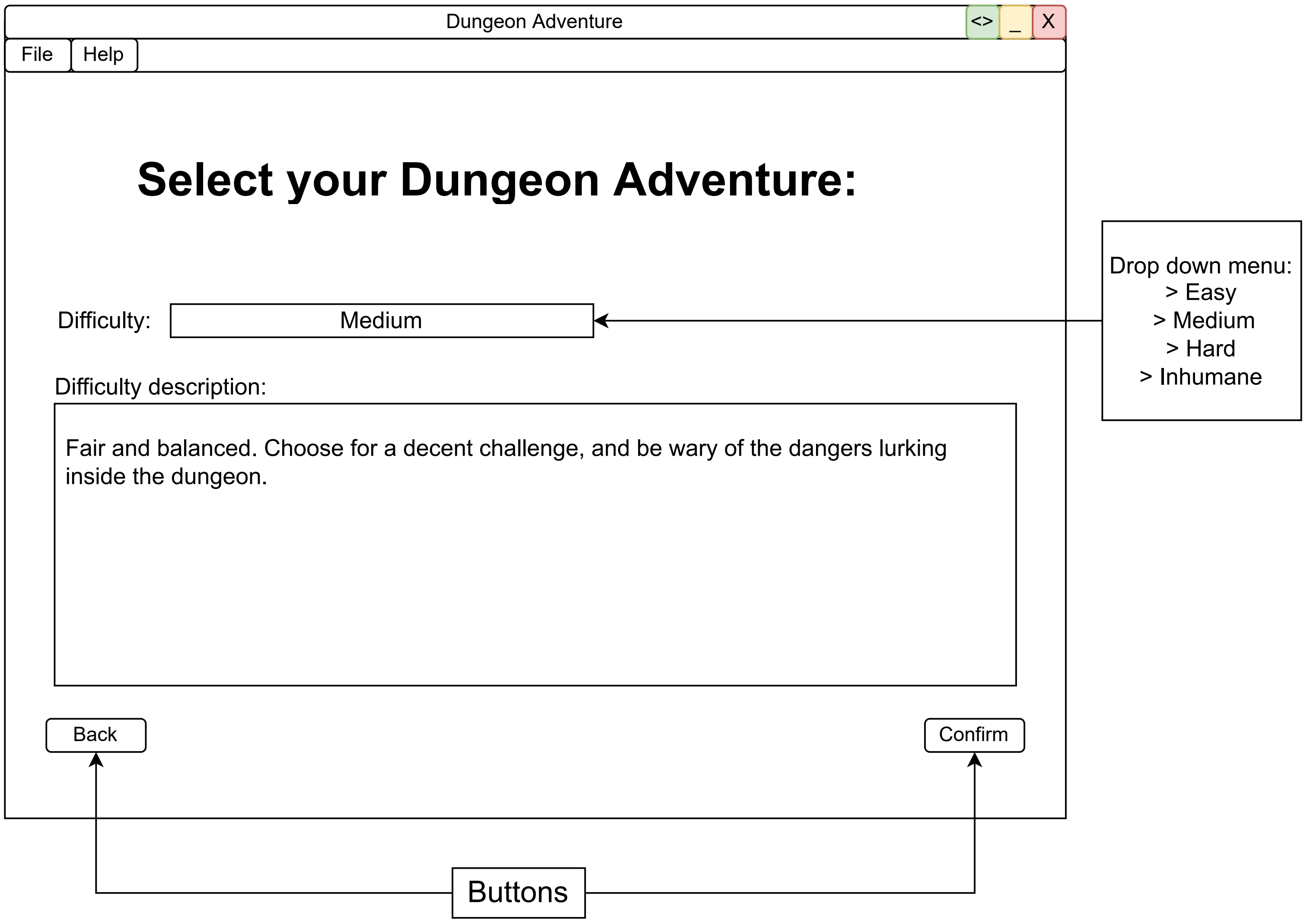


Game Start

User story: As a user, I want to play Dungeon Adventure.

The user will run the program and select from three different options.

- New Game: Player will start a completely new randomly generated Dungeon Adventure. Selecting this option will advance to another screen.
- Load Game: Player will resume a previously played game. Selecting this option will advance to another screen.
- Quit Game: Player will exit program.



