RENTOMAX PROJECT MANUAL

GROUP C3

***GROUP NAME: The Girls!***

*Scrum Master: Clara Wilt*

*Scrum Master Email:* [*caw00688@uga.edu*](mailto:caw00688@uga.edu)

*Team Member: Maddie Lu*

*Email:* [*mel41242@uga.edu*](mailto:mel41242@uga.edu)

*Team Member: Valerie Penaranda*

*Email:* [*vp53834@uga.edu*](mailto:vp53834@uga.edu)

*Team Member: Shreeya Chalasani*

*Email:* [*sc09737@uga.edu*](mailto:sc09737@uga.edu)

*Team Member: Simran Singh*

*Email:* [*shs70191@uga.edu*](mailto:shs70191@uga.edu)

*Team Member: Christina Zaparianova*

*Email:* [*christinazaprianova@gmail.com*](mailto:christinazaprianova@gmail.com)

MIST 4620– Systems Analysis and Design Team Project Manual

Spring 2024

**Introduction**

**Objective:** The objective of the team project is to analyze system requirements using the tools and techniques taught in the class and to produce a working prototype of a system.

**Prototype System:** The description of the system and a set of preliminary requirements will be provided to you later. Like in any project, requirements are likely to change during the course of the project. You will have the opportunity to meet with the project sponsor and elicit/clarify requirements. Eliciting requirements, clarifying these, and modeling these are an important part of a systems analyst’s job.

**Software Tools:** This will be a web-based system. You will use the Salesforce platform (https://[www.salesforce.com/)](http://www.salesforce.com/)) to develop the system. Low-code platforms, such as Salesforce, are a very prevalent means of developing systems. Being exposed to and developing skills in using these is a valuable asset in the job market.

**Methodology:** Teams will use an Agile methodology based on SCRUM. Agile methodologies are being used by many companies. The core idea is to produce fully functioning system feature(s) in a short period of time (a Sprint) using self-managing teams. The system you develop for the course will consist of two 11-day sprints each involving a different feature or features of the system.

The project functionality is first expressed as the set of features (user stories) that constitute the scope of the system (Product Backlog). Each sprint involves one or more user stories. For each sprint, the team develops a task list (the Sprint Backlog) which includes everything that is needed to completely implement the sprint user stories – analysis, development, deployment and testing.

**Deliverables:** The deliverables consist of text files and various models produced during analysis. As the sprint prototypes (plus final project) are completed, a video demo and file should be created that explains how to access the prototype and provides a short user manual.

**Description of Deliverables**

### [DELIVERABLE #1](#_bookmark0)

* 1. [​Project Team Organization](#_bookmark1)
  2. [​System Overview and Visioning](#_bookmark2)
  3. [​System Analysis](#_bookmark3)
  4. [Identification of Prototype Use Cases](#_bookmark4)

### [DELIVERABLE #2](#_bookmark5)

* [List of user stories implemented in Sprint 1](#_bookmark6)
* [A working prototype](#_bookmark5)

1. [**DELIVERABLE #3** (FINAL PROJECT DELIVERABLE)](#_bookmark7)

* [List of user stories implemented in Sprint](#_bookmark6) 2
* [Final Project Deliverable (Including cumulative file of all three deliverables);](#_bookmark8)
* [A working prototype](#_bookmark9)

### [PROJECT PRESENTATION GUIDELINES](#_bookmark10)

**SUBMISSION INSTRUCTIONS**

1. All project deliverables should be submitted on eLC. There will be eLC assignment dropbox folders for each project deliverable. The deliverables should be submitted as WORD files. They should be named as Del#GROUPNAME (e.g., Deliverable 1 for the Team Dawgs should be named Del1DAWGS).
2. All deliverables should be submitted in a single integrated file that accumulates over time. For example, the file submitted for deliverable 3 should contain deliverables 1 and 2 as well.
3. The submission should have a cover page that includes the Group Name, name and email of every member of the group, and the name and email of the scrum master of the group with whom the course instructor may need to communicate should there be an issue with your deliverable. Then, there should be a section page for Deliverable 1 (i.e., a page that says Deliverable 1) – and then deliverable 1, a section page for Deliverable 2, then deliverable 2, a section page for Deliverable 3, and then Deliverable 3.

DELIVERABLE 1

# DELIVERABLE # 1

## Project Team Organization

### BEFORE WORKING ON THE DELIVERABLE

The various Agile methodologies stress the use of self-managing teams. Team success has been shown to be dependent on trust between team members. Producing various parts of Deliverable #1 is an opportunity to creatively discuss how members of the team will work together. Since the methodology will be SCRUM-based, team members should review the SCRUM materials (Scrum Guide [on eLC], Scrum Basics 101- videos 1-4 and Agile Simulation Videos [1](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-1-meet-the-agile-team-2/), [2](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-2-agile-lifecycle-overview/), and [3](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-3-purpose-agenda-team-norms/) before completing this deliverable.

If you are prompted to pay for access to the content, you can create an account **with your UGAMail email address** at this URL: <https://agilevideos.com/university-georgia-student-registration>

#### SCRUM Team Biography

Complete the following (*in the table on the next page*):

* + - * 1. Select one member of the team to serve as the scrum master (facilitator). This person will also serve as a liaison between the team and the instructor in case the instructor needs to contact your group. Good organizational skills, willingness to confront team members when necessary, willingness to facilitate the SCRUM process, and a strong interest in the project itself are critical skills for the scrum master.
        2. Most projects require the team to have members who are strong in facilitation skills (as described above), talking with clients, analyzing requirements, business writing, database design, programming, presentation development, presentation, overall technical skills, and overall analytical skills.

Which people in your group have which strengths? **List your group members and the strengths from the list above that each member feels s/he contributes to the team**. Also list each member’s areas where they feel they are weak (some areas may be neither a member’s strength nor a weakness so you can omit those).

Based on each person’s strengths, what are the duties/role(s) of each team member? I need to know at a minimum who is your go-to person as **business analyst, developer, database expert, user experience expert, client liaison, and scrum master**. Note that multiple people can serve in each role. Though the idea behind SCRUM is for each member to be a generalizing specialist (that is, be able to pitch in across areas), it is also good to know who has in-depth skills in an area (i.e., T-shaped skillset). So you may want to indicate who is/are your primary team members in an area and who can also pitch as secondary.

* + - * 1. Now think of your team **as a whole**. In which of the above areas is your team strongest, and in which is it not as strong? (Be honest.)

### SCRUM TEAM BIOGRAPHY & TEAM MEMBER ROLES

***GROUP NAME: The Girls!***

*Scrum Master Name: Clara Wilt*

*Scrum Master Email: caw00688@uga.edu*

|  |  |  |
| --- | --- | --- |
| ***Group Member*** | ***Skills*** | ***Role(s)*** |
| *Name: Christina Zaparianova*  *Email: christinazaprianova@gmail.com* | *Strengths: Writing, Process Diagrams, Graphic Design*  *Weaknesses: Salesforce, Math* | *Primary: User Experience Expert*  *Secondary: Business Analyst* |
| *Name: Maddie Lu*  *Email: mel41242@uga.edu* | *Strengths: Process Diagrams, Writing, Math, Analysis*  *Weaknesses: Salesforce* | *Primary: Business Analyst*  *Secondary: User Experience Expert* |
| *Name: Valerie Penaranda*  *Email: vp53834@uga.edu* | *Strengths: Salesforce, Organization, Process Diagrams*  *Weaknesses: Writing* | *Primary: Developer*  *Secondary: Scrum Master* |
| *Name: Clara Wilt*  *Email: caw00688@uga.edu* | *Strengths: Organization, Process Diagrams*  *Weaknesses: Salesforce* | *Primary: Scrum Master*  *Secondary: Client Liaison* |
| *Name: Simran Singh*  *Email: shs70191@uga.edu* | *Strengths: Written Communication, SQL*  *Weaknesses: Salesforce, R* | *Primary: Database Expert*  *Secondary: Client Liaison* |
| *Name: Shreeya Chalasani*  *Email: sc09737@uga.edu* | *Strengths: Data Visualizations, Business Process Diagrams*  *Weaknesses: Salesforce* | *Primary: Client Liaison*  *Secondary: Database Expert* |

**TEAM STRENGTHS:** Our team's strongest skills are analysis, problem solving, and organization.

**TEAM WEAKNESSES:** We concluded our weaknesses are salesforce and database management.

## System Overview and Visioning

#### Client Organization Background: (1-2 pages)

1. **Description of Organization:** What organization/department is the system developed for? Briefly provide a short overview of this organization/department.

For this project, we are developing a system for the company RentoMax. RentoMax is an apartment rental company that manages its properties across various locations in Atlanta. Currently, the company manages their processes on paper and manually, so it wants to implement an automated system for its key processes with Salesforce.

Some of the key features that RentoMax wants to gain is the ability to add properties and tenants and keep records of them in the system. This is especially vital for this company because it is an apartment rental company. The units the property contains consist of many features that need to be put into the system, as well as linking property managers into the system in order to help automate everything. It’s also important that tenants can send maintenance requests, and this will also be another thing the system will be able to help automate.

1. **Stakeholders** (watch [https://agilevideos.com/videoscategory/agile-simulation-](http://agilevideos.com/videos/agile-simulation-part-6-stakeholder-analysis/) [videos/lessons/agile-simulation-part-6-stakeholder-analysis](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-6-stakeholder-analysis/)/[)](http://agilevideos.com/videos/agile-simulation-part-6-stakeholder-analysis/): Who are the stakeholders and why? Draw a stakeholder context diagram.

This system will have many stakeholders relying on it. The following diagram will contain the stakeholders from the organization.

|  |  |  |
| --- | --- | --- |
| Stakeholders Diagram | Who | Why |
|  | RentoMax | RentoMax is the company that wants the automated system, so they would be the primary stakeholder due to this. |
|  | Tenants | The tenants that live in the RentoMax properties are also stakeholders. Since they live in those units, they have a large stake in this automation process as their livelihood depends on it. |
|  | Employees | The employees at RentoMax, for example the property managers and maintenance staff, are also stakeholders. Their livelihood also depends on this project since they work for RentoMax, and also this will make them more efficient in their jobs once everything is automated. This will make processes smoother and quicker. |

Stakeholder Context Diagram

#### Users:

* 1. Who are the users for the system? Property Manager, Tenants, System Administrator
  2. Watch: [https://agilevideos.com/videoscategory/agile-simulation-](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-7-user-personas/) [videos/lessons/agile-simulation-part-7-user-personas/](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-7-user-personas/)and ***create 2-3 personas for your project***.

