Hello and thank you for trying out this prototype.

So to go over all the features I'll start with the input system. I've used the InputSystem feature from unity, and added a custom script called "PlayerInputManager" which in hind sight, was a bad name since the InputSystem has a component with the same name in a different name space. The PlayerInputManager(mine that is) initializes the input for the user, then sends that input to components that derive from input interfaces. For example there is an IMoveInputReceiver which as its name describes, receives move input. There is also an IactionInputReciever which receives action inputs. The reason for separating these is that the designer may not want to have any actions available to the user, or they may want to remove movement controls. With this system there are very little strict script requirements, making breaking the project a little difficult.

The next component I'll go over is the OutfitManager. This component uses AnimatorOverrideControllers to swap the characters animator at run time. This could be alternatively just swapping out the sprites and creating a custom animation system, or enabling and disabling objects, but the reason I've chosen to do it this way is that it's only swapping sprites which is not very performance impacting. Another reason is time, I could have created separate animators for each piece of clothing, but instead for the sake of time just changed the entire outfit. However if given a week I would do something more customize-able. The OutfitManager also allows the designer to create different NPC's all from the same prefab, but just swapping the start outfit index, which is very handy.

The InteractionManager is something I think is fairly cool. It checks every 10 frames for interaction objects within a range set by the designer. The interaction objects are defined using an interface "IInteractable" which allows for abstraction of the entire interaction system. This is cool because you could make objects that could be picked up by simply deriving it from IInteractable and then simply adding it to the inventory. The 10th frame check is only for the sake of the interaction prompt. When you get within range of an interaction object you need to know that, so that's why it's checking every 10th frame for an interaction object. However when the interaction button is pressed it checks then, so it's not really needed to check every 10th frame if you don't want an interaction prompt.

The shop system is very simple. It basically extends the OutfitManager, to allow for equipping new outfits. I've created a scriptable object for purchasable outfits, to make adding new outfits a little easier. The shop on start then checks through each outfit in the shop and compares it to the starting outfit the player has. That outfit is then considered purchased.
Each scriptable object outfit can be given it's own price, it's own icon, and it's own description, which will all be displayed to the player through the shop display interface.
The money system is fairly simple too, the designer sets the starting money, then when something wants to spend money they ask if the player can afford it.

The dialogue system uses a few separate components together. Each dialogue menu is considered a "DialogueChunk" and you can add buttons and text to each element right from the character dialogue component. Once the character is interacted with the dialogue system opens the dialogue display for the user. Every part of this system uses the UnityEvent system to make it much more flexible for the designer.

Finally I spent the rest of the time allocated to make the demo presentable. There is a layer mask

system so that you appear behind or in front of items depending on where you stand. There is an audioclip player which plays the bird sound effects with a random and a constant wait combined to make for a very customize-able sound effect system. The scene is laid out with lodes of items to make it feel a little bit more real. The footstep sound effect system is done in script so that the designer doesn't have to add the sound effect sounds to the animator.

I think I've done really well considering I made this in a little under 2 days. I could do better in a few areas, but honestly who couldn't? I think I've done well because this project can easily be expanded apon. There are very few features I'm unhappy with. This project is flexible enough to be able to take any particular system and plug it into another project with out through many if any errors, which is the way I believe systems should be programmed.