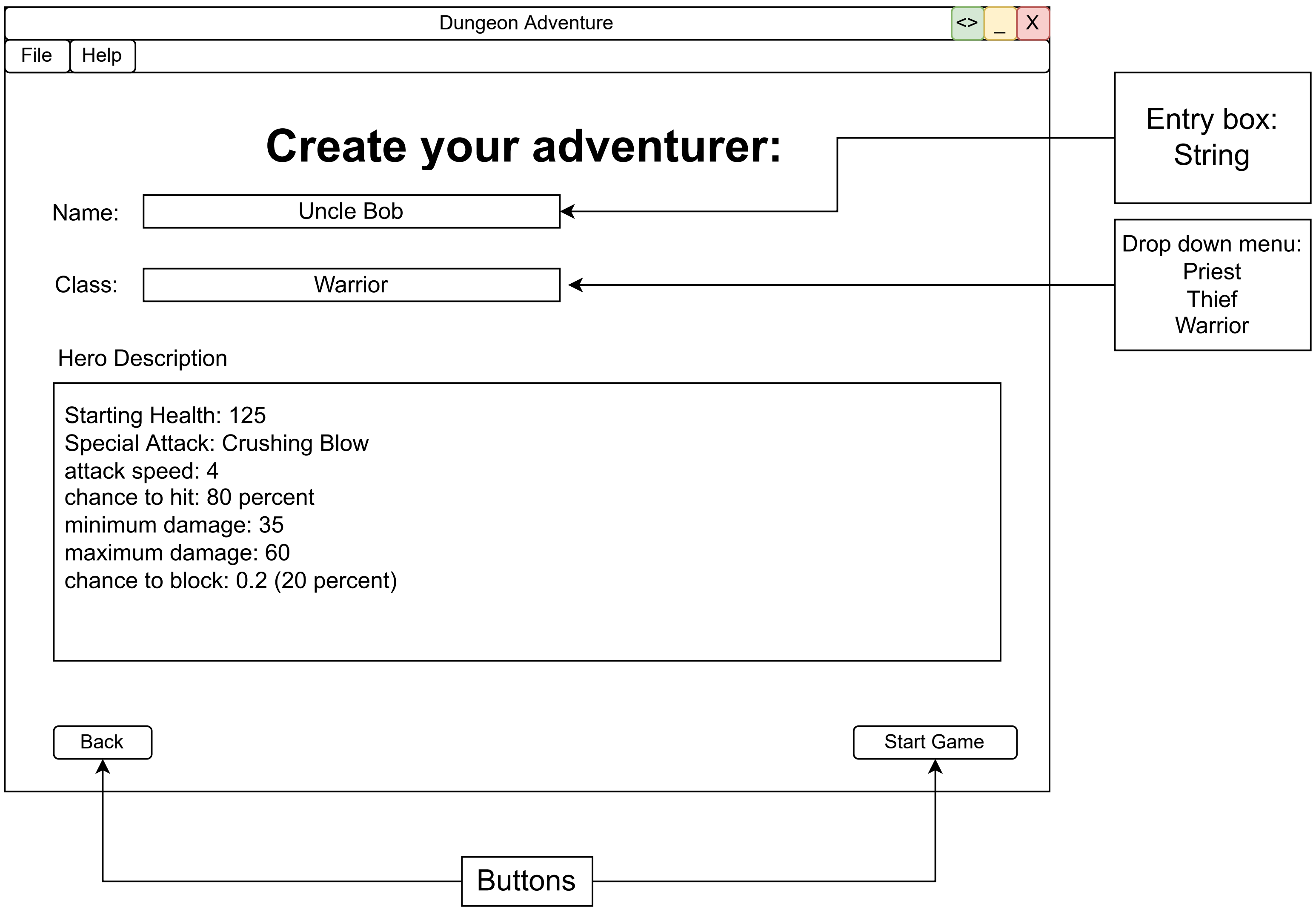
Fair and balanced. Choose for a decent challenge, and be wary of the dangers lurking inside the dungeon.

Dungeon Difficulty

User story: As a user, I want to play Dungeon Adventure with different difficulties.

The user will select from three different options.

- Difficulty Selection (drop-down menu): Easy, Medium, Hard, Inhumane
- Difficulty description: Current setting information
- Buttons: As described. Selecting these option will advance to another screen.



