# Functional specification: Dull Dave’s misadventure

## Purpose of Dull Dave’s misadventure:

To entertain the user through the story conveyed in text-based adventuring

## Sequence of story events for Dull Dave’s misadventure:

Introduce character – Dull Dave -

-you have no choice options to change anything about Dull Dave except for his middle name, which is taken as input and saved to the ‘Dulldave’ class as a variable attribute.

-Will be called within ‘\_\_str\_\_()’ to print full name of dull <var> dave

Set the scene –

-Describe to the user Dull Dave’s character

-life situations and current circumstances (preparing to go to the corner shop)

-Describe current room

-Describe the room he is in including items (items will be saved as attributes of room

-Provide 3 basic ‘chores’ to solve via 1 basic item collection each

-allow the user to pick up and use items required in selected rooms. Keywords such as ‘use’ or ‘go’ will allow the user to command dull dave. ‘grab’ will allow user to add item to their inventory for use in other rooms

-at least once: item pickup location ≠ location to be used

-min 5 items in starting room

Accident event –

-Dave accidentally smells a particularly pungent piece of purple mould whilst trying to clean inside a cabinet in his kitchen which knocks him out

-during this sequence the user has no input

-consider putting system wait commands in the program to help show passage of time whilst he regains consciousness

Post-event –

-Dull Dave wakes up and the environment is still the previous rooms but the rooms and items are changed as well as 2x newly created in-game npcs being created (created as instances of the npc class)

-All rooms now have altered descriptions and held items (& held item descriptions)

-All inventory items have now been modified **(would this mean it would be better to have parent and child item classes also?**

**This would mean that item descriptions could be held remotely (regardless of player circumstances) and referenced when needed from elsewhere, so it does not need to be ascertained whether they are being carried or their location etc.)**

Quest – Dull Dave then completes a small quest using interaction with the npcs and using new properties of items

## Required features:

Reference txt:

-Use Python file I/O to parse txt and compile into a list of strings.

This list will be iterated over and stored into a dictionary with a corresponding code-word as a key for each string value parsed from .txt

Functions:

-Main menu function

subfunctions:

-start new game (create a new game instance from ‘game’ class)

-load game (?) find or collect a saved path for an existing saved game

-Could be saved locally in a defined directory in the user’s computer and referenced upon a ‘load\_game command

-Would need to store all data of each class including events triggered and current room location (likely stored within ‘game’ class)

Objects:

-parent class for each room (x4 instances)

-child class for each room post-event (see story) (at least x4 instances)

**-idea**: **post-event (see story) a child class for each room is created and the item’s values**

- player class (1x instance per user at one time)

-initiating:

-takes input of Dull dave’s middle name

-stores all current game events that have been triggered. The events will be referenced from a room class

-stores current inventory of items

-has methods for what to do if:

- inventory is too full

-output for current inventory call by player

-output for

-npc class (x2 instances to be created from it)

-methods detail their interaction with the user by referencing player class’ current completed events and triggered items

-created **post-event** (see story above).

-if defeated or no longer relevant to story destroy instance of class

-game class -unique instance for each new game created by user

## logic flow for user command:

## 

Game end

Yes (gamestart() ‘exit’ == True)

Welcome player via string & new full name

Player.procedure(<room\_instance>):

No

Carry out command via Player.eventMethods() (this might mean new room argument).

Game complete?

Call Player.procedure(<room\_instance>):

gameStart() while loop --> “exit” == False:

Call Player.eventMethods(“commandString”).

If command cannot be carried out return None

yes

Is the command valid/ in a valid format?

Function will call a command validation sub-function.

Call Player.eventMethods(“contents”):

(prints a list of currently available items in the room)

Take keyboard input for a command

no

Call Player.eventMethods(“surroundings”):

Will print a description of the current room

Rtn = None. continue

while True:

Middle name entered by user, saved inside of Player.propertiesDict

gameStart() is called, game is introduced and string input is asked of player for middle name

[Game loads and all standard class instances are loaded