Name: Property Manager

Characteristics: Manages a single property consisting multiple units

Goals: Have all information in a centralized system (tenant information and unit information)

Likes: Centralized system to that allows for easy accessibility and use

Dislikes: Information is on paper

Name: Tenant

Characteristics: Occupies units and has maintenance requests

Goals: Easily request maintenance and have the request fulfilled quickly

Likes: Efficiently have requests fulfilled

Dislikes: Requests aren’t fulfilled in a timely manner

Name: System Administrator

Characteristics: Handles unresolved requests

Goals: Less volume of unresolved requests

Likes: Request is escalated immediately after 48 hours in centralized system

Dislikes: Unresolved requests

**Watch** [https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-4-vision-statement-product-box/) [simulation-part-4-vision-statement-product-box/](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-4-vision-statement-product-box/) & [https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-5-](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-5-product-visioning/) [product-visioning/](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-5-product-visioning/) for parts 1.2.2 – 1.2.4.

#### System Vision Document (1-2 pages – see p13 in Satzinger for an example)

1. Problem Description (provide a description of the problem/opportunity the project approvers find *compelling*; then describe how your solution will address the issue)
2. System Capabilities
3. Business Benefits

**Problem Description**

RentoMax, a apartment rental company operating primarily in Atlanta is currently facing significant inefficiencies and operational challenges. This is likely due to its reliance on paper-based processes, which are difficult to maintain updated and organized. The heavy use of manual methods results in a lack of centralized data management, leading to difficulties in effectively tracking properties, tenant management and handling maintenance requests. Consequently, RentoMax experiences delays in responding to tenant issues and potential errors in management.

We recommend implementing a Salesforce based system to automate RentoMax’s key business processes. By transitioning to a digital platform, RentoMax can streamline property and tenant management, as well as enhance the efficiency of maintenance request handling. The system will provide a centralized hub for managing properties, tenants, and maintenance requests, enabling seamless communication, data tracking, and reporting capabilities.

**System Capabilities**

Property Management: The system will allow RentoMax to effortlessly add, update, and manage properties across different locations in Atlanta. Property details, including unit specifications such as bedrooms, bathrooms, square footage, and amenities, will be meticulously cataloged and easily accessible within the system.

Tenant Management: RentoMax will have the ability to maintain comprehensive records of tenants, including their personal information, lease agreements, and contact details. The system will facilitate efficient tenant onboarding, lease management, and communication, thereby improving tenant satisfaction and retention rates.

Maintenance Request Handling: Tenants will be empowered to submit maintenance requests through various channels, including phone, email, or a user-friendly web interface. The system will categorize maintenance requests. Requests will be assigned to specialized maintenance staff, with automated status tracking.

Reporting and Dashboard: RentoMax will leverage the system's reporting and dashboard functionalities to gain valuable insights into maintenance request trends, response times, and overall property management performance. Customizable reports and visual dashboards will enable informed decision-making, resource allocation, and strategic planning.

**Business Benefits**

Increased Operational Efficiency: By automating manual processes and centralizing data management, RentoMax will experience significant improvements in operational efficiency, reducing time-consuming administrative tasks and minimizing the risk of errors.

Enhanced Tenant Satisfaction: The system's streamlined maintenance request handling and communication channels will lead to faster response times and improved tenant satisfaction. Prompt resolution of issues will contribute to higher tenant retention rates and more referrals.

* + - 1. **For all business benefits identified above, come up with conditions of satisfaction (i.e., project-level definition of done).**

Data Accuracy: Property and tenant data entered into the system must be accurate, up-to-date, and consistently maintained to ensure reliability and integrity.

Timely Request Handling: Maintenance requests should be categorized, assigned, and resolved within agreed-upon timeframes.

* + - 1. **Use the product-box technique to describe the vision for your system.**

Key Features:

* Automated property and tenant management
* Streamlined maintenance request handling
* Comprehensive reporting and dashboard capabilities

Benefits:

* Operational efficiency gains
* Enhanced tenant satisfaction and retention
* Informed decision-making for strategic planning

Target Users:

* Property managers
* Maintenance staff
* Tenants

Technology Stack:

* Salesforce platform
* Web interface for tenants
* Email and phone integration for maintenance requests

Integration:

* Seamless integration with existing Salesforce environment
* Mobile compatibility for on-the-go access

Release Plan:

* Phase 1: Basic property and tenant management functionality
* Phase 2: Implementation of maintenance request handling features
* Phase 3: Enhancements to reporting and dashboard capabilities

## System Analysis

**Information Gathering Techniques, Process Requirements, Use Case Diagram, User Interface Flow, Domain Class Diagram, CRUD Matrix**

System Analysis is essentially a high level analysis that covers the complete project scope. Watch [https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-8-use-case-diagrams/) [simulation-part-8-use-case-diagrams/](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-8-use-case-diagrams/) & [https://agilevideos.com/videoscategory/agile-](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-9-process-diagram-ui-flow/) [simulation-videos/lessons/agile-simulation-part-9-process-diagram-ui-flow/](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-9-process-diagram-ui-flow/)

#### Information Gathering Techniques

How did your group collect the information regarding the proposed system? Describe:

* the data gathering methods that you used and why (e.g., interviewing, questionnaires, review of best practices or other existing systems similar to what you are developing, etc.)
* the interview questions that you used
* any other pertinent information regarding other data gathering methods

#### When planning your interviews, questionnaires, and other methods, please be sure to take into account these three perspectives on the system: the process view, the user view, and the user interface view, as described in Chapter 3 of the textbook. These are used to build the Process Diagram, the Use Case Diagram, and the UI Flow Diagram respectively. Be sure that your questions will help you gather the information needed for these diagrams.

#### Conducting interviews and questionnaires with the stakeholders of RentoMax allows for us to effectively gather information on what requirements they have and what features they would like to see on developed on the system. Some of the questions asked:

#### What current issues are you facing in managing the rental properties?

#### What is the current process you have of managing maintenance requests?

#### What are some of the non-negotiable features you would like to see the new system have?

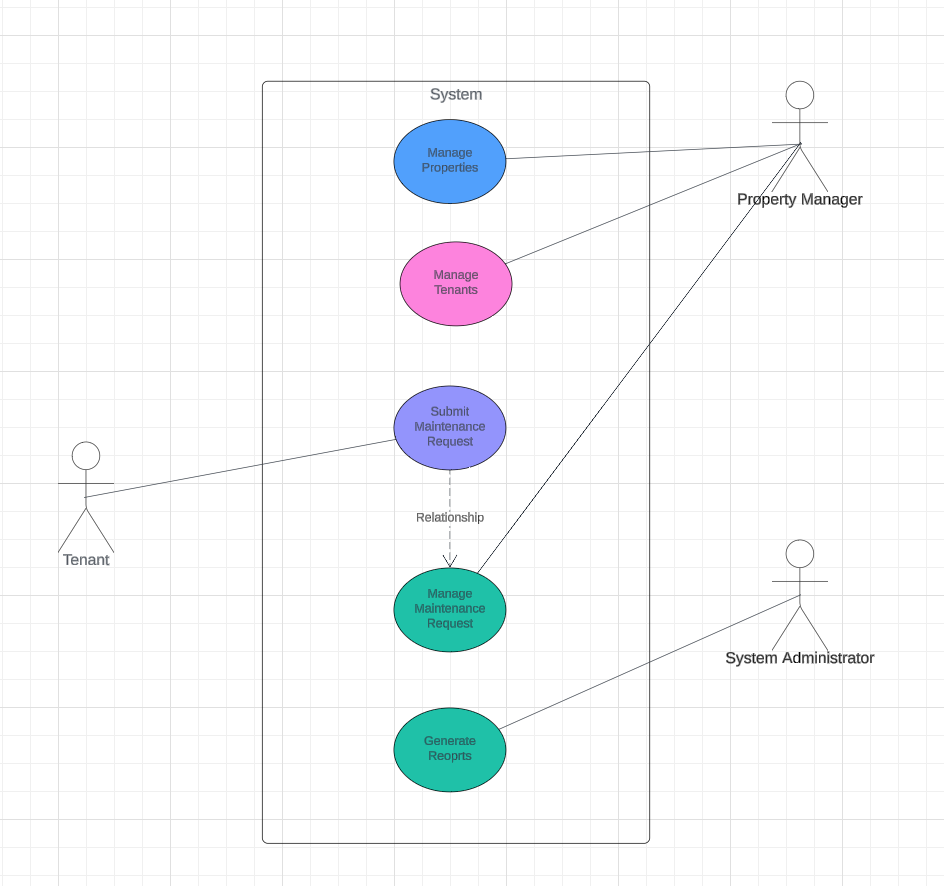
* + - 1. **Process Requirements**

This describes the OVERALL process (high level) for your proposed system. Use the modeling tool of your choice to model the overall business process.

|  |  |
| --- | --- |
| Business Process Name: | Goal/Objective: |
| Property Management | * Add new properties and their details to the system * Manage property information * Assign property managers to specific properties |
| Tenant Management | * Add new tenants and their information to the system * Assign tenants to specific units within properties * Track tenant lease agreements and rent payments |
| Maintenance Management | * Tenants submit maintenance requests (via phone, email, or web) * Classify requests by type (electrical or plumbing) * Assign maintenance requests to appropriate staff based on specialty (electrical or plumbing) * Track the status of maintenance requests (pending, in process, resolved) * Escalate unresolved requests (pending > 48 hours) to the system administrator |
| Reporting | * Generate reports on maintenance requests, including time to resolve and request status * Provide dashboards to view key metrics such as pending maintenance requests and completed requests |

#### Use Case Diagram

The Use Case Diagram (UCD) will depict the major functional requirements that will be delivered by your system. You should include a screenshot of your UCD from your modeling tool in your WORD file.

