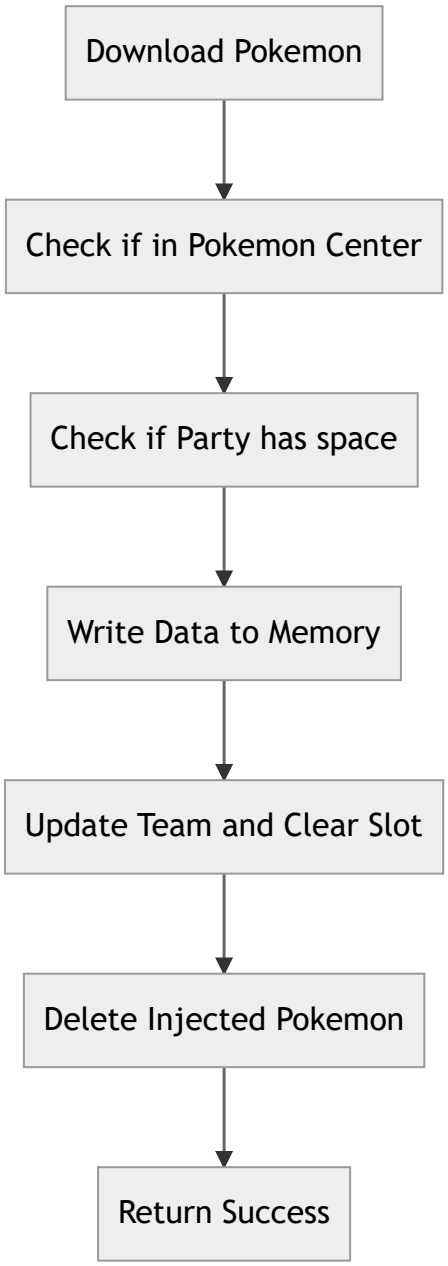
Game Specific DMA Logic

Download & Inject Pokémon Logic

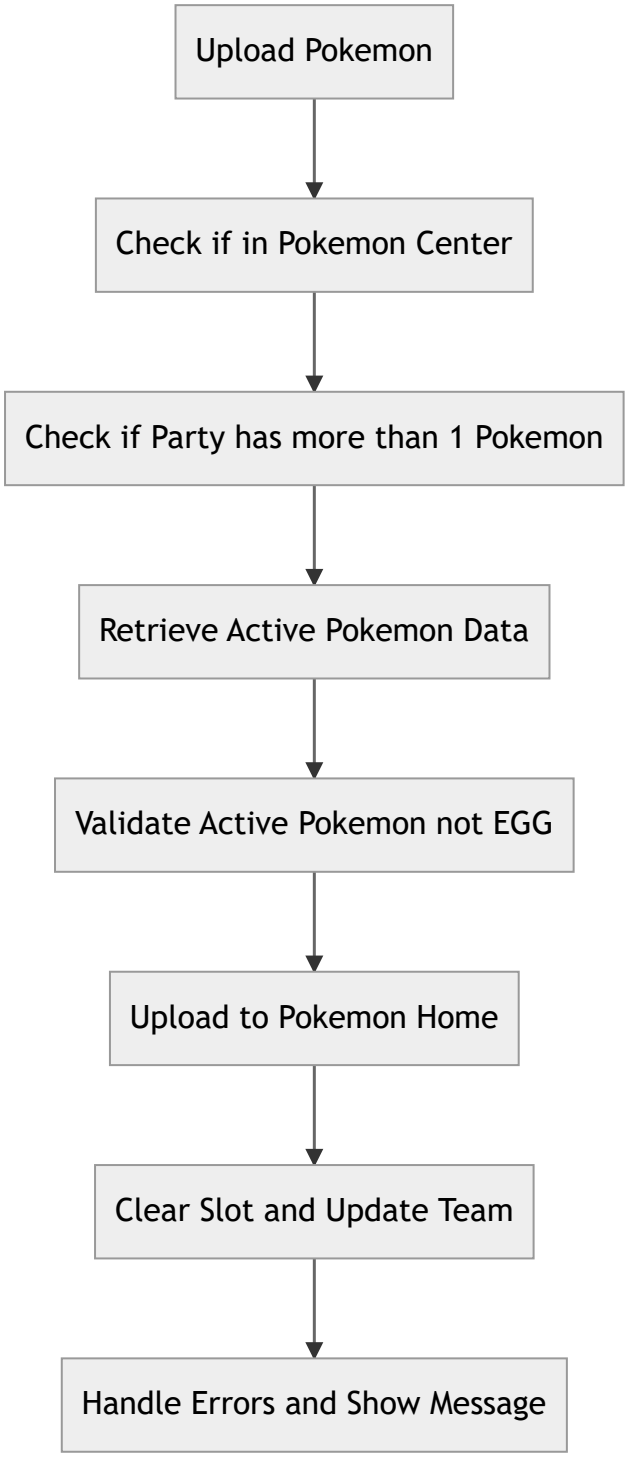


Summary:
This function injects a new Pokémon into the player's party. It first checks if the player is in a Pokémon Center and whether there's space in the party. If both conditions are met, it writes the new Pokémon's data into memory and updates the team accordingly.

