Requirements Engineering

PropertySys – A Property Rental System

Submitted By: Conor O’Brien (T00193238)

Computing with Multi Media

Date Submitted: 12/12/2017

**Table of Contents**

[**1.** **Introduction** 4](#_Toc508200802)

[**2.** **Functional Components** 5](#_Toc508200803)

[**3.** **User Requirements** 6](#_Toc508200804)

[**3.1.** **PropertySys will perform Ownership management** 6](#_Toc508200805)

[**3.2.** **PropertySys will perform Property management** 6](#_Toc508200806)

[**3.3.** **PropertySys will perform Tenant’s management** 6](#_Toc508200807)

[**3.4.** **PropertySys will perform Administration analysis** 6](#_Toc508200808)

[**4.** **System Requirements** 7](#_Toc508200809)

[**4.1.** **System Level Use Case Diagram** 8](#_Toc508200810)

[**4.2.** **Manage Owners** 8](#_Toc508200811)

[**4.2.1.** **Add Owner** 8](#_Toc508200812)

[**4.2.2.** **Update Owner** 12](#_Toc508200813)

[**4.2.3.** **Remove Owner** 15](#_Toc508200814)

[**4.2.4.** **Query Owner** 17](#_Toc508200815)

[**4.3.** **Manage Property** 19](#_Toc508200816)

[**4.3.1.** **Add Properties** 19](#_Toc508200817)

[**4.3.2.** **Update Property** 22](#_Toc508200818)

[**4.3.3 Remove Property** 25](#_Toc508200819)

[**4.3.4 Query Property** 27](#_Toc508200820)

[**4.4.** **Tenants** 29](#_Toc508200821)

[**4.4.1.** **Rent Property** 29](#_Toc508200822)

[**4.4.2 Update Tenant** 33](#_Toc508200823)

[**4.4.3 Remove Tenant** 36](#_Toc508200824)

[**4.4.4 Query Tenant** 38](#_Toc508200825)

[**4.5 Process Admin** 40](#_Toc508200826)

[**4.5.1 Calculate Rent** 40](#_Toc508200827)

[**5.** **System Model** 42](#_Toc508200828)

[**5.1 Level-0 DFD** 42](#_Toc508200829)

[**5.2 Level-1 DFD** 43](#_Toc508200830)

[**5.3** **Level-2 DFD (Process P1: Owner)** 44](#_Toc508200831)

[**5.4** **Level-2 DFD (Process P2: Property)** 45](#_Toc508200832)

[**5.5** **Level-2 DFD (Process P3: Tenant)** 46](#_Toc508200833)

[**5.6** **Level-2 DFD** (Process P4: Admin) 47](#_Toc508200834)

[**6.** **Data Model (Class Diagram)** 48](#_Toc508200835)

[**6.1** **Class Diagram** 48](#_Toc508200836)

[**6.2** **Relational Schema** 49](#_Toc508200837)

[**6.3** **Database Schema** 50](#_Toc508200838)

[**7.** **Conclusion** 52](#_Toc508200839)

# **Introduction**

This property rental system is designed to allow admins to utilize owners, properties and tenants to save a database of properties owned by owners being rented to tenants. This project is intended on delivering a clear and concise format to deliver and smooth UI for administrators to utilize for their system. The system will use 3 main tables with 4 sub functions for each of them. I intend on developing a fully-embellished project later down the line.

# **Functional Components**

# **User Requirements**

## **PropertySys will perform Ownership management**

1.1 PropertySys will add a new owner.

1.2 PropertySys will update a specific owner.

1.3 PropertySys will remove a specific owner.

1.4 PropertySys will list details of a specific owner.

## **PropertySys will perform Property management**

2.1 PropertySys will add a new property.

2.2 PropertySys will update a specific property.

2.3 PropertySys will remove a specific property.

2.4 PropertySys will list a specific property.

## **PropertySys will perform Tenant’s management**

3.3 PropertySys will rent a property to a new tenant.

3.2 PropertySys will update a specific tenant.

3.4 PropertySys will remove a specific tenant.

3.1 PropertySys will list details of a specific tenant.

## **PropertySys will perform Administration analysis**

4.1 PropertySys will calculate the total of renting for specific tenants

# **System Requirements**

Include a brief overview of the system requirements.