user

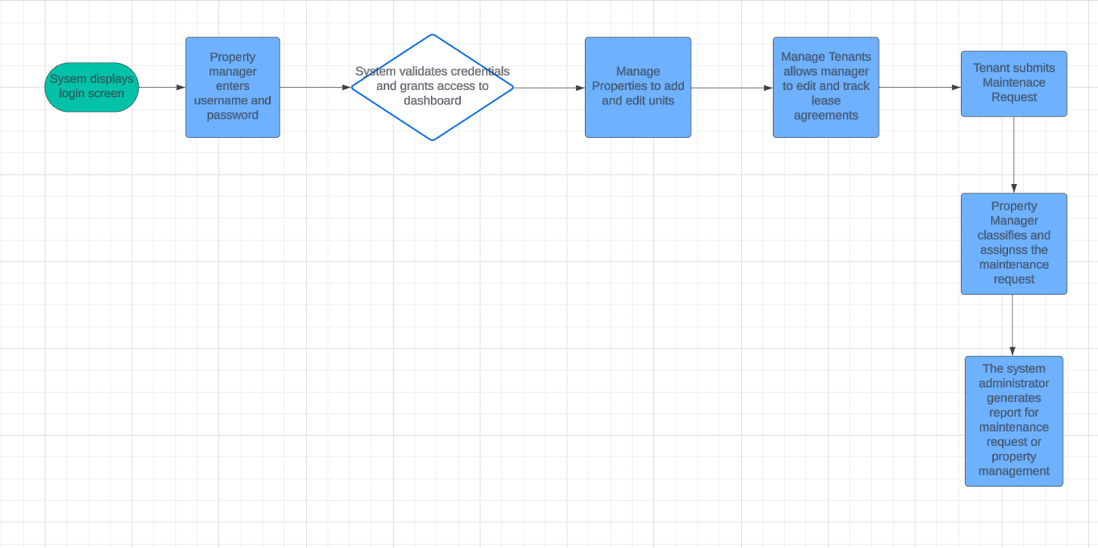
#### Brief Descriptions of your Use Cases

Include a table that provides a brief description of each use case included in your UCD. Column 1: Use case name (should correspond to your UCD)

Column 2: A brief but comprehensive description of functionality provided by the use case Column 3: Actors.

|  |  |  |
| --- | --- | --- |
| Use Case Name | Brief Description | Actor |
| Manage Properties | Add new properties and their details (including units)  Update existing property information  Assign property managers to specific properties | Property Manger |
| Manage Tenants | Add new tenants and their information  Assign tenants to specific units within properties  Manage lease agreements  Track rent payments | Property Manager |
| Submit Maintenance Requests | Tenants submit requests for maintenance (electrical or plumbing) through phone, email, or web interface | Tenant |
| Manage Maintenance Requests | Property manager classifies request type (electrical/plumbing)  Assigns maintenance request to appropriate | Property Manager |
| Generate Reports | System administrator generates reports on various aspects  Creates reports on maintenance requests (including time to resolve and request status)  Generates property management dashboards (showcasing pending/completed requests) | System Administrator |

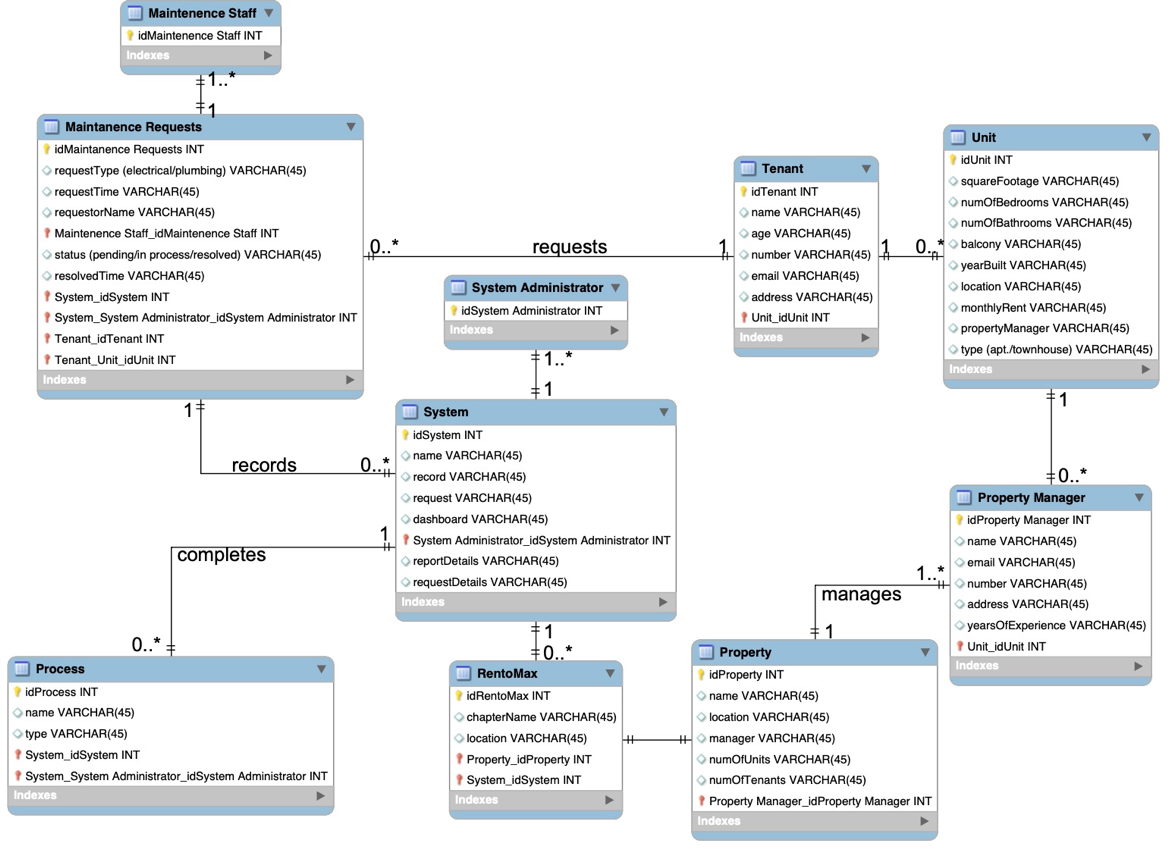
#### User Interface (UI) Flow – Shreeya

Include a diagram that shows the flow of UI screens. See [https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-9-](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-9-process-diagram-ui-flow/) [process-diagram-ui-flow/](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-9-process-diagram-ui-flow/) for an example

#### Domain Class Diagram (DCD) -- Simran

The Domain Class Diagram will show the major domain classes (entities and attributes) that your system will need to keep track of. You should include a screenshot of your DCD from your modeling tool in your WORD file.

**RentoMax Domain Class Diagram**



#### CRUD Matrix - Val

This shows which Use Case creates/reads/updates/deletes which domain class from your UCD. Table should have Use Cases as rows and classes as columns. The appropriate letters C, R, U, D (or blank) should be placed in each cell.

CRUD Matrix

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case vs. Entity/Domain Class | Tenant | System | Property Manager | System Administrator |
| Add Properties and Tenants |  | R | C |  |
| Manage Property |  |  | R |  | |
| Record Properties and Tenants |  | R | C |  |
| Store Tenant Information |  | R | C |  | |
| Send Maintenace Request | C | R |  |  | |
| Display Request Status |  | R |  |  |
| Escalate Request |  | U |  | R | |
| Report Request Details |  | C |  |  | |
| Show Dashboard |  | R |  |  | |

## Identification of Prototype User Stories

#### Break your Use Cases (Features) into User Stories

Decompose your MUST HAVE Use Cases into User Stories **if needed (sometimes this is not necessary)**. For example, a Use Case called “Submit Claim” may have multiple user stories such as “Enter Claim”, “Validate Claim”, “Send Notification”. (see [https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-10-](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-10-agile-writing-user-stories/) [agile-writing-user-stories/](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-10-agile-writing-user-stories/) & [https://agilevideos.com/videoscategory/agile-simulation-](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-11-story-slicing/) [videos/lessons/agile-simulation-part-11-story-slicing/](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-11-story-slicing/) [)](http://agilevideos.com/videos/agile-simulation-part-11-story-slicing/) and our class slides from chapter 3 that discuss various ways of doing this.) Present this in a table. Take your brief use case description table from PART 3 and include any user stories that derive from breaking down any of your use cases. Include a brief description of each.

As a property manager I want to manage properties by adding new properties and their details to the system.

As a tenant, I want to be able to submit maintenance requests through a variety of platforms.

As a system administrator, I want to be able to create reports on the time it takes to resolve maintenance requests and generate property dashboards.

#### Use Case Description

Using the ***Use Case Description template*** on eLC (also see Satzinger book), provide an overview of the highest priority user story on your product backlog.

|  |  |  |
| --- | --- | --- |
| Use Case Name | Brief Description | Actor |
| Manage Properties | Add new properties and their details (including units)  Update existing property information  Assign property managers to specific properties | Property Manger |
| Manage Tenants | Add new tenants and their information  Assign tenants to specific units within properties  Manage lease agreements  Track rent payments | Property Manager |
| Submit Maintenance Requests | Tenants submit requests for maintenance (electrical or plumbing) through phone, email, or web interface | Tenant |
| Manage Maintenance Requests | Property manager classifies request type (electrical/plumbing)  Assigns maintenance request to appropriate | Property Manager |
| Generate Reports | System administrator generates reports on various aspects  Creates reports on maintenance requests (including time to resolve and request status)  Generates property management dashboards (showcasing pending/completed requests) | System Administrator |

#### Prioritize User Stories

As Product Owners, we have already created a prioritized product backlog for you using the MoSCoW Methodology (Must Have, Should Have, Could Have, Won’t Have). You may revise the backlog to take into account dependencies among use cases in the backlog (e.g., “enter

claim” must be done before “validate claim”) (though we think we have done this for you already). In addition, include non-functional user stories and prioritize those as well (e.g., if you are storing information in a database, then you will need to set up your data model and tables before starting to develop any functionality). This final prioritized list of user stories is your product (aka release) backlog. This is what you need to turn in.

#### Estimate Effort

Estimate the amount of effort required to develop each user story in your product backlog. To estimate your effort: (a) use your use case descriptions and discuss the story to understand what is involved in developing each use case and each user story; (b) play planning poker with your team to estimate the number of story points (1, 2, 3, 5, 8, 13) for each user story/non-functional user story (these are indicators of complexity); (c) break down each user story into its set of activities to estimate the amount of time it would take to develop; (d) identify the capacity in hours that your team has available for a 11-day sprint; (e) estimate your velocity: the number of story points that you can get done in a sprint (based on capacity and corresponding number of points). (see [https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-14-story-point-sizing/) [simulation-part-14-story-point-sizing/](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-14-story-point-sizing/)[)](http://agilevideos.com/videos/agile-simulation-part-14-story-point-sizing/) (see our class slides on release planning, estimation).

