Project Description: Rental Management Application

Project Overview:

A Rental Management Application is a comprehensive software solution developed to facilitate the management of rental properties, ensuring that property owners, property managers, and tenants can effectively and efficiently handle all aspects of the rental process. The application serves as a centralized platform for various tasks, including property listing, tenant onboarding, lease management, maintenance requests, and financial transactions.

Key Features and Functionalities:

1. Property Listings:
    - Property owners can create detailed listings of their rental properties, including property descriptions, rental rates, and high-quality photos.
    - Property managers can easily organize and promote available properties for rent.
2. Tenant Management:
    - Property managers can manage tenant information, including tenant applications, references, and lease agreements.
    - Tenants can submit applications, review lease terms, and communicate with property managers.
3. Lease Tracking:
    - The application helps property managers and property owners keep track of lease agreements, including lease terms, renewal dates, and rent payments.
4. Maintenance Requests:
    - Tenants can submit maintenance requests and track their status.
    - Property managers can assign and manage maintenance tasks efficiently.
5. Financial Management:
    - The application enables property owners and managers to handle financial transactions, including rent collection.
6. Communication and Messaging:
    - The application includes communication tools for property managers, property owners, and tenants to exchange messages and notifications.
7. Document Management:
    - Users can store and manage essential documents such as lease agreements, rental applications, and property inspection reports.
8. Reporting and Analytics:
    - The application provides insights into property performance, financial data, occupancy rates, and maintenance history through reports and analytics.
9. Security and Data Protection:
    - Robust security measures ensure the safety of sensitive data, including personal and financial information.

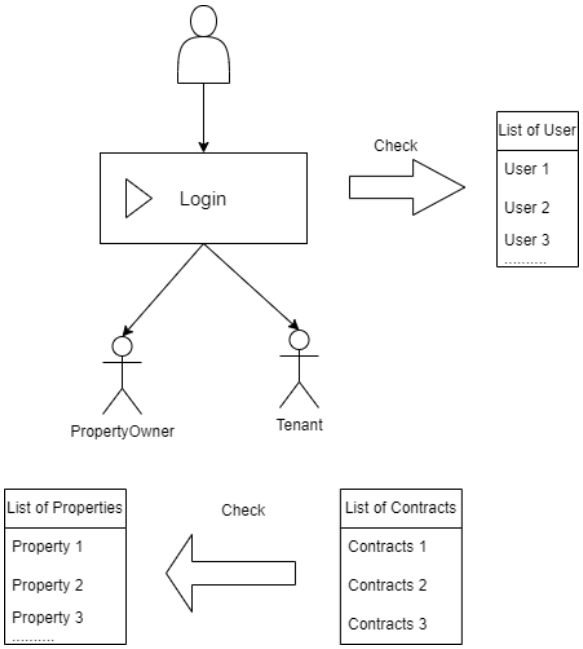
Assume that our Rental Management Application includes some of the essential classes and relationships as follows:

Classes:

1. **User**
   - **Attributes**: UserID, Password, Email, UserType
   - **Operations:** CreateUser**,** ResetPassword()
2. **RegisteredUser**
   - **Attributes**: listOf User
   - **Methods**: addUser(), removeUser()
3. **Property:**
   - **Attributes**: PropertyID, Description, **PropertyOwnerId**
4. **PropertyOwner**
   - **Attributes**: **User, listOfProperties**
   - **Methods**: addProperty(), removeProperty ()
5. **Tenant**
   - **Attributes**: **User, listOfContract**
   - **Methods**: requestCreateRentalContract(), requestTerminateRentalContract()
6. **RentalContract/ Lease:**
   - **Attributes**: ContractID, **PropertyID**, **TenantID**, RentAmount
   - **Methods**: CreateContract(), TerminateContract()

Relationships:

1. PropertyOwner can own multiple properties.
2. Tenants may have multiple rental contracts.
3. Property can belong to multiple rental contracts.



Practice Assignment 2: Version control and Collaboration

For the sake of understanding Github and team collaboration, some part of this assignment will require you to work in groups. The part where you must collab with your colleagues will be colored red.

I. Version control (20%)

1. Add one "**User story**" to your Trello which includes 3 technical tasks: "Create **GitHub repository;** Publish **"Local repository**" to GitHub; Add GitHub link;
2. Create a "**GitHub repository**" as your main (root) directory.
3. Publish your "**Local repository**" to GitHub.
4. Add your group members as your Collaborators (in case you want your repository to remain "private"). Skip this step if you leave them as "public".
5. Add your GitHub repository link to "**User story**".

II. Collaboration (40%)

1. Create 5 separate "**User story**" for 5 essential classes described above.
2. Implement "**User**" and "**Property"** class.
3. Other member creates separate Branch from the main repository.
4. Each member chose to implement one of the remaining class.
5. Other member creates a pull request to the repository owner.
6. Repository owners merge each branch into their main.

II. Make sure that your board mirrors your current progress. (40%)