- **Validation:** Ensures player is in Pokémon Center and there's space in the party.
- **Injecting Data:** Writes the new Pokémon's data into memory.

- **Cleanup:** Updates the team and deletes the injected Pokémon from its original source (MySQL DB).

Upload & Shift Pokémon Logic



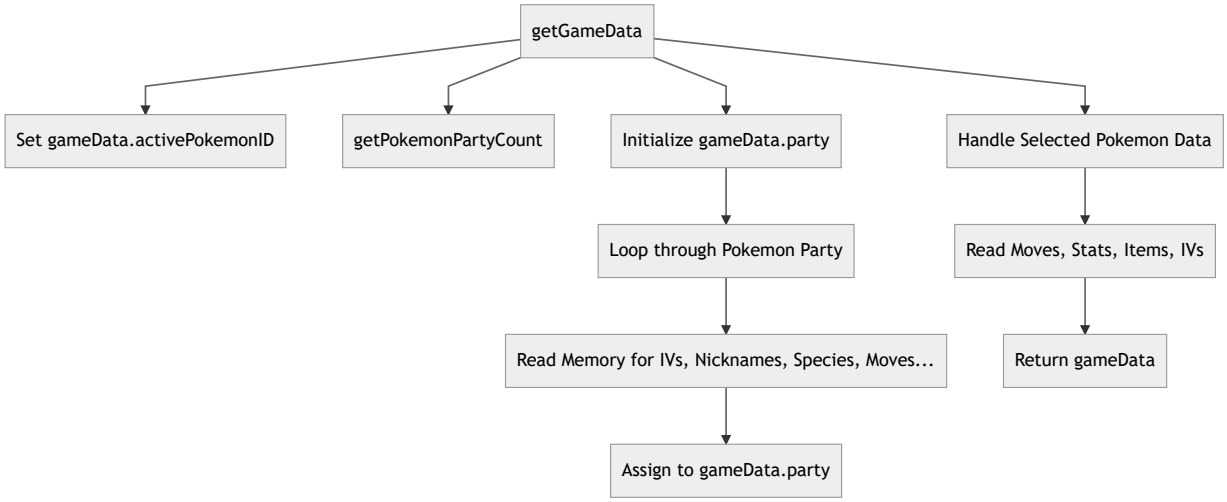
Summary:

The `uploadPokemonFromMemory` function uploads a Pokémon from the player's active party to the Pokémon Home system. It ensures that the player is in a Pokémon Center

and that there are at least two Pokémon in the party. It then uploads the active Pokémon, clears the slot, and updates the team.

- **Validation:** Ensures player is in Pokémon Center and has more than one Pokémon.
- **Data Retrieval:** Retrieves active Pokémon data and checks it's not an "EGG."
- **Upload & Update:** Uploads to Pokémon Home, clears slot, and updates the team.

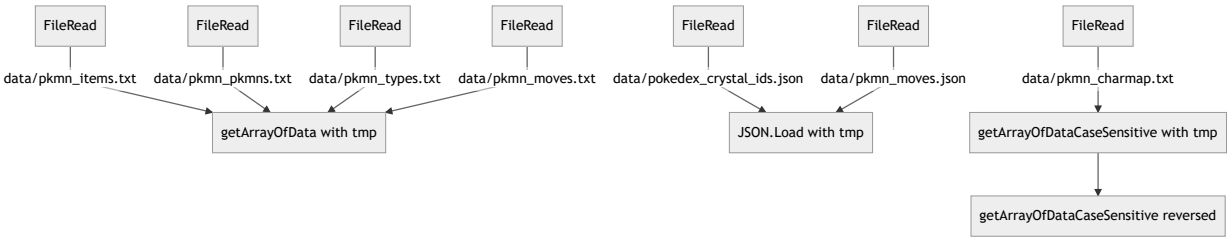
Get Game Data Logic



Summary:
The `getGameData` function retrieves data for the active Pokémon, including species, nickname, stats, IVs (individual values), moves, and items. It collects information from memory addresses and formats it into a structured object (`gameData`), which is then returned.