Story Points Breakdown

|  |  |
| --- | --- |
| **Use Case** | **Story Points** |
| **Add Properties/Units** | **10** |
| * Create Properties | 5 |
| * Create Unit and Assign to Properties | 5 |
| **Assign Property Managers** | **3** |
| **Add Tenants** | **10** |
| * Create Tenants | 5 |
| * Assign Tenants to Units/Properties | 5 |
| **Submit Maintenace Request** | **10** |
| * Create Request Form | 5 |
| * Create Case | 5 |
| **Manage Maintenance Request** | **5** |
| **Display Request Status** | **3** |
| **Escalate Request** | **5** |
| **Generate Reports** | **10** |
| * Create Maintenance Request Details Report | 5 |
| * Create Maintenance Status Report | 5 |
| **Show Dashboard** | **7** |
| TOTAL | 63 |
| Velocity | 20 – 35 |

Team Time Capacity

|  |  |
| --- | --- |
| Team Member | Hours/11 Day Sprint |
| Valerie Penaranda | 2-6 |
| Clara Wilt | 2-6 |
| Maddie Lu | 2-6 |
| Christina Zaparianova | 2-6 |
| Shreeya Chalasani | 2-6 |
| Simran Singh | 2-6 |

#### Release Planning: Plan your Sprints

Based on the use cases, your prioritization, and your effort estimates, plan your sprints by creating an initial sprint backlog. What user stories will be developed in which sprint? You have two sprints each of which is 11 days long [(https://agilevideos.com/videoscategory/agile-](http://agilevideos.com/videos/agile-simulation-part-15-release-planning/) [simulation-videos/lessons/agile-simulation-part-15-release-planning/).](https://agilevideos.com/videoscategory/agile-simulation-videos/lessons/agile-simulation-part-15-release-planning/) Color code the stories for each sprint using a different color. Use this same color scheme on your product backlog for the sprint deliverables that follow.

|  |
| --- |
| **Use Case** |
| **Add Properties/Units** |
| * Create Properties |
| * Create Unit and Assign to Properties |
| **Assign Property Managers** |
| **Add Tenants** |
| * Create Tenants |
| * Assign Tenants to Units/Properties |
| **Submit Maintenace Request** |
| * Create Request Form |
| * Create Case |
| **Manage Maintenance Request** |
| **Display Request Status** |
| **Escalate Request** |
| **Generate Reports** |
| * Create Maintenance Request Details Report |
| * Create Maintenance Status Report |
| **Show Dashboard** |

DELIVERABLE 2

# DELIVERABLE #2

Deliverable #2 will consist of your outputs for the first sprint. Document your progress following the below structure. In addition, you will also need to submit a demo of your working prototype demo (in its current state):

## Structure of Sprint

Each use case or user story for the sprint is analyzed in more detail to understand the requirements (see [http://agilevideos.com/videos/agile-simulation-part-16-iteration-pre-](http://agilevideos.com/videos/agile-simulation-part-16-iteration-pre-planning-backlog-grooming/) [planningbacklog-grooming/).](http://agilevideos.com/videos/agile-simulation-part-16-iteration-pre-planning-backlog-grooming/)

#### Prioritized Product Backlog

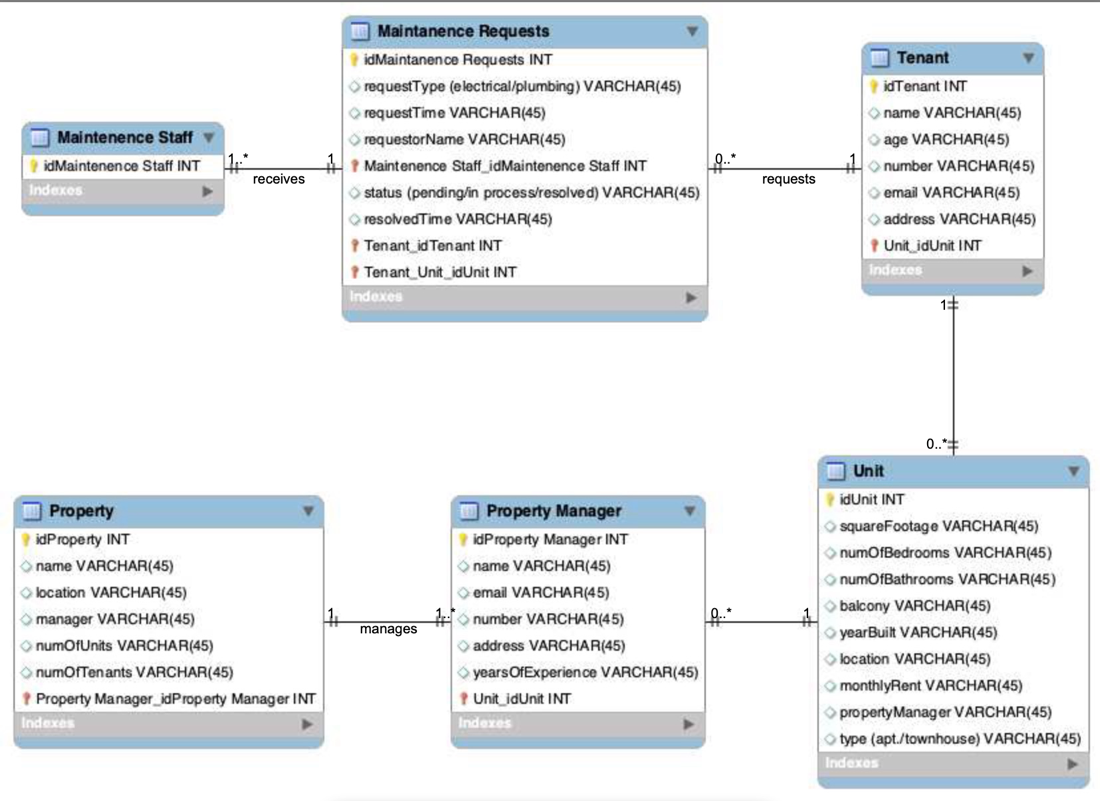
Include your product backlog which shows in different colors the use cases included in each sprint. Highlight the use cases that are included in the current sprint. If a use case has multiple user stories, include these in the table under the use case. ***Each sprint should use a different color – so we can tell at a glance what is being done in the current sprint, what has been done in the previous sprint, and what will be done in the next sprint***.

|  |  |  |
| --- | --- | --- |
| **Use Case** | **Priority** | **Acceptance Criteria** |
| **Add Properties/Units**  As a property manager I want to manage properties by adding new properties and their details to the system. | **Must-Have** | - must be able to update a property  - a property can consist of multiple units  - must be able to create new property  - must be able to update a property  - must be able to delete a property (an associated units) |
| **Assign Property Managers** | **Must-Have** | * Show attributes of manager * Allow manager to manage one property * Allow manager to oversee multiple units within property |
| **Add Tenants** | **Must-Have** | * Be able to create new tenants * Be able to assign tenants to unit in a property * Tenant has all attributes * Be able to update tenant information * Be able to delete tenant |
| **Submit Maintenace Request**  As a tenant, I want to be able to submit maintenance requests through a variety of platforms. | **Must-Have** | * Allow tenants to submit maintenance requests by web or email. |
| **Manage Maintenance Request** | **Must-Have** | * Must be able to enter information for maintenance requests * Maintenance requests should have a status of “pending, in-process, or resolved” * Requests should have relevant attributes such as type of request, reason, priority, request origin, status, and tenant name * Should be able to send email notification to respective maintenance personnel. |
| **Display Request Status** | **Must-Have** | * Maintenance Request attributes should be displayed * Email notification should be sent to the right maintenance person |
| **Escalate Request** | **Must-Have** | * If a maintenance request is not resolved in more than 48 hours, escalate it to the system administrator |
| **Generate Reports**  As a system administrator, I want to be able to create reports on the time it takes to resolve maintenance requests and generate property dashboards. | **Must-Have** | * Create a report of the maintenance requests information such as the date requested and request type |
| **Show Dashboard** | **Must-Have** | * Show dashboards of generated reports |

#### An updated Use Case Diagram, brief Use Case Description table, and Domain Class Diagram IF ANY CHANGES HAVE BEEN MADE

Remember, the iterative nature of analysis and design and the process of discovery in analysis often necessitate that we revise some of our initial documents. Include this ONLY if you have made changes.

**RentoMax Domain Class Diagram**



#### Starting Sprint Backlog

Before starting each sprint, the team should develop a detailed Sprint Backlog – a task list showing all steps needed to produce a working prototype, who is responsible for each task and how long each task is expected to take. The task list should include work required to produce the artifacts listed below. In some cases, however, additional work will be needed to define the prototype and these tasks should be included in the list (see [http://agilevideos.com/videos/agilesimulation-part-17-sprint-planning-meeting/).](http://agilevideos.com/videos/agile-simulation-part-17-sprint-planning-meeting/) Please note that the number of tasks will depend on complexity of the user stories in the sprint. Very simple user stories may not require decomposition into tasks.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Use Case | Task Owner | Status | Sprint | Estimated Effort (hr) |
| 1. Add Properties/Units |  | In progress | 1 |  |
| 1.1 Create Properties | Shreeya | In progress | 1 | 5 |
| 1.2 Create Units | Valerie | In progress | 1 | 5 |
| 1.3 Make relationships between Units and Properties | Maddie | In progress | 1 | 5 |
| 2. Assign Property Manager |  | In progress | 1 |  |
| 2.1 Create Property Mangers | Christina | In progress | 1 | 5 |
| 2.2 Create Relationship between property manager and property | Clara | In progress | 1 | 5 |
| 3. Add Tenants |  | In progress | 1 |  |
| 3.1 Create Tenants | Shreeya | In progress | 1 | 5 |
| 3.2 Assign Tenant to Unit/Property | Valerie | In progress | 1 | 5 |
| 4. Submit Maintenance Requests |  | In progress | 1 |  |
| 4.1 Make request form (Type) | Simran | In progress | 1 | 5 |
| 5. Manage Maintenance Requests |  | Not started | 2 |  |
| 5.1 Assign to appropriate staff | Christina | Not started | 2 | 5 |
| 6. Display Request Status |  | Not started | 2 |  |
| 6.1 Create request status | Clara | Not started | 2 | 5 |
| 6.2 Create email notification for staff | Simran | Not started | 2 | 5 |
| 7. Test & Debug |  | Not started | 2 |  |
| 7.1 Check the functionality of the current prototype | Valerie |  | 2 | 5 |
| 8. Escalate Request |  | Not started | 2 |  |
| 8.1 Automate escalation after 48 hours if status unchanged | Maddie | Not started | 2 | 5 |
| 8.2 Email notification for system administrator | Christina | Not started | 2 | 5 |
| 9. Generate Reports |  | Not started | 2 |  |
| 9.1 Automate report for tenant by unit type | Simran | Not started | 2 | 10 |
| 9.2 Automate report for tenant by rental status | Shreeya | Not started | 2 | 10 |
| 9.3 Automate report of unit information by property | Valerie | Not started | 2 | 10 |
| 9.4 Automate report of maintenance request by request type | Maddie | Not started | 2 | 10 |
| 9.5 Automate report of case information by property / unit | Christina | Not started | 2 | 10 |
| 10. Show Dashboard |  | Not started | 2 |  |
| 10.1 Automate dashboard of all reports | Clara | Not started | 2 | 5 |
| 11. Test for functionality |  | Not started | 2 |  |
| 11.1 Submit form and perform a demo | Valerie | Not started | 2 | 5 |

#### A Brief Description of all User Stories Developed in the Sprint

#### As a property manager I want to manage properties by adding new properties and their details to the system.

#### Have a system that will add all attributes of properties, tenants, and units to easily manage RentoMax

#### As a tenant, I want to be able to submit maintenance requests through a variety of platforms.

#### Have a system that effectively accepts maintenance requests and assigns them to the correct staff member with updating statuses.