Create your adventurer:

Name: Uncle Bob

Class: Warrior

Hero Description

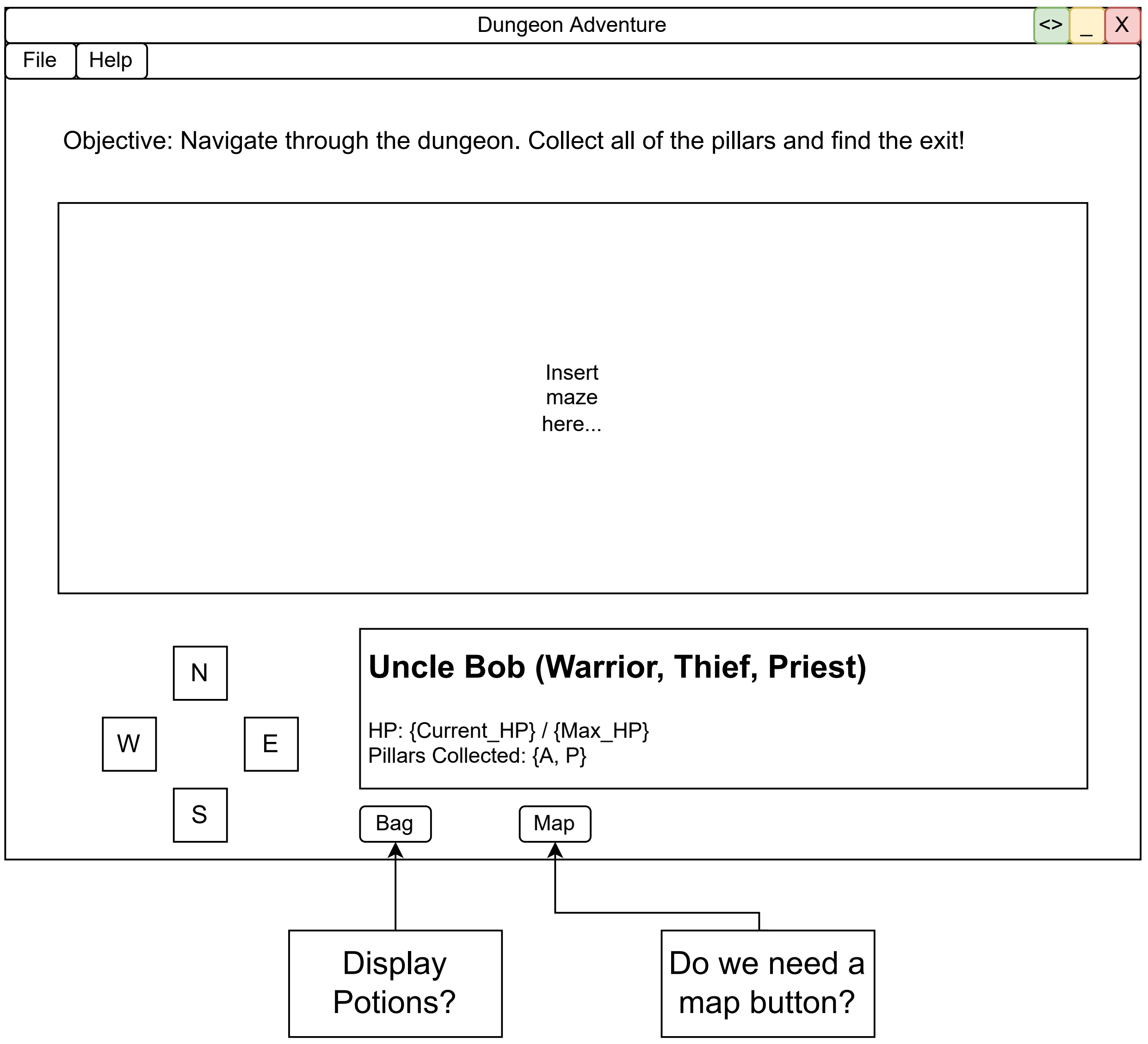
Starting Health: 125
Special Attack: Crushing Blow
attack speed: 4
chance to hit: 80 percent
minimum damage: 35
maximum damage: 60
chance to block: 0.2 (20 percent)

Adventurer Creation

User story: As a user, I want to select different adventurers and personalize them to my play style.

The user will run the program and select from three different options.

- Name: {User selected name, keyboard input}
- Class: (Drop down menu) - Warrior, Thief, Priest
- Buttons: Back - Go back to Dungeon; Start Game - Begin adventure



Objective: Navigate through the dungeon. Collect all of the pillars and find the exit!