- **Active Pokémon:** Retrieves the active Pokémon's data (`activePokemonID`).
- **Party Info:** Loops through the party and gets data for each Pokémon.
- **Assigns Data:** Fills `gameData` with the Pokémon's details (species, stats, IVs, etc.).

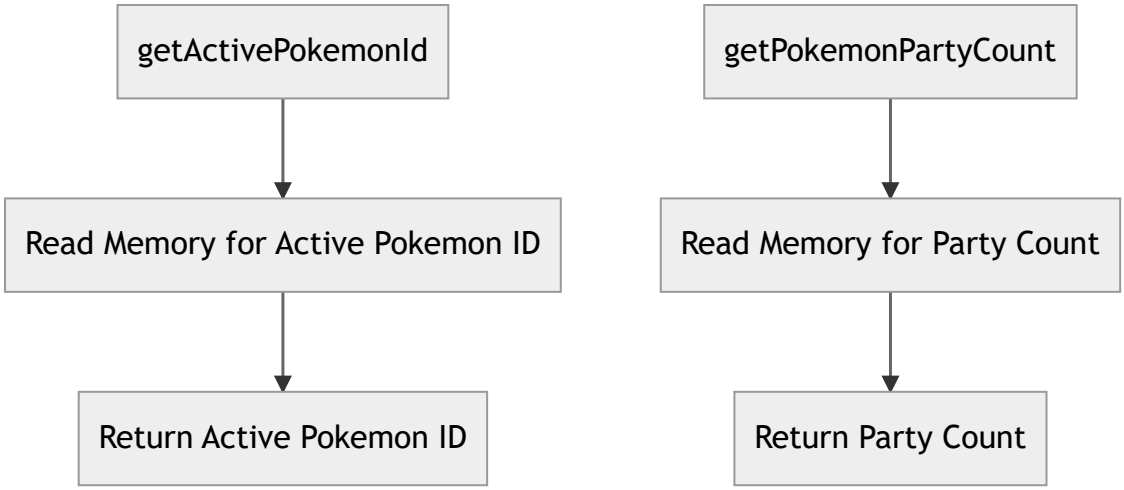
Common Enums & Prefetched Data Load



Summary:
This flow represents how various data files (like Pokémon items, Pokémon species, types, moves, etc.) are read from different file paths. Each `FileRead` node corresponds to a file being read and its contents processed by functions like `getArrayOfData` or `JSON.Load` to populate arrays or objects. The `chars` array is also processed with case-sensitive data reading.

- **File Reading:** Data files are read from paths (`pkmn_items.txt` , `pkmn_pkmns.txt` , etc.).
- **Data Processing:** Functions like `getArrayOfData` and `JSON.Load` convert raw text into structured arrays or objects for further use.