#### As a system administrator, I want to be able to create reports on the time it takes to resolve maintenance requests and generate property dashboards.

#### Have a system that will effectively automatically create a dashboard from reports to have data that will help RentoMax.

* + 1. **Acceptance Criteria for each User Story in the Sprint**

You will be given a set of acceptance criteria for each use case. These provide the definition of “done” for the use case/user story. These are things that the product owner will verify to know that the use case is done correctly and completely. Combine 2.1.4 and 2.1.5 in a single table with four columns: user story name, brief description, acceptance criteria, and whether each acceptance criterion was met or not. If the acceptance criterion was not met in your sprint, discuss why and discuss future plans to address it.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Brief Description** | **Acceptance Criteria** | **Met** |
| **Add Properties/Units**  As a property manager I want to manage properties by adding new properties and their details to the system. | **Able to add properties and unit to the system** | - must be able to update a property  - a property can consist of multiple units  - must be able to create new property  - must be able to update a property  - must be able to delete a property (an associated units) | - |
| **Assign Property Managers** | **Assign property managers to property/units in system** | * Show attributes of manager * Allow manager to manage one property * Allow manager to oversee multiple units within property |  |
| **Add Tenants** | **Add tenants to property/units in the system** | * Be able to create new tenants * Be able to assign tenants to unit in a property * Tenant has all attributes * Be able to update tenant information * Be able to delete tenant |  |
| **Submit Maintenace Request**  As a tenant, I want to be able to submit maintenance requests through a variety of platforms. | **Tenants are able to submit maintenance requests through different platforms** | * Allow tenants to submit maintenance requests by web or email. |  |
| **Manage Maintenance Request** | **Maintenance requests have statuses and notifications** | * Must be able to enter information for maintenance requests * Maintenance requests should have a status of “pending, in-process, or resolved” * Requests should have relevant attributes such as type of request, reason, priority, request origin, status, and tenant name * Should be able to send email notification to respective maintenance personnel. |  |
| **Display Request Status** | **The maintenance attributes are displayed and notifications are sent** | * Maintenance Request attributes should be displayed * Email notification should be sent to the right maintenance person |  |
| **Escalate Request** | **Maintenance requests are able to be escalated if not resolved** | * If a maintenance request is not resolved in more than 48 hours, escalate it to the system administrator |  |
| **Generate Reports**  As a system administrator, I want to be able to create reports on the time it takes to resolve maintenance requests and generate property dashboards. | **Reports are generated** | * Create a report of the maintenance requests information such as the date requested and request type |  |
| **Show Dashboard** | **Report dashboards are shown** | * Show dashboards of generated reports |  |

#### Sprint Review - Tutorial Screencast

Sprint Review involves demonstrating the functionality of your prototype (i.e., the use cases developed in the current sprint) to the product owner, project sponsor, management, etc. (see [http://agilevideos.com/videos/agile-simulation-part-22-demo/).](http://agilevideos.com/videos/agile-simulation-part-22-demo/) Use any web-based tool (e.g., screencast.com, screencast-o-matic.com, or youtube) to develop your prototype

demonstration. During the sprint review, in addition to demonstrating the functionality you have developed and showing how the user stories you have developed meet acceptance criteria, you also need to assess project progress against the sprint goals determined during the sprint planning meeting and the sprint backlog (that is, have you finished with all of your user stories for the sprint, have you met all acceptance criteria for each user story, are you ahead, are you behind?

Why?). Include the URL for the presentation in your deliverable.

Demo is submitted separately in assignments.

#### Sprint Retrospective:

**(source:** [**http://www.mountaingoatsoftware.com/agile/scrum/sprint-retrospective/)**](http://www.mountaingoatsoftware.com/agile/scrum/sprint-retrospective/))

No matter how good a Scrum team is, there is always opportunity to improve. Although a good Scrum team will be constantly looking for improvement opportunities, the team should set aside a brief, dedicated period at the end of each sprint to deliberately reflect on how they are doing and to find ways to improve. This occurs during the sprint retrospective.

The sprint retrospective is usually the last thing done in a sprint. Many teams will do it immediately after the sprint review. The entire team, including the Scrum Master should participate. Although there are many ways to conduct an agile sprint retrospective, our recommendation is to conduct it as a start-stop-continue meeting. This is perhaps the simplest, but often the most effective way to conduct a retrospective. Using this approach each team member is asked to identify specific things that the team should:

* + - * Start doing
      * Stop doing
      * Continue doing

There are many variations on this simple format. The Scrum Master can facilitate this sprint retrospective meeting by asking everyone to just shout out ideas during the scrum. The Scrum Master can go around the room asking each person to identify any one thing to start, stop or continue. Or, for example, he or she can tell everyone to focus on identifying something to stop this time because not much attention has been paid to things to stop in recent retrospectives.

After an initial list of ideas has been brainstormed, teams will commonly vote on specific items to focus on during the coming sprint. At the end of the sprint, the next retrospective is often begun by reviewing the list of things selected for attention in the prior sprint retrospective.

**What to turn in for your Sprint Retrospective:** A write-up of results of sprint retrospective: that is, what has been working well in your team and you would like to continue doing it; what issues/problems you faced and the reasons for these problems, and as a result, what should you change (start doing and stop doing) (see [http://agilevideos.com/videos/agilesimulation-part-23-](http://agilevideos.com/videos/agile-simulation-part-23-sprint-retrospective/) [sprint-retrospective/).](http://agilevideos.com/videos/agile-simulation-part-23-sprint-retrospective/)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| What Went Well | What to Improve | Problems | Reasons | Changes |
| Communication | Communication of time and schedules for meeting times | People’s availability was not lining up with a possible meeting | Communication is key to collaborating on the application development. | Try to eliminate unnecessary meetings, and ensure all participants are in attendance |
| Timeliness of Work | Everyone’s work should be submitted ahead of time. | Getting our work in on time is crucial to ensuring timely delivery of the project. | Frustration and stress rise when delegated tasks are not completed. | Experimenting with programming in advance of the team’s due date. |
| Testing of the system | Testing should be done throughout the development of the application, if possible, after every step. | The testing of the system poses problems, as bugs and errors are less likely to be identified at the end of a sprint. | Testing is crucial in saving time and minimizing wasted effort. | Try to test as frequently as possible, saving many cases relating to maintenance  requests. Follow each request as it flows through the system, and make sure it is functional. |
| Design & Identify Functionality | Clarifying priorities is important to designing the system. | Having unclear priorities led team members to incorrectly estimate the scope of the project. | Clarify in-depth use cases on the functionality of the system. | Make sure each step is thoroughly documented with descriptive use cases. |
| Collaboration and Willingness to Work | Too many in-sprint bugs being found in | People must come prepared for meetings to ensure productivity. | To efficiently and effectively create a system that everyone understands | Everyone come to project meeting in person |
| Understanding of Salesforce | Creating Relationships and Rules | Lack of understanding in Salesforce | Not enough practice | Trial and error |
| Technical Background of Salesforce | Reviewing items we have learned before in-class or through trailmix activities | Not having enough experience | Not having a background in Salesforce | More practice |

DELIVERABLE 3

# DELIVERABLE #3 (FINAL PROJECT DELIVERABLE)

Deliverable #3 will consist of your outputs for the first and second sprints separately. This final project deliverable should be a cumulative document that also includes your prior two deliverables. In addition, the final project deliverable should also contain the following:

## Final Project Details

#### Table of Contents

Deliverable 1......................................................................................................................................4

1.1. Project Team Organization............................................................................................5

1.1.1. SCRUM Team Biography.............................................................................5

1.1.2. SCRUM Team Biography & Team Member Roles......................................6

1.2. System Overview and Visioning...................................................................................7

1.2.1. Client Organization Background...................................................................7

1.2.2. System Vision Document..............................................................................9

1.2.3. For all business benefits identified above, come up with conditions of satisfaction.............................................................................................................10

1.2.4. Use the product-box technique to describe the vision for your system.......10

1.3. System Analysis...........................................................................................................12

1.3.1. Information Gathering Techniques..............................................................12

1.3.2. Process Requirements..................................................................................12

1.3.3. Use Case Diagram........................................................................................13

1.3.4. Brief Descriptions of your Use Cases..........................................................14

1.3.5. User Interface Flow......................................................................................14

1.3.6. Domain Class Diagram................................................................................16

1.3.7. CRUD Matrix...............................................................................................17

1.4. Identification of Prototype User Stories.......................................................................18

1.4.1. Break your Use Cases into User Stories.......................................................18

1.4.2. Use Case Description...................................................................................18

1.4.3. Prioritize User Stories..................................................................................19

1.4.4. Estimate Effort.............................................................................................19

1.4.5. Release Planning: Plan your Sprints............................................................20

Deliverable 2....................................................................................................................................21

2.1. Structure of Sprint........................................................................................................22

2.1.1. Prioritized Product Backlog.........................................................................22

2.1.2. Updated Use Case Diagram, brief Use Case Description table, and Domain Class Diagram........................................................................................................24

2.1.3. Starting Sprint Backlog................................................................................24

2.1.4. A Brief Description of all User Stories Developed in the Sprint.................27

2.1.5. Acceptance Criteria for each User Story in the Sprint.................................27

2.1.6. Sprint Review – Tutorial Screencast............................................................29

2.1.7. Sprint Retrospective.....................................................................................30

Deliverable 3....................................................................................................................................32

3.1. Final Project Details.....................................................................................................33

3.1.1. Table of Contents.........................................................................................33

3.1.2. Executive Summary.....................................................................................34

3.1.3. Use Case Diagram and Domain Class Diagram..........................................35

3.1.4. System Prototype (Demo)............................................................................36

3.1.5. User Manual.................................................................................................38

3.1.6. Screenshots...................................................................................................38

3.1.7. Project Evaluation........................................................................................42

3.1.8. Project Retrospective: Lessons Learned......................................................42

4. Project Presentation Guidelines...................................................................................................44

* + - 1. **Executive Summary: An Expanded Version of your System Vision Statement** The one-page executive summary should have the following (it is an expanded version of your system vision statement):

1. One brief paragraph explaining the system you are developing and business problems that the system addresses and key benefits provided by your system. Make a compelling case for your system.

