The game was designed to function with as little hard referencing as possible and therefore uses events for communication between components. Components themselves I tried to keep as small as possible, and in charge of one thing only wherever possible, in order to make debugging as easy and painless as possible. Most of the system is driven by components on the Player GameObject as well as the ShopController on the Shop GameObject. The UI was kept separate from data held in these objects and is only used as a view layer.

During the work done on the test I was trying to build it up one functionality at a time. I started with basic movement and interaction with the shopkeeper, then adding shop UI and buying and selling items. Afterwards the inventory and finally the equipping of items was added thus completing the required functionality. This way the progress was more easily measured as each new part was worked on until it was complete instead of doing a small part of each part of the code.

I think the code is sufficiently decoupled and relatively generalised but I do think it could be written in a more modular way. The goal was to try and keep the components modular but due to time constraints a more direct approach was used when needed. I am happy with with routing of function calls in some areas like UI to Shop and then player Inventory instead of a direct call. This kind of separation decreases the chance of bugs at a cost of somewhat larger components. In some cases there was a direct access to fields instead of using properties, but I have tried to keep it to a minimum.