Helper functions

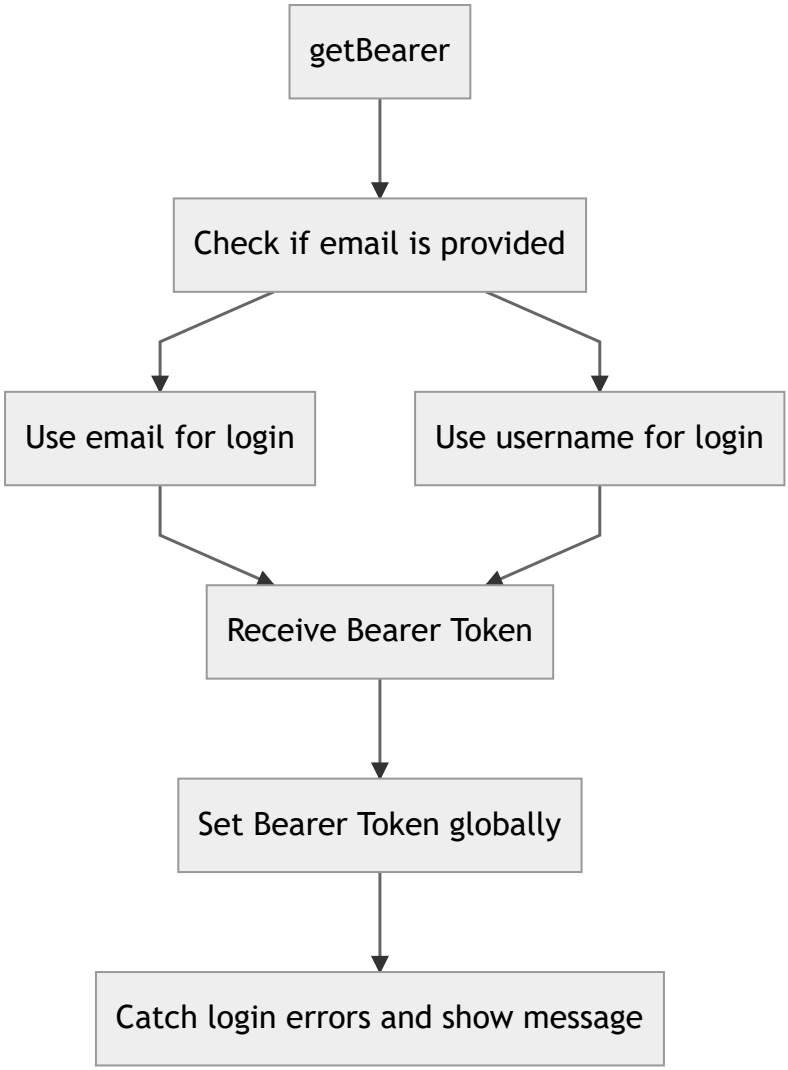


Summary:
These are helper functions to retrieve specific data about the player's current Pokémon. `getActivePokemonId` fetches the ID of the active Pokémon, while `getPokemonPartyCount` fetches the number of Pokémon in the player's party. Both functions read data from memory addresses and return the respective values.

- **Active Pokémon ID:** Retrieves the ID of the active Pokémon.
- **Party Count:** Retrieves the number of Pokémon in the party.

Pokémon Home API Integration

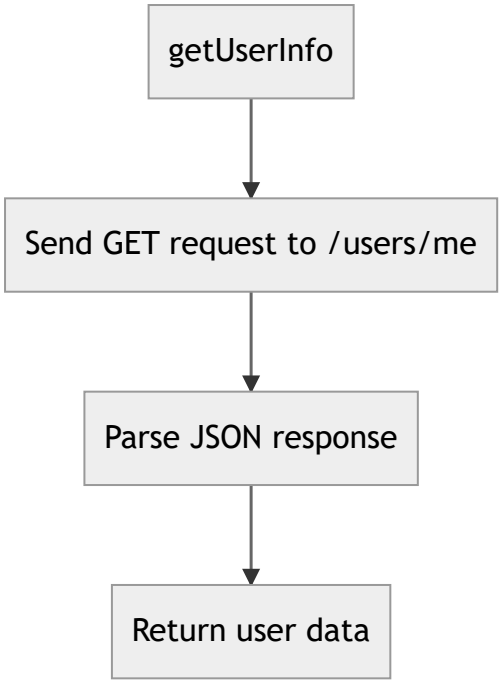
Authentication Flow



The `getBearer` function handles user authentication by logging in with either a username or email and password. Based on the user input, the function determines whether to use an email or username and sends a `POST` request to Pokémon Home for a bearer token, which is used for future requests.

- **Authentication:** Differentiates between email and username login.
- **Bearer Token:** Sends credentials to the API and receives a token.
- **Error Handling:** If authentication fails, it shows an error message.

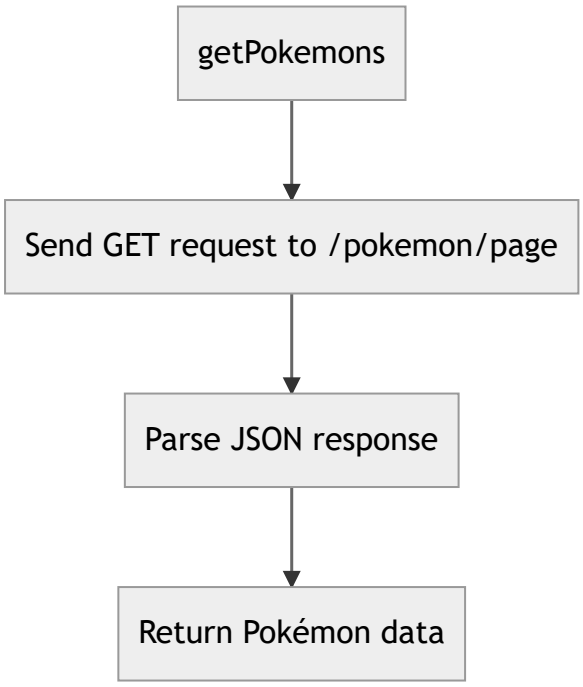
User Info Retrieval



The `getUserInfo` function sends a `GET` request to the `/users/me` endpoint to retrieve user details, which are then parsed from JSON and returned.

- **GET Request:** Makes a `GET` request to fetch user details.
- **Parsing JSON:** Parses the response and returns the user information.

Fetch Pokemon Data



The `getPokemons` function sends a `GET` request to retrieve a list of Pokémon from the Pokémon Home server. The page number is used as a parameter, and the function returns the parsed JSON response.

- **GET Request:** Requests Pokémon data from the server.
- **Return Data:** Parses and returns the list of Pokémon.