RentoMax, a leading apartment rental company in Atlanta, currently has manual, paper processes in place, and they are hindering efficiency and customer satisfaction. To optimize its operations, our team created a Salesforce platform that would improve operational efficiency, cut costs and enhance overall customer satisfaction. This transformative solution aims to automate key business processes such as maintenance requests and escalations, request status, and a tenant tracking system. Additionally, the platform will include information about the properties themselves, such as the number of bedrooms and bathrooms, square footage, etc.

By adopting our innovative Salesforce-based system, RentoMax can effectively tackle these pressing issues and unlock significant advantages compared to its competitors. Through improved communication and information handling, RentoMax can greatly enhance organizational efficiency and mitigate the risks associated with manual operations. Moreover, our system centralizes maintenance request management, allowing tenants to submit issues through various channels and automatically directing them to the appropriate personnel. With features such as status tracking and escalation protocols for unresolved matters, RentoMax ensures prompt resolution for higher tenant satisfaction. The reporting capabilities and a comprehensive dashboard offer insights into maintenance performance, facilitating targeted improvements and overall progress tracking. By investing in our advanced system, RentoMax stands to not only boost operational efficiency but also cultivate stronger tenant relationships and satisfaction. This, in turn, can lead to increased tenant retention rates, minimized vacancies, and improved financial performance. Embracing our cutting-edge solution enables RentoMax to maintain competitiveness in the market, foster growth, and solidify its position as a leader in the industry.

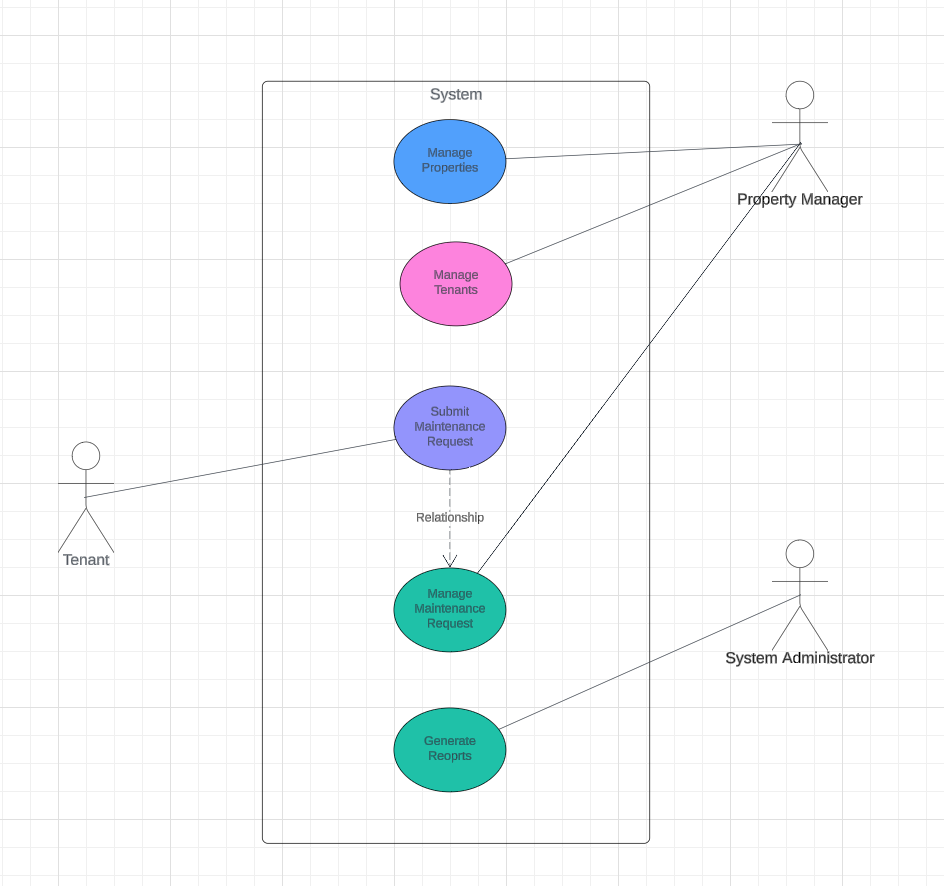
1. One paragraph briefly summarizing the key functionality of your system. What can your system do and how does it address key business needs?

Our advanced Salesforce-based system offers a solution designed to meet the fundamental needs of RentoMax by automating and centralizing property and tenant management procedures. Key features of the system include effortless integration for adding and storing property and tenant data, as well as efficient handling of maintenance requests. By enabling tenants to submit requests through various channels and automatically directing them to specialized staff based on the type of issue, the system ensures timely resolutions and escalations as required. Furthermore, it provides RentoMax with reporting capabilities and a comprehensive dashboard view, furnishing valuable insights into both pending and completed maintenance requests. By optimizing communication channels, enhancing operational efficiency, and elevating tenant satisfaction, our system effectively addresses the challenges associated with manual, paper processes.

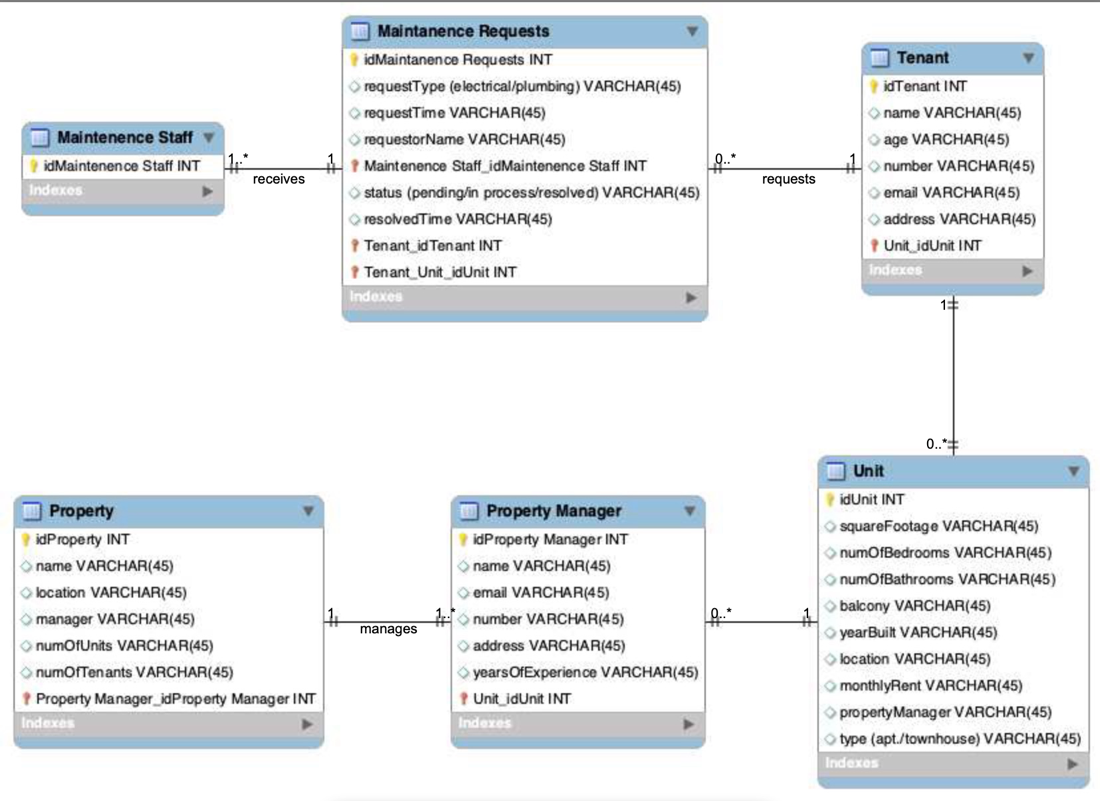
#### Use Case Diagram and Domain Class Diagram

The Use Case Diagram and Domain Class Diagram for your system. Make sure that these correspond to the functionality provided by your actual system.

Use Case Diagram



Domain Class Diagram



#### System Prototype (Demo)

Provide the URL link to your prototype and any information needed to access it (e.g., user name and password). Points will be deducted if I cannot access your prototype. Also create a demo of your prototype and provide the URL for the web-accessible demonstration of your prototype.

<https://universityofgeorgia269-dev-ed.develop.lightning.force.com/lightning/o/Case/list?filterName=Recent>

Username: [vp53834@uga.sandbox](mailto:vp53834@uga.sandbox)

Password: ValePena2002#

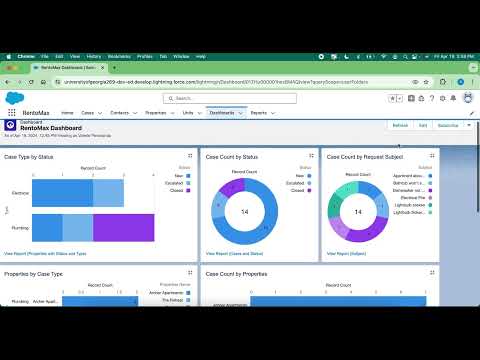
Test the prototype by giving it to a couple of your friends who are not technically inclined and see if they can operate it without problems. Also test if the prototype will run with various web browsers. The prototype should run on any reasonable browser.

Reports should look professional and neat and should include example data that is sensible to your users. Reports should include appropriate headings for all columns. Your goal for the reports is to make sure that if someone picks up a report, he will be able to read for himself what the report is.

The prototype will be graded on its appearance (30 points), ease of use (40 points), and extent to which it accomplishes the logic of the use cases that you were assigned to prototype (100 points).

In recording the video that demonstrates your prototype, put yourselves in the stakeholders’ shoes and tell a story with your prototype. That is, present your use cases in a logical manner in terms of how they help carry out a business process or function (from beginning to end).