Uncle Bob (Warrior, Thief, Priest)

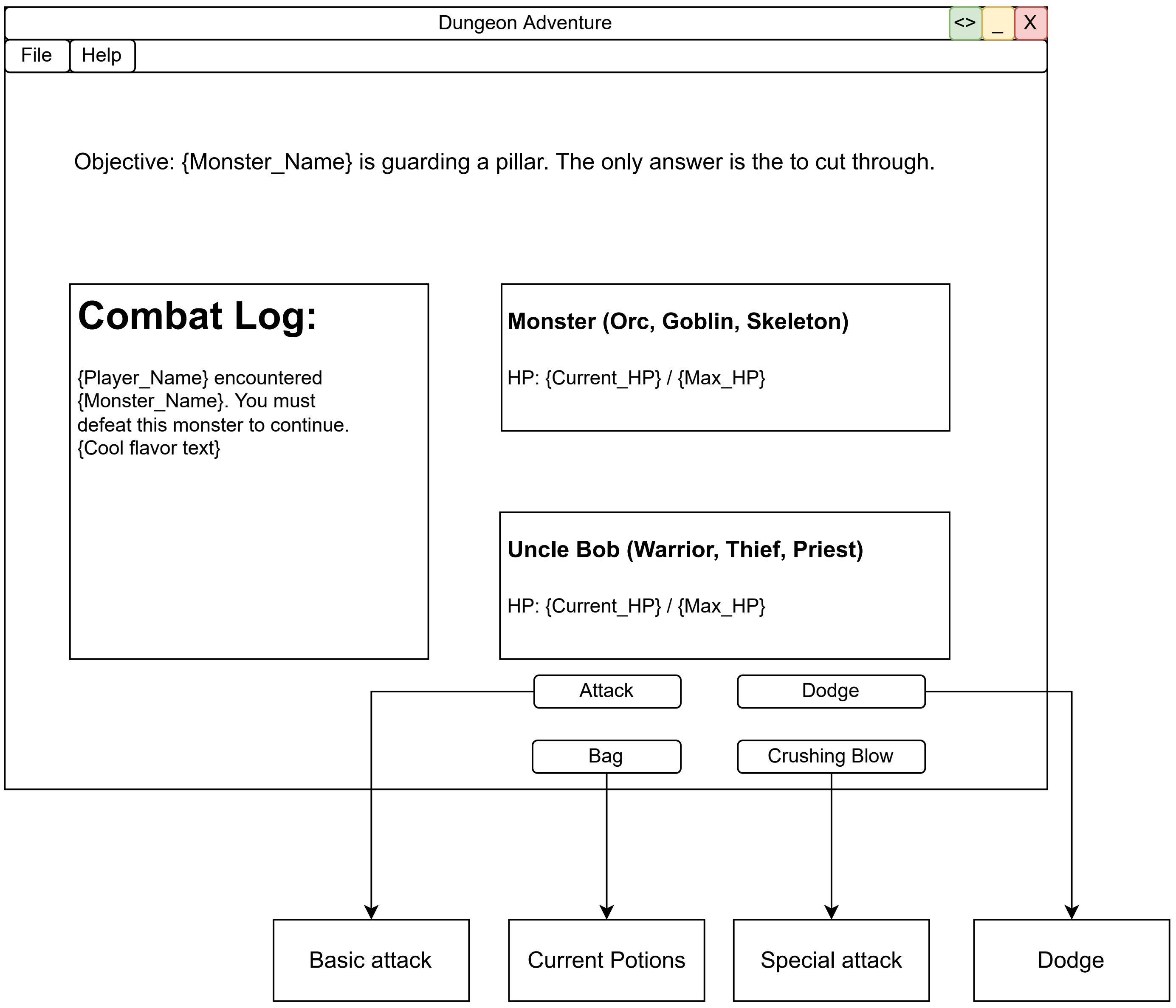
HP: {Current_HP} / {Max_HP}
Pillars Collected: {A, P}

Dungeon Crawler

User story: As a user, I want to select different adventurers and personalize them to my play style.

The user will run the program and select from three different options.

- Difficulty Selection (drop-down menu): Easy, Medium, Hard, Inhumane
- Difficulty description: Current setting information
- Buttons: As described. Selecting these option will advance to another screen.


Dungeon Adventure
<>
_
X
File
Help
Objective: {Monster_Name} is guarding a pillar. The only answer is the to cut through.
Combat Log:
{Player_Name} encountered {Monster_Name}. You must defeat this monster to continue. {Cool flavor text}
Monster (Orc, Goblin, Skeleton)
HP: {Current_HP} / {Max_HP}
Uncle Bob (Warrior, Thief, Priest)
HP: {Current_HP} / {Max_HP}
Attack
Dodge
Bag
Crushing Blow
Basic attack
Current Potions
Special attack
Dodge

Combat:

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.