Functional Requirements

* Individual should be able to create savefile to save their progress and start their adventure.
* Individual should be able to control movement and use the features given to character (skill reading, player status, settings, inventory, map).
* The user are able to interact with NPC to help them in their adventures.
* The user are able to interact with environment captivating player with difficulties and achivements.
* The user are able to view item and but them (single item can be bought in bulk).
* User are able to enter turn-based battle where there character interacts with enemies in battle.
* User are able to use their items, skills against enemy.

Non functional requirement:

* Scalability: system should be able to handle increasing number of player party and items.
* Compatibility: The system should work seamlessly in windows.
* Performance: system should work seamlessly in atleast thirty frame per second.
* Reliability: System should be reliable without any majot breakdown

Design Constraints:

* User interface should be intuitive and user-friendly.
* Designed to accommodate potential increase and user load.
* It should fully support windows.
* It should be compatible with at least windows 7, 10 and 11.

Implementation Constraints:

* Using unity engine which uses C# animations, front-end and back-end will be designed.
* System will use SQLite database for storage and retrieval of information, including customer detail and transaction records.
* System will be build in unity engine and which can then be run in .exe file.
* System should maintain the standard of early 20’s and late 90’s animation.
* System should be compatible with hardware components, including processor, memory and storage found in standard computing device.
* Integration with server side language and database management system.
* Compatibility with chosen operating system.
* Response time against user interactive should be smooth and responsive.

Other Non Functional Requirements:

* Response time for critical user interaction.
* System should work seamlessly without compromising its performance and feature.
* System should be highly reliable and hsould have fault tolerance and disaster recovery.
* Ensure accuracy and consistency throughout the system.
* Provide comprehensive document for user.
* Implement access controls and encryption and other security measures to safeguard exploitation.
* Consider environmental impact of system, such as energy consumption and resoure utilization for sustainability if possible.

will the following can be included in dfd: register savefile character feature

Yes, **"register savefile"** and **"character feature"** can be included in a **Data Flow Diagram (DFD)**, but they would be represented as part of the system's **processes** or **data stores** depending on the level of detail and how the system operates. Let’s explore how these components would fit into a DFD:

**1. Register Savefile:**

* **Process**: In the DFD, the "register savefile" would typically be represented as a **process** that handles saving the user's game progress. This process would interact with the **user** (as an external entity), receiving data such as the user’s progress, character status, and inventory. It would then store this information in a **data store** (e.g., **Savefile Database**).
* **Data Flow**:
  + The **user** provides data (e.g., current game progress) to the "register savefile" process.
  + The process then sends the data to a **Savefile Data Store** where it is stored for future use.
* **DFD Representation**:
  + **External Entity**: User
  + **Process**: Register Savefile
  + **Data Store**: Savefile Data Store
  + **Data Flow**: Data (progress, inventory, etc.) flows from User to Process, and from Process to Data Store.

**2. Character Feature:**

* **Process**: The "character feature" (e.g., skills, player status, inventory, etc.) would also be a **process** or set of processes in the DFD. These processes would involve the user interacting with different aspects of their character (like **viewing skills**, **checking player status**, or **accessing the inventory**).
* **Data Flow**:
  + The **user** provides input (e.g., requesting to view the inventory or player status).
  + The system processes this request and provides the relevant data (such as showing the inventory or character stats) from an appropriate **data store** (like **Character Data Store** or **Inventory Data Store**).
* **DFD Representation**:
  + **External Entity**: User
  + **Process**: View Character Feature (this could be multiple processes like "View Skills", "View Inventory", etc.)
  + **Data Store**: Character Data Store, Inventory Data Store (depending on the feature)
  + **Data Flow**: Data (character stats, inventory items, etc.) flows from the Data Store to the Process and then to the User.

**How to Represent These in a DFD:**

1. **Level 0 DFD (Context Diagram)**:
   * This level is a high-level representation of the system. Here, both **register savefile** and **character feature** might be included as part of the **central system process** that communicates with external entities (e.g., the User).
   * The **User** interacts with the system by registering a savefile and accessing character features.
2. **Level 1 or Deeper DFD**:
   * Here, these concepts would be broken down into smaller, more detailed processes.
   * For example, the **Register Savefile** process would be split into several subprocesses like **Save Character Data**, **Save Inventory**, **Save Progress**, etc.
   * The **Character Feature** could be represented by multiple processes like **View Inventory**, **View Player Stats**, **Use Skills**, etc., with data flows from and to relevant data stores (e.g., **Character Database**, **Inventory Database**).

**Conclusion:**

Both **"register savefile"** and **"character feature"** can definitely be included in a DFD as **processes**. The **register savefile** would typically be related to saving the player’s progress and interacting with a data store, while **character features** would involve multiple processes related to displaying or interacting with the character’s data. These processes can be represented with their corresponding data flows to and from **external entities** and **data stores**.

**You said:**

what about npc interaction, environment interaction, shop purchase, turn base battle, setting manager

**ChatGPT said:**

ChatGPT

The components you've mentioned — **NPC interaction**, **environment interaction**, **shop purchase**, **turn-based battle**, and **setting manager** — can all be represented in a **Data Flow Diagram (DFD)** as processes, data flows, and data stores. Here's how each of these elements might be mapped in a DFD:

**1. NPC Interaction:**

* **Process**: In a DFD, **NPC interaction** would be a **process** where the user interacts with Non-Player Characters (NPCs) to receive quests, information, or other assistance.
* **Data Flow**: The **user** sends an interaction request (e.g., choosing to talk to an NPC) to the **NPC Interaction Process**, which retrieves relevant information or responses from an **NPC Data Store** or another system resource.
* **DFD Representation**:
  + **External Entity**: User
  + **Process**: NPC Interaction
  + **Data Store**: NPC Data Store (stores NPC dialogue, quests, or actions)
  + **Data Flow**: Interaction data flows between the User and NPC Interaction process, and the NPC Interaction process accesses data from the NPC Data Store.

**2. Environment Interaction:**

* **Process**: The **environment interaction** could be a **process** where the player interacts with the game world, such as solving puzzles, finding items, or engaging in environment-based challenges (e.g., opening a chest, triggering traps, etc.).
* **Data Flow**: The **user** initiates an action (e.g., interacting with an object or location in the environment). The process retrieves environmental data (e.g., the object’s status, puzzle state) from a related **environmental data store**.
* **DFD Representation**:
  + **External Entity**: User
  + **Process**: Environment Interaction
  + **Data Store**: Environmental Data Store (stores information on interactive objects, puzzle states, etc.)
  + **Data Flow**: Data related to the environment (item information, puzzle state) flows between the User and the Environment Interaction process, and from the process to the Environmental Data Store.

**3. Shop Purchase:**

* **Process**: **Shop purchase** would be a process that handles the user purchasing items from a store in the game. The user could view available items and choose to buy them.
* **Data Flow**: The **user** sends a purchase request (e.g., selecting an item to buy) to the **shop purchase process**, which interacts with an **inventory or shop data store** to check the availability of the item, deduct the cost, and update the player’s inventory and currency balance.
* **DFD Representation**:
  + **External Entity**: User
  + **Process**: Shop Purchase
  + **Data Store**: Inventory Data Store (stores available items for sale)
  + **Data Store**: Currency Data Store (tracks player’s available money)
  + **Data Flow**: Data (selected items, currency) flows between the User, Shop Purchase process, Inventory Data Store, and Currency Data Store.

**4. Turn-Based Battle:**

* **Process**: **Turn-based battle** would be a central **process** where the player engages in a combat scenario with enemies. This process would handle turns, combat decisions, and the use of skills, items, or actions.
* **Data Flow**: The **user** would interact with the battle process by making decisions (e.g., attack, defend, use skill), which in turn modifies the state of the battle (e.g., health points, skill cooldowns, item usage) by interacting with the **battle data store** and the **enemy data store**.
* **DFD Representation**:
  + **External Entity**: User
  + **Process**: Turn-Based Battle
  + **Data Store**: Battle Data Store (stores current battle state, player/enemy stats, etc.)
  + **Data Store**: Enemy Data Store (stores enemy stats, skills, etc.)
  + **Data Flow**: Data (player actions, battle state) flows between the User, Turn-Based Battle process, Battle Data Store, and Enemy Data Store.

**5. Setting Manager:**

* **Process**: The **setting manager** is a process that manages the system’s settings, such as sound, graphics, game preferences, and other configurations.
* **Data Flow**: The **user** interacts with the setting manager to update or view settings. The process retrieves and updates user preferences stored in a **settings data store**.
* **DFD Representation**:
  + **External Entity**: User
  + **Process**: Setting Manager
  + **Data Store**: Settings Data Store (stores user preferences for settings such as volume, resolution, etc.)
  + **Data Flow**: Data (user preferences) flows between the User and the Setting Manager process, and the process interacts with the Settings Data Store.

**Summary of DFD Representation:**

* **External Entities**: User (and potentially NPCs or other external systems like a shop or battle system).
* **Processes**:
  + **NPC Interaction**
  + **Environment Interaction**
  + **Shop Purchase**
  + **Turn-Based Battle**
  + **Setting Manager**
* **Data Stores**:
  + **NPC Data Store**
  + **Environmental Data Store**
  + **Inventory Data Store**
  + **Currency Data Store**
  + **Battle Data Store**
  + **Enemy Data Store**
  + **Settings Data Store**

**Example DFD Flow:**

1. **User** interacts with the **NPC Interaction process**, which retrieves NPC dialogue or quest data from the **NPC Data Store**.
2. The **User** purchases an item through the **Shop Purchase process**, which checks availability in the **Inventory Data Store** and deducts currency from the **Currency Data Store**.
3. During a **Turn-Based Battle**, the **User** provides commands (attack, use item) to the **Turn-Based Battle process**, which updates the **Battle Data Store** and interacts with the **Enemy Data Store**.

**Levels of DFD:**

* At a **Level 0 DFD (Context Diagram)**, these processes would be high-level representations, with external entities (e.g., User) interacting with the system's central process.
* At **Level 1 DFD** and beyond, you would break down these processes into more specific subprocesses (e.g., "Battle Command", "Item Purchase", "View NPC Dialogue") and show how data flows between them.

By representing these components in a DFD, you can clearly visualize how the data moves between different processes, entities, and stores in your system.