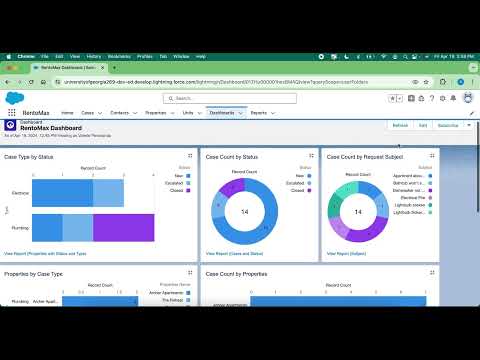
Demo link: <https://youtu.be/sKcijIKA8BI>

[](https://youtu.be/sKcijIKA8BI)

#### User Manual (Video)

Create a user manual (video) where you instruct the user on how to use the system. The user manual should include step-by-step instructions (everything from “go to website …” onward). A great way of doing this is through a video. This can be the same or different from the demonstration of your prototype. Remember: The objective of the demo video is to demonstrate the functionality of your system and it is limited to 5 minutes. A training video needs to provide detailed step by step instructions on how to use your system and may be much longer and much more detailed than your demo. If your demo can serve both purposes, then that’s fine. Provide the link to your demo. If not, provide the link to your training video.

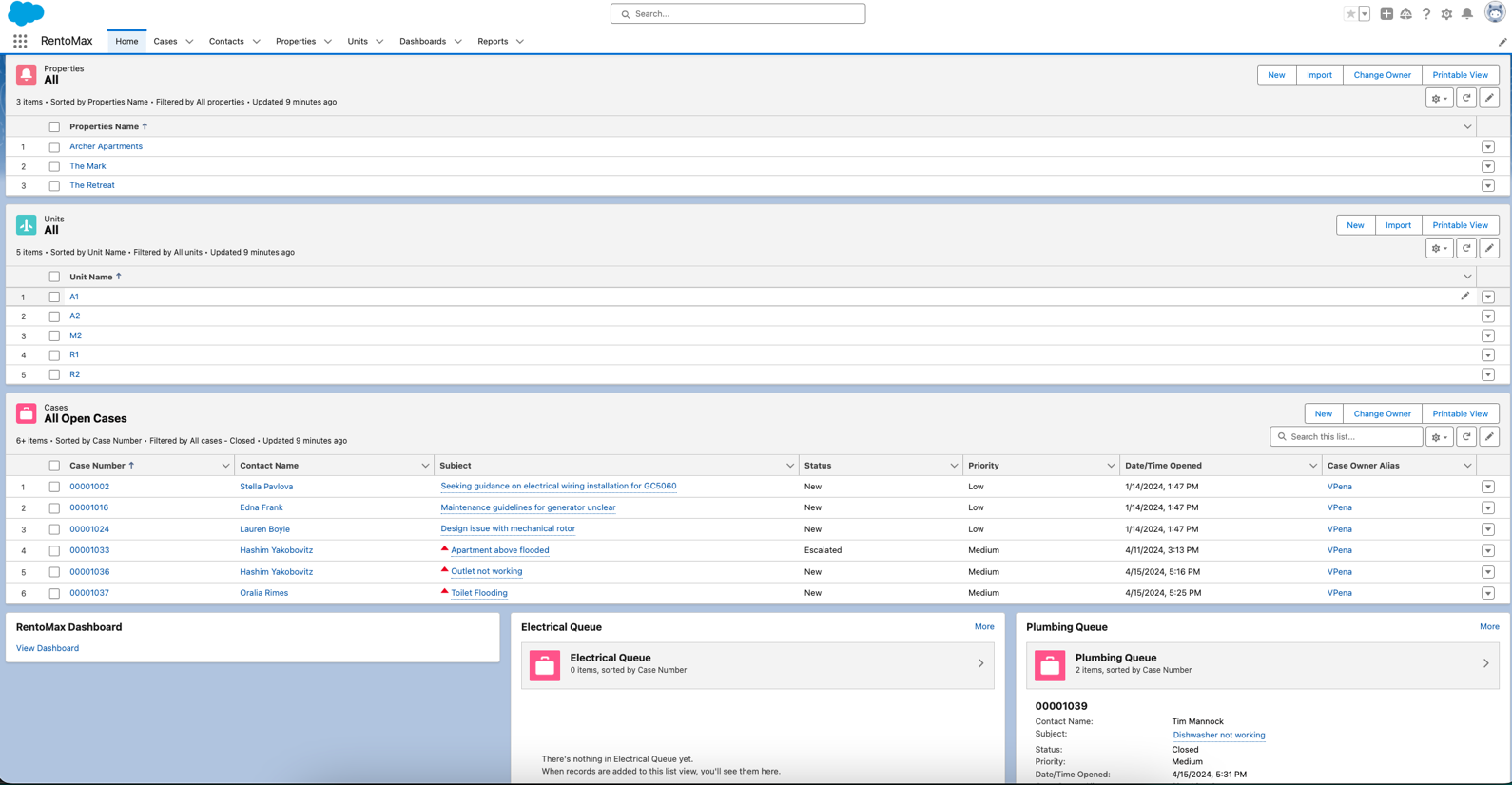
User Manual Link: <https://youtu.be/sKcijIKA8BI>

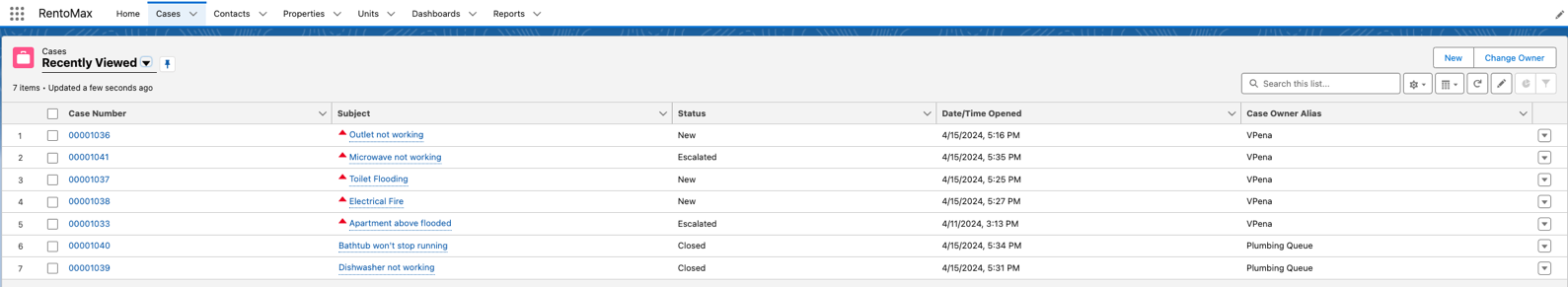
[](https://youtu.be/sKcijIKA8BI)

#### Screenshots

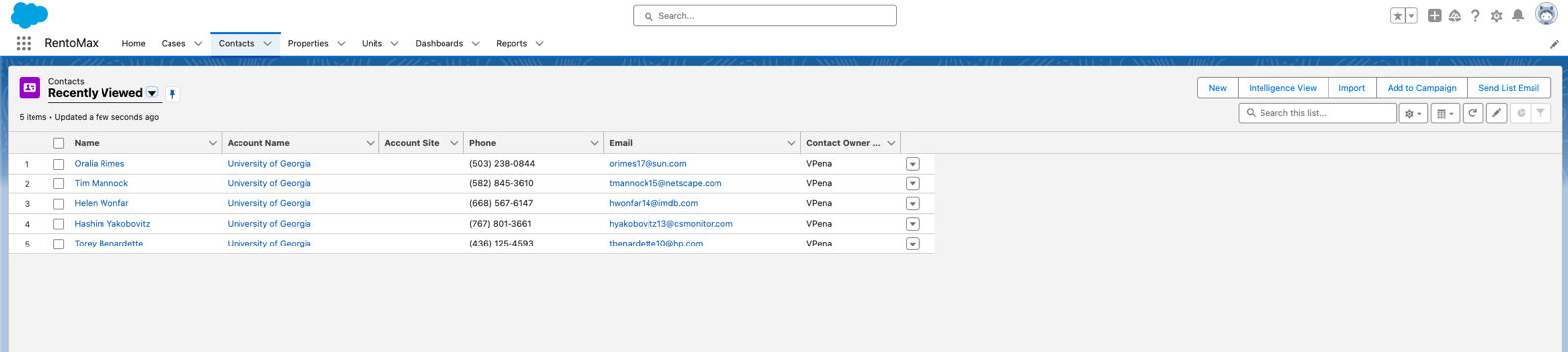
Provide screenshots of screens in the prototype. Also provide a sample print of each report (either the printed report or a screenshot of the report). Organize and label these.

