```
{
    "user_id": int,    # Auto-incrementing unique identifier
    "name": str,       # User's full name
    "email": str,      # User's email (must be unique and contain '@')
    "age": int         # User's age (must be > 0)
}
```

Global Variables
- **`users`**: List storing all user records
- **`next_user_id`**: Counter for generating unique user IDs (starts at 1)

Core Components

Utility Functions

`is_valid_email(email, users, current_user_id=None)`
Validates email format and checks for duplicates.

Parameters:
- `email`: Email string to validate
- `users`: List of existing users
- `current_user_id`: Optional. Used during updates to exclude current user from duplicate check

Returns: `True` if valid, `False` otherwise

Validation Rules:
- Must contain '@' symbol
- Cannot be empty or whitespace only
- Must be unique across all users (except when updating current user)

`get_valid_positive_int(prompt)`
Prompts user for input and validates it's a positive integer.

Parameters:
- `prompt`: String to display to user

Returns:
- Positive integer if valid
- `None` if invalid

Validation Rules:
- Must be numeric
- Must be greater than 0

`find_user_by_id(users, user_id)`
Searches for a user by their ID.

Parameters:
- `users`: List of user records
- `user_id`: ID to search for

Returns:
- User dictionary if found
- `None` if not found

`display_user(user)`
Formats and prints user information in a readable format.

Parameters:
- `user`: User dictionary to display

Output Example:
```
----------------------------------------
ID   : 1
Name : John Doe
Email: john@example.com
Age  : 30
----------------------------------------
```

CRUD Operations

`add_user(users)`
Creates a new user record.

Process:
1. Prompts for name (required, non-empty)
2. Prompts for email (validated for format and uniqueness)
3. Prompts for age (must be positive integer)
4. Creates user with auto-generated ID
5. Appends to users list
6. Increments `next_user_id`

Error Handling:
- Empty name → "Error: Name cannot be empty"

- Invalid/duplicate email → "Error: Invalid or duplicate email"
- Invalid age → "Error: Age must be a number greater than 0"

`view_all_users(users)`
Displays all users in the system.

Process:
1. Checks if users list is empty
2. Iterates through all users
3. Calls `display_user()` for each record

Edge Cases:
- Empty list → "No users found"

`search_user_by_id(users)`
Finds and displays a specific user by ID.

Process:
1. Prompts for user ID
2. Validates ID is positive integer
3. Searches for user
4. Displays user if found

Error Handling:
- No users exist → "No users found"
- Invalid ID format → "Invalid ID"
- User not found → "User not found"

`update_user(users)`
Updates an existing user's information.

Process:
1. Prompts for user ID to update
2. Finds user by ID
3. For each field (name, email, age):
   - Shows current value
   - Prompts for new value
   - Pressing Enter keeps existing value
4. Validates new email and age if provided
5. Updates user record

Features:

- Partial updates supported (can update individual fields)
- Current values displayed as defaults
- Email uniqueness validated (excluding current user)

Error Handling:
- No users exist → "No users found"
- Invalid ID → "Invalid ID"
- User not found → "User not found"
- Invalid/duplicate email → "Error: Invalid or duplicate email"
- Invalid age → "Error: Age must be a number greater than 0"

`delete_user(users)`
Removes a user from the system.

Process:
1. Prompts for user ID to delete
2. Finds user by ID
3. Asks for confirmation (y/n)
4. Removes user if confirmed

Safety Features:
- Confirmation required before deletion
- Can cancel operation by entering anything other than 'y'

Error Handling:
- No users exist → "No users found"
- Invalid ID → "Invalid ID"
- User not found → "User not found"
- Cancelled → "Deletion cancelled"

Menu System

`show_menu()`
Displays the main menu with available operations.

Menu Options:
1. Add User
2. View All Users
3. Search User by ID
4. Update User
5. Delete User
6. Exit

`main()`
Main program loop that handles user interaction.

Process:
1. Displays menu
2. Waits for user input
3. Routes to appropriate function based on choice
4. Loops until user selects "Exit" (option 6)

Error Handling:
- Invalid menu choice → "Invalid choice"

Usage Examples

Adding a User
```
Choose an option: 1
Enter name: Alice Smith
Enter email: alice@company.com
Enter age: 28
User added successfully
```

Updating a User
```
Choose an option: 4
Enter user ID to update: 1
Press Enter to keep existing value
Name [Alice Smith]: Alice Johnson
Email [alice@company.com]:
Age [28]: 29
User updated successfully
```

Deleting a User
```
Choose an option: 5
Enter user ID to delete: 1
Are you sure you want to delete this user? (y/n): y
User deleted successfully
```