Your hierarchy chart / User requirements summarise these……..

|  |  |  |
| --- | --- | --- |
| **Functional**  **Requirements** | **Non-Functional Requirements** | **Domain**  **Requirements** |
| 1. To give the manager the option of adding, updating, removing and querying owners.  2. To give the landlord the option of adding, updating, removing and querying properties.  3. To give the sales agent the option of adding, updating, removing and querying tenants.  2. To maintain a log of the changing of relations between tenants and properties along with the relationship between properties and owners.  3. The system will export a receipt in a .PDF format when a property is rented to a tenant and the receipt will be sent to the new tenant. | 1. The system must be accurate to the correct details. The relationships between the files must be maintained.  2. System must be easy to use for the different users so the design will be simplistic.  3. The system will be secure. Each separate will have their own unique login and password for the system. | 1. As the information in the software is highly sensitive, the software will not contain online capabilities but the manager, sales agent and landlord can access the software from the sales office. |

## **System Level Use Case Diagram**

SYSTEM

ManagerSales Agent

Applicant

Landlord

## **Manage Owners**

This set of functions allows a manager to input, edit and query necessary information regarding owners that they might need within their system.

### **Add Owner**

This function records an owners details on the system. An owner is identified by an OwnerID.

Activity Diagram: Add Owner

Manager System

Invokes Add Owner

Assign Owner ID

Display UI

Enter Data

Reset UI

Display Confirmation Message

Continue

Save Owner Details

Save Owner Details

Error Message

Valid?

Validate Data

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Owner** | |
| **Use Case Id** | 4.1.1 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Owner | |
| **Description** | This function records an owner’s details on the system. | |
| **Preconditions** | All owners must be verified beforehand to be on the list of owners. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the Add Owner function.  **Step 4:** Manager enters the required data:   * Forename * Surname * Address Line 1 * Address Line 2 * County * Phone * Email | **Step 2:** The system assigns the next Owner ID.  **Step 3:** System displays the UI  **Step 5:** System validates the data:   * All fields must be entered * Phone must be numeric only * Email must be valid format   **Step 6:** System assigns Activity as ‘A’ meaning the Owner is Active  **Step 7:** Save data in Owners File  **Step 8:** Display a confirmation Message  **Step 9:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Field not entered:** |  | **Step 5:** A blank field detected  **Step 6**: Display message “This field must be entered”  **Step 7:** Position cursor in offending field and return to step 4 |
| **Non-numeric entered in phone:** |  | **Step 5:** A non-numeric detected in phone field  **Step 6:** Display message “The field must be all numeric characters”  **Step 7:** Position cursor in offending field and return to step 4 |
| **Email address had invalid format:** |  | **Step 5**: An invalid format entered for email  **Step 6:** Display message “Email format invalid”  **Step 7:** Position cursor in Email field and return to step 4 |
| **Conclusions** | The valid owner has been added to the list of owners. | |
| **Post conditions** | This owner is now able to own properties from the properties file. | |
| **Business Rules** | The owner must have a valid email address. | |
| **Implementation Constraints** |  | |

### **Update Owner**

This function updates an owner’s details on the system. An owner is identified by their OwnerID.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Owner** | |
| **Use Case Id** | 4.1.2 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Owner | |
| **Description** | This function allows administration to update any specific owners within the owners file. | |
| **Preconditions** | The owner must be verified beforehand to be on the list of owners. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the update owner function.  **Step 3:** Manager enters owner surname (or part of).  **Step 6:** Manager selects owner to update from DataGrid.  **Step 9:** Manager updates owner details:   * Surname * Forename * Street * Town * County * Phone * Email | **Step 2:** Display UI.  **Step 4:** The system checks if the search field was entered.  **Step 5:** The system retrieves a list of all owners with matching surname from the Owners file and displays it on a DataGrid in the UI.  **Step 7:** System checks if owner is selected.  **Step 8:** Systemretrieves all details for the selected owner from the Owners file and populates text boxes with the data from the selected object.  **Step 10:** System validates Owner details:   * All fields must be entered * Phone must be numeric only * Email address must have correct format   **Step 11:** Update owner details in Owner File.  **Step 12:** Display Confirmation message  **Step 13:** Clear the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Field not entered:** |  | **Step 10:** A blank field detected  **Step 11:** Display message “All fields must be entered”  **Step 12:** Position cursor in offending field and return to step 9 |
| **Non-numeric entered in phone:** |  | **Step 10:** A non-numeric detected in phone field  **Step 11:** Display message “This field must be all numeric characters”  **Step 12:** Position cursor in offending field and return to step 9 |
| **Email address has invalid format:** |  | **Step 10:** An invalid format entered for email  **Step 11:** Display message “This field must have a valid format”  **Step 12:** Position cursor in Email field and return to step 9 |
| **Surname search field not entered:** |  | **Step 5:** A blank field detected  **Step 6:** Display message “A Surname must be entered here”  **Step 7:** Position cursor in offending field and return to step 4 |
| **Conclusions** | The specific owner data has been updated. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Remove Owner**

This function allows the user to remove a specific owner from the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Remove Owner** | |
| **Use Case Id** | 4.1.3 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Owner | |
| **Description** | This function allows administration to remove any specific owners within the owners file. | |
| **Preconditions** | The owner must be verified beforehand to be on the list of owners. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the remove owner function.  **Step 3:** Manager enters owner surname (or part of)  **Step 6:** Manager selects owner to remove. | **Step 2:** Display UI  **Step 4:** System validates that the field was entered  **Step 5:** The system retrieves a list of all owners with matching surname from the Owners file and displays it on a DataGrid in the UI.  **Step 7:** Systemretrieves all details for the selected owner from the Owners file and populates text boxes with the data from the selected object.  **Step 8:** System sets status for specified Owner as 'I' meaning the Owner is Inactive.  **Step 9:** Display Confirmation message  **Step 10:** Clear the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Surname search field not entered:** |  | **Step 4:** A blank field detected  **Step 5:** Display message “This field must be entered”  **Step 6:** Position cursor in offending field and return to step 3 |
| **Conclusions** | The selected owner’s status has been set to Inactive | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Query Owner**

This function will query for a specific owner that is verified and listed on the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Query Owners** | |
| **Use Case Id** | 4.1.4 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manger | |
| **Other Participating Actors** | Owner | |
| **Description** | This function lists all owners on the system. | |
| **Preconditions** | All owners must be verified beforehand to be on the list of owners. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The admin invokes the list all owners function.  **Step 3:** Manager enters owner surname (or part of) | **Step 2:** System displays the UI  **Step 4:** System validates that the field was entered  **Step 5:** The system retrieves a list of all owners with matching surname from the Owners file and displays it on a DataGrid in the UI. |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Surname search field not entered:** |  | **Step 4:** A blank field detected  **Step 5:** Display message “This field must be entered”  **Step 6:** Position cursor in offending field and return to step 3 |
| **Conclusions** | The owner listed is a verified property owner. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## **Manage Property**

This set of functions allows a manager to input, edit and query necessary information regarding owners that they might need within their system.

### **Add Properties**

This function allows the manager to allocate new properties to certain properties on the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Add Property** | |
| **Use Case Id** | 4.2.1 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Owner | |
| **Description** | This function records a properties details on the system. | |
| **Preconditions** | All properties must be verified to be on the list of properties. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the Add Property function.  **Step 3:** Manager enters the surname of the owner they want to add a property to  **Step 6:** Manager selects an Owner from the list of results in the DataGrid.  **Step 10:** Manager enters the required data:   * Rent per month * Bedrooms * Bathrooms * House Type * Street * Town * County | **Step 2:** System displays the UI  **Step 4:** System checks that the surname was entered  **Step 5:** The system retrieves a list of all owners with matching surname from the Owners file and displays it on a DataGrid in the UI.  **Step 7:** Systemretrieves all details for the selected owner from the Owners file and populates text boxes with the data from the selected object.  **Step 8:** The system assigns the next Property ID.  **Step 9:** System displays property fields for the manager to enter  **Step 11:** System validates the data:   * All fields must be entered * Rent per month must be numeric only   **Step 12:** System assigns status ‘A’ to indicate the property is now Active.  **Step 13:** System assigns OwnerID to the new property based on the selected owner result.  **Step 14:** Save data in Properties File  **Step 15:** Display a confirmation Message  **Step 16:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Surname search field not entered:** |  | **Step 5:** A blank field detected  **Step 6:** Display message “This field must be entered”  **Step 7:** Position cursor in offending field and return to step 3 |
| **Non-numeric entered in Rent per month:** |  | **Step 10:** A non-numeric detected in phone field  **Step 11:** Display message “The field must be all numeric characters”  **Step 12:** Position cursor in offending field and return to step 3 |
| **Field not entered:** |  | **Step 10:** A blank field detected  **Step 11:** Display message “This field must be entered”  **Step 12:** Position cursor in offending field and return to step 9 |
| **Conclusions** | The property has been added to the list. | |
| **Post conditions** | The property is now able to house tenants. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **Update Property**

This function updates a properties details on the system. A property is identified by their PropertyID.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Property** | |
| **Use Case Id** | 4.2.2 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Owner | |
| **Description** | This function allows administration to update any specific properties within the properties file. | |
| **Preconditions** | The property must be verified beforehand to be on the list of properties. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the update property function.  **Step 3:** Manager enters property street  **Step 6:** Manager selects property to update.  **Step 9:** Manager enters updated property details, which may be one of:   * Rent per month * Bedrooms * Bathrooms * House Type * OwnerID | **Step 2:** Display UI  **Step 4:** System validates property field was entered  **Step 5:** The system retrieves a summary of all properties with matching street from the properties file and displays on UI  **Step 7:** System validates that a property was selected  **Step 8:** System retrieves all details for the selected owner from the UI and displays on UI  **Step 10:** System validates property details:   * All fields must be entered * Rent per month must be numeric only.   **Step 11:** Save updated property details in Property File.  **Step 12:** Display Confirmation message  **Step 13:** Clear the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Field not entered:** |  | **Step 10:** A blank field detected  **Step 11:** Display message “This field must be entered”  **Step 12:** Position cursor in offending field and return to step 9 |
| **Non-numeric entered in Rent per month (RPM):** |  | **Step 10:** A non-numeric detected in RPM field  **Step 11:** Display message “The field must be all numeric characters”  **Step 12:** Position cursor in offending field and return to step 9 |
| **Street search not entered:** |  | **Step 4:** Search field not entered  **Step 5:** Display message “Search field not entered!”  **Step 6:** Position cursor in Search field and return to step 3 |
| **Conclusions** | The specific property data has been updated. | |
| **Post conditions** | All of these properties are able to be rented out by owners and to tenants. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **4.3.3 Remove Property**

This function allows the user to remove a specific owner from the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Remove Property** | |
| **Use Case Id** | 4.2.3 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Owner | |
| **Description** | This function allows administration to remove any specific properties within the property file. | |
| **Preconditions** | The property must be verified beforehand to be on the list of properties. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the remove properties function.  **Step 3:** Manager enters property street  **Step 6:** Manager selects property to remove. | **Step 2:** Display UI  **Step 4:** System validates that the field was entered  **Step 5:** The system retrieves a summary of all properties with matching street from the Properties file and displays on UI  **Step 7:** System checks if a property was selected  **Step 8:** System sets status for specified Property as 'Inactive'  **Step 9:** Save updated property details in Property File.  **Step 10:** Display Confirmation message  **Step 11:** Clear the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Search field not entered:** |  | **Step 4:** A blank field detected  **Step 5:** Display message “This field must be entered”  **Step 6:** Position cursor in offending field and return to step 3 |
| **Property not selected:** |  | **Step 7:** Property not selected  **Step 8:** Display message “Property must be selected!”  **Step 9:** Position cursor in offending field and return to step 3 |
| **Conclusions** | The specific owner data has been updated. | |
| **Post conditions** | All of these properties are able to be rented out by owners and to tenants. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **4.3.4 Query Property**

This function allows the manager to search for a property that is listed on the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Query Property** | |
| **Use Case Id** | 4.2.4 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manger | |
| **Other Participating Actors** | Owner | |
| **Description** | This function lists all properties on the system. | |
| **Preconditions** | All properties must be verified beforehand to be on the list of properties. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The admin invokes the list all properties function.  **Step 3:** Manager enters property street  **Step 6:** Manager selects which properties details they would like to see | **Step 2:** System displays the UI  **Step 4:** System validates that the field was entered  **Step 5:** The system retrieves a summary of all properties with matching street from the Properties file and displays on UI  **Step 7:** System validates that the field was entered  **Step 8:** System displays all relevant information of the specific property  **Step 9:** Display Confirmation message  **Step 10:** Clear UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Property not selected:** |  | **Step 7:** A blank field detected  **Step 8:** Display message “An owner must be selected!”  **Step 9:** Position cursor in offending field and return to step 6 |
| **Street not entered:** |  | **Step 4:** A blank field detected  **Step 5:** Display message “This field must be entered”  **Step 6:** Position cursor in offending field and return to step 3 |
| **Conclusions** |  | |
| **Post conditions** | This property is able to be rented out to tenants by an owner. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## **Tenants**

### **Rent Property**

This function allows the manager to allocate new tenants to certain properties on the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Rent Property** | |
| **Use Case Id** | 4.3.2 | |
| **Priority** | High | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Tenant, Owner | |
| **Description** | This function allocates a new tenant to a property which has an owner | |
| **Preconditions** | The Tenant must be 18 years of age to rent a property | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the Rent Property function  **Step 4:** Manager one or any of the following:   * Town * No Beds   **Step 7:** Manager selects the specific property to be rented and selects New or Existing tenant.  **Step 8: If existing tenant go to step13.**  **Step 10:** Manager enters the required data:   * Forename * Surname * Phone * Email * DOB   Step 13: manager enters tenant surname or part of.  Step 16: Manager selects the tenant renting the property and goes to step 18 | **Step 2:** The system assigns the next BookingID  **Step 3:** System displays the UI  **Step 5:** System checks that at least one field is entered    **Step 6:** The system retrieves all available properties with matching criteria available on the Start\_Date from the Property file and the Rentals File and displays on the UI  **Step 9: System prompts Manager to enter tenant details.**  **Step 11:** System validates the data:   * All fields must be entered * Phone must be numeric only * Email must be correct format * DOB must be over the age of 18   Step 12: System assigns next Tenant\_ID and go to step 17.  **Step 14: Sysytem checks that surname is entered**  **Step 15: System retrieves a summary of all tenants with matching surname and displays on UI**  **Step 17:** Save Tenant data in Tenants File:   * **Tenant\_ID** * Forename * Surname * Phone * Email * DOB   Step 18: Set End\_Date to Start\_Date plus 12 months  **Step 19:** Save Rental data in Rentals File:   * Rental\_ID * Prop\_ID * Tenant\_ID * Start\_Date * End\_Date * Monthly\_Rent   **Step 20:** Display a confirmation Message  **Step 21:** Reset UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Tenant field not entered:** |  | **Step 9:** A blank field detected  **Step 10:** Display message “This field must be entered”  **Step 11:** Position cursor in offending field and return to step 8 |
| **Non-numeric entered in phone field:** |  | **Step 9:** A non-numeric detected in phone field  **Step 10:** Display message “The field must be all numeric characters”  **Step 11:** Position cursor in offending field and return to step 8 |
| **Email address has incorrect format:** |  | **Step 9:** An invalid format entered for email  **Step 10:** Display message “Email format invalid”  **Step 11:** Position cursor in Email field and return to step 8 |
| **DOB has to 18 years of age or older:** |  | **Step 9:** An invalid date format entered  **Step 10:** Display message “Age has to be 18 years of age or older”  **Step 11:** Position cursor in offending field and return to step 8 |
| **Street search field not entered:** |  | **Step 5:** A blank field detected  **Step 6:** Display message “This field must be entered”  **Step 7:** Position cursor in offending field and return to step 4 |
| **Property not selected:** |  | **Step 7:** A blank field detected  **Step 8:** Display message “A property must be selected”  **Step 9:** Position cursor in offending field and return to step 6 |
| **Conclusions** | The tenant is now on the list of valid tenants. | |
| **Post conditions** | The tenant has been assigned to a property of their choosing from the properties file. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **4.4.2 Update Tenant**

This function updates a tenants details on the system. A tenant is identified by their TenantID.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Update Tenant** | |
| **Use Case Id** | 4.3.2 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Tenant | |
| **Description** | This function allows administration to update any specific tenant within the tenant file. | |
| **Preconditions** | The tenant must be verified beforehand to be on the list of tenants. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the update tenant function.  **Step 3:** Manager enters tenant surname (or part of)  **Step 5:** Manager selects tenant to update.  **Step 8:** Manager enters updated tenant details, which may be one of:   * Forename * Surname * Phone * Email * DOB * Status * PropID | **Step 2:** Display UI  **Step 4:** The system retrieves a summary of all tenants with matching surname from the Tenant file and displays on UI  **Step 6:** System checks that a tenant was selected  **Step 7:** System retrieves all details for the selected tenant from the UI and displays on UI  **Step 9:** System validates tenant details:   * All fields must be entered * Phone must be numeric only. * Email has correct format * DOB is over the age of 18   **Step 10:** Save updated tenant details in Tenant File.  **Step 11:** Display Confirmation message  **Step 12:** Clear the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Field not entered** |  | **Step 8:** A blank field detected  **Step 9:** Display message “This field must be entered”  **Step 10:** Position cursor in offending field and return to step 7 |
| **Non-numeric entered in Phone field** |  | **Step 8:** A non-numeric detected in RPM field  **Step 9:** Display message “The field must be all numeric characters”  **Step 10:** Position cursor in offending field and return to step 7 |
| **Email address has invalid format** |  | **Step 8:** An invalid format entered for email  **Step 9:** Display message “Email format invalid”  **Step 10:** Position cursor in Email field and return to step 7 |
| **DOB enter as under the age of 18:** |  | **Step 8:** DOB under the age of 18 entered  **Step 9:** Display message “Invalid date of birth entered”  **Step 10:** Position cursor in offending field and return to step 7 |
| **Tenant not selected:** |  | **Step 6:** Tenant not selected  **Step 7:** Display message “A tenant must be selected”  **Step 8:** Position cursor in offending field and return to step 6 |
| **Search field not entered** |  | **Step 5:** A blank field detected  **Step 6:** Display message “This field must be entered”  **Step 7:** Position cursor in offending field and return to step 4 |
| **Conclusions** | The specific tenant data has been updated. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **4.4.3 Remove Tenant**

This function allows the user to remove a specific tenant from the system

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Remove Tenant** | |
| **Use Case Id** | 4.3.3 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Tenant | |
| **Description** | This function allows administration to remove any specific tenants from within the tenant file. | |
| **Preconditions** | The tenant must be verified beforehand to be on the list of tenants. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the remove tenant’s function.  **Step 3:** Manager enters tenant surname (or part of)  **Step 6:** Manager selects Tenant to remove. | **Step 2:** Display UI  **Step 4:** System validates that the field was entered  **Step 5:** The system retrieves a summary of all tenants with matching surname from the Tenants file and displays on UI  **Step 7:** System checks that a tenant was selected  **Step 8:** System sets status for specified Tenant as 'Inactive'  **Step 9:** Save updated Tenant details in Tenant File.  **Step 10:** Display Confirmation message  **Step 11:** Clear the UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Field not entered** |  | Step 4: A blank field detected  Step 5: Display message “This field must be entered”  Step 6: Position cursor in offending field and return to step 3 |
| **Search field not entered:** |  | **Step 4:** A blank field detected  **Step 5:** Display message “This field must be entered”  **Step 6:** Position cursor in offending field and return to step 3 |
| **Conclusions** | The specific tenant data has been updated. | |
| **Post conditions** | The tenant can no longer rent out any property on the system. | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

### **4.4.4 Query Tenant**

This function will list the tenant that are verified and listed on the system.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Query Tenant** | |
| **Use Case Id** | 4.3.4 | |
| **Priority** | Low | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** | Tenant | |
| **Description** | This function lists all tenants on the system. | |
| **Preconditions** | All tenants must be verified beforehand to be on the list of tenants. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the list all tenants function.  **Step 3:** Manager enters tenant surname (or part of)  **Step 6:** Manager selects which tenant they want to query | **Step 2:** System displays the UI  **Step 4:** System validates that the field was entered  **Step 4:** The system retrieves a summary of all tenants with matching surname from the Tenant file and displays on UI  **Step 7:** System validates that the field was entered  **Step 8:** System displays all relevant information of the specific tenant from the Tenant file.  **Step 9:** Display Confirmation message  **Step 10:** Clear UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
|  |  |  |
| **Conclusions** | The tenant specified is a valid tenant in the Tenant file. | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

## **4.5 Process Admin**

### **4.5.1 Calculate Rent**

This function allows the Manager to calculate the total rent for the month for a list of tenants or all tenants.

Manager

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Calculate Rent** | |
| **Use Case Id** | 4.5.1 | |
| **Priority** | Medium | |
| **Source** | Manager | |
| **Primary Business Actor** | Manager | |
| **Other Participating Actors** |  | |
| **Description** | This function allows the Manager to calculate the rent for a specific list of tenants | |
| **Preconditions** | All tenants must be verified beforehand to be on the list of properties. | |
| **Trigger** |  | |
| **Expected Scenario** | **Manager** | **System** |
|  | **Step 1:** The manager invokes the Calculate Rent function.  **Step 3:** Manager selects tenants from the list until they press calculate button  **Step 7:** Manager selects either yes or no | **Step 2:** System displays the UI  **Step 4:** System validates that a tenant was entered  **Step 5:** The system retrieves a summary of all tenant with matching attribute data from the search result and displays on UI  **Step 6:** System asks Manager is that all the tenants they wish to calculate from**.**  **Step 8:** System validates that the yes button was entered  **Step 9:** System uses the PropID of each Tenant to add all of the Rent per Month from the Properties file  **Step 10:** System displays the total on UI  **Step 11:** Display Confirmation message  **Step 12:** Clear UI |
| **Alternate Scenarios** | **Actor Action** | **System Response** |
| **Tenant not entered:** |  | **Step 4:** Tenant not entered  **Step 5:** Display message “Tenant must be selected”  **Step 6:** Position cursor in offending field and return to step 3 |
| **No option selected:** |  | **Step 8:** No option selected  **Step 9:** Display list of tenants again with the tenants previously selected in step 3 highlighted.  **Step 10:** Position cursor in offending field and return to step 3 |
| **Conclusions** | The rent of the specified tenants is calculated | |
| **Post conditions** |  | |
| **Business Rules** |  | |
| **Implementation Constraints** |  | |

# **System Model**

The following dataflow diagrams have been produced for the system:

## **5.1 Level-0 DFD**

Application Rent

Tenant

Rent Property

Applicant

Correspondence Property

## **5.2 Level-1 DFD**

P2

Process

Property

Applicant

Application PropID Rent

D2 Property

P3

Process Tenant

Rent

OwnerID

Receipt Tenant Details

D1 Owner

D3 Tenants

P1

Process

Owner

## **Level-2 DFD (Process P1: Owner)**

P1.1

Add Owner

Details

New Owner

Details New Owner Details

P1.2

Update Owner

Owner DetailsCurrent Details

P1.3

Remove Owner

D1 Owner

Specific Owner

Inactive status

P1.4

Query Owner

## **Level-2 DFD (Process P2: Property)**

D1 Owner

P2.4

Query Property

D2 Property

P2.1

Add Property

New Property

Details Details

P2.2

Update Property

Details New Prop Details

Prop Details Current Details

P2.3

Remove Property

Specific Property

Inactive status

## **Level-2 DFD (Process P3: Tenant)**

D2 Property

P3.1

Rent Property

Details

Details

New Tenant

P3.2

Update Tenant

Details New Tenant Details

D3 Tenant

Tenant Details Current Details

P3.3

Remove Tenant

Specific Tenant

P3.4

Query Tenant

Inactive status

## **Level-2 DFD** (Process P4: Admin)

D2 Properties

D3 Tenants

Tenant Details Rent per month

P4.1

Calculate Rent

# **Data Model (Class Diagram)**

In this data model, there will be a class diagram which shows the way the property, owner and tenant files interact with each other. A relational schema which details the relational attributes of the various files and checks for relational integrity. A database schema which details the specific attributes of the files.

## **Class Diagram**



## **Relational Schema**

**Owner** {OwnerID, Forename, Surname, Street, Town, County, Phone, Email, DOB, Status, Dateregistered}

**Property** {PropID, Rentpermonth, Bedrooms, Bathrooms, Housetype, Street, Town, County, Status, OwnerID}

**Tenant** {TenantID, Forename, Surname, Phone, Email, DOB, Status, PropID}

## **Database Schema**

**Schema: Property Rental System**

**Relation: Owner**

Attributes:

OwnerID numeric(10) unique not null auto\_increment

Forename char(15) not null

Surname char(15) not null

Street char(20) not null

Town char(20) not null

County char(15) not null

Phone char(15)

Email char(25) not null

DOB Date

Status char(8) not null

Dateregistered Date not null

**Primary Key**: OwnerID

**Relation: Property**

Attributes:

PropID numeric(10) unique not null auto\_increment

Rentpermonth numeric(5) not null

bedrooms numeric(2) not null

bathrooms numeric(2) not null

housetype char(15) not null

Street char(20) not null

Town char(20) not null

County char(15) not null

Status char(8) not null

OwnerID numeric(10) unique

**Primary Key**: PropID

**Foreign Key:** OwnerID references OwnerID in Owner Table

**Relation: Tenant**

Attributes:

TenantID numeric(10) unique not null auto\_increment

Forename char(15) not null

Surname char(15) not null

Phone char(15) not null

Email char(25) not null

DOB Date not null

Status char(8) not null

PropID numeric(10) unique not null

**Primary Key**: TenantID

**Foreign Key**: PropID references PropID in Property Table

# **Conclusion**

This project was a beneficial experience in that it helped me learn about how to plan ahead of time. I encountered certain problems throughout this project, particularly developing my use-case narratives and their intricacies. It proved to be a useful exercise in foresight and discipline for developing work on my project. In hindsight, I would have learned earlier about the details of the particular pieces of the project.