Home Page

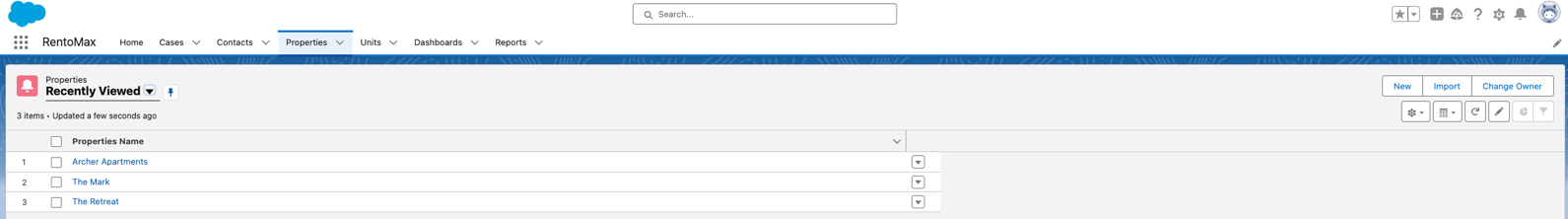
Cases Page



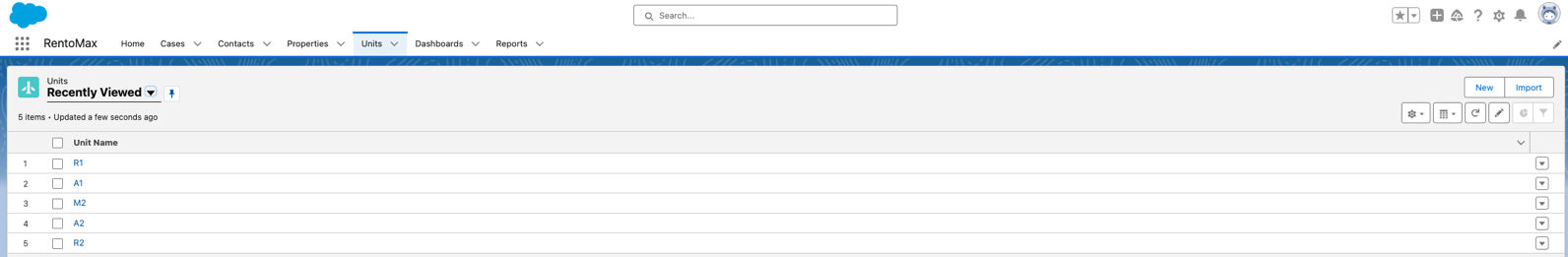
Contacts Page

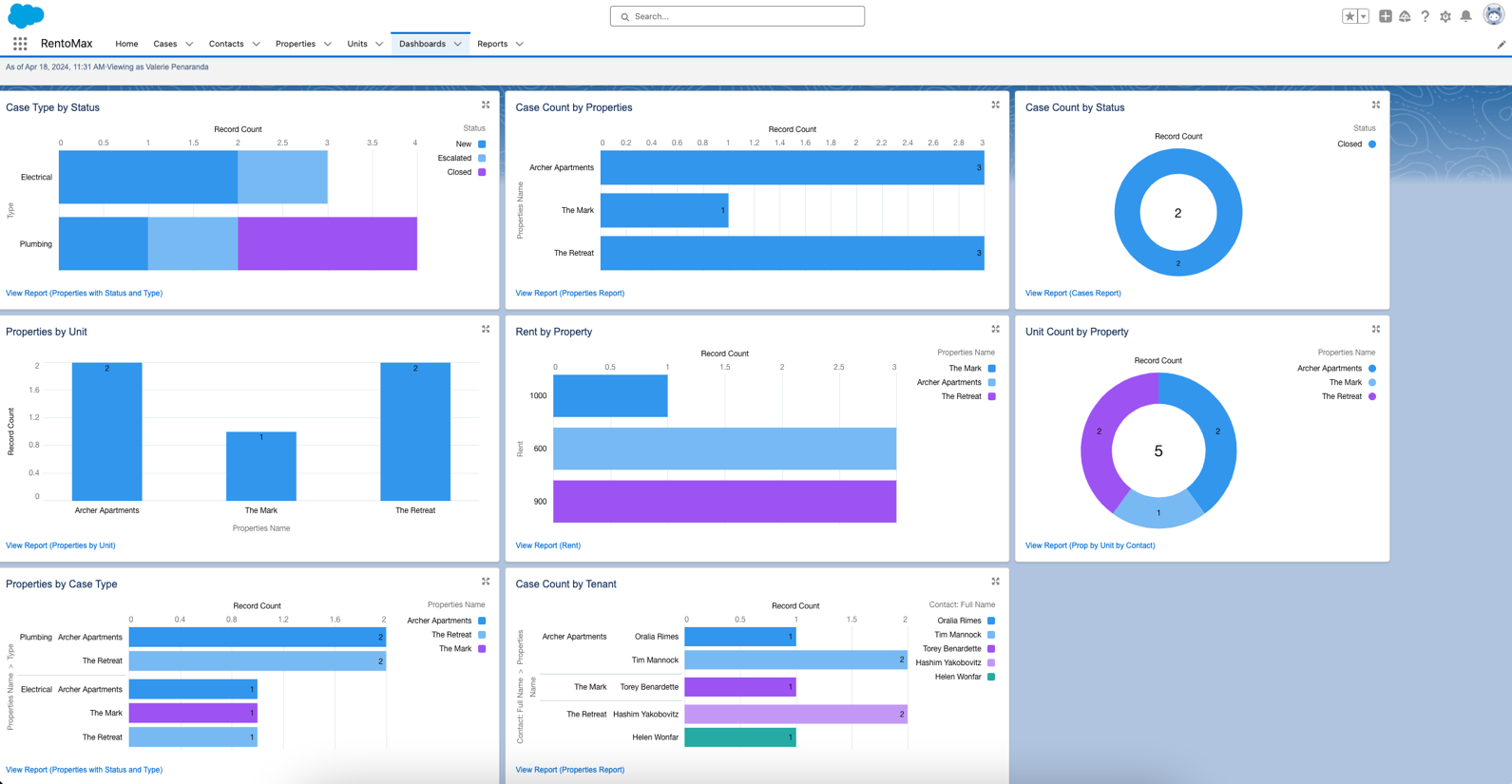


Properties Page



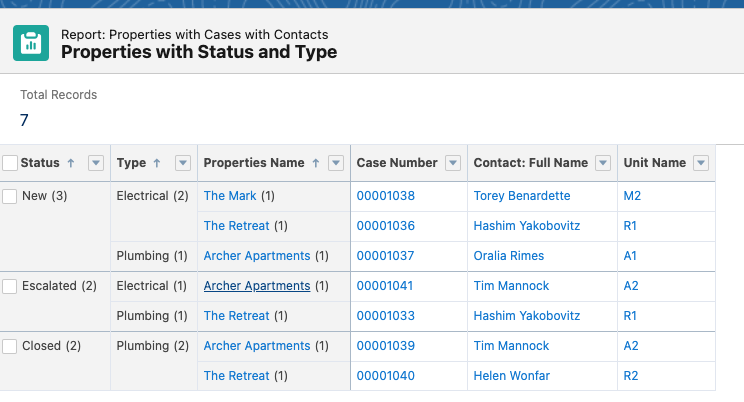
Units Page

Dashboard

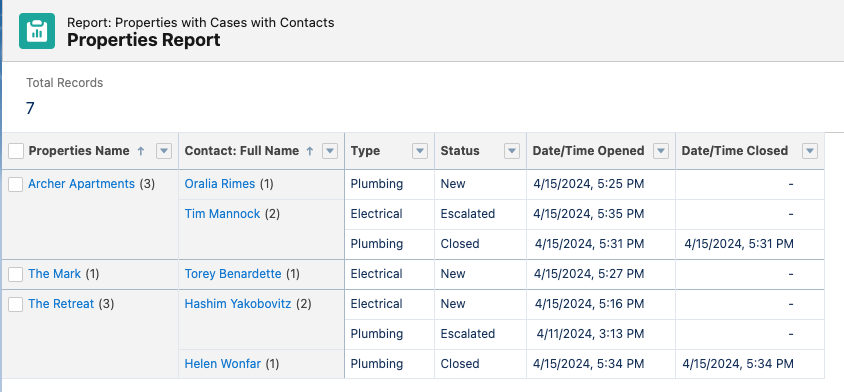


Reports:

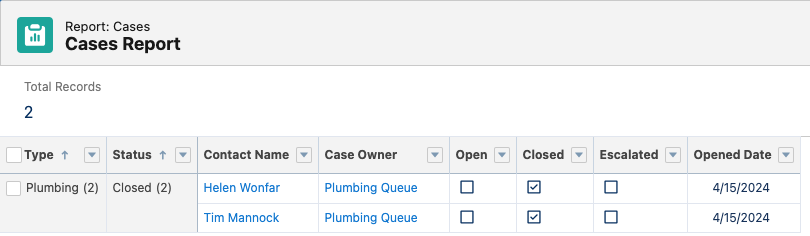
Properties with Status and Type



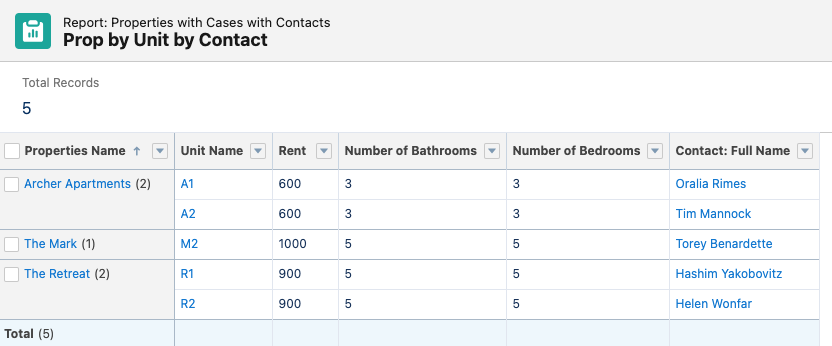
Properties Report



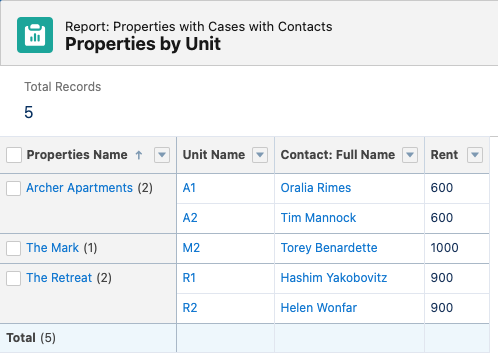
Cases Report



Properties by Unit by Contact



Properties by Unit



#### Project Evaluation

Provide an honest evaluation of your system. What are you proud of? What can be improved? What would you have done differently in retrospect? Which requirements have you delivered? Which not? Why? What’s went well? What did not go well as it should?

Include any feedback from potential users of the system on your final prototype.

Reflecting on the overall system, as a group we are all proud of the system's functionality we created. The system can create, add, and delete tenants, units, and properties. The system runs smoothly for the features stated but has room for improvement. We were able to escalate the cases have it be immediately assigned to system administrator. We were also able to assign users to the appropriate queues. Finally, we were able create a custom report to show properties to units to case. Next, we could improve the assignment of users by having more salesforce licenses and user interface if we had more time. At the beginning of building the application creating the custom objects, relationships, and rules went smoothly. Lastly, there were two major problems that we ran into. The first problem was getting the close case option to show up in the status picklist. The second problem we ran into was creating a report that we could properly use in the dashboard.

#### Project Retrospective: Lessons Learned

One of the things we don’t do enough in systems development is project assessment. Therefore, I would like you to reflect upon “Lessons Learned” from the project and discuss those. These can be in terms of learning various aspects of SA&D (e.g., how requirements can evolve, how to manage scope creep, difficulty in communicating with clients, usefulness of use case diagrams, not properly eliciting requirements or assessing technical feasibility up front, etc) or how to work better in teams, or project management. What do you think worked effectively? What do you think you would change next time and how?

A key lesson that our team learned was the importance of clearing mapping out our priorities and visions. As different individuals, we each had a different vision and ideas on improvements to make for the RentoMax system, but we learned to discuss the pros and cons of each approach, to make a decision that well-suits the best team and has the well-being of the company in mind. Continuing to exchange ideas really allowed us to work more efficiently as a team, and it avoided any hiccups in terms of miscommunications. Another valuable lesson we learned was always communicating with the client and higher-ups, to ensure we were doing everything according to plan. Anytime anybody in the group had a question for the client, or something they wanted to clear up, it was never left unspoken. We kept the client frequently updated by providing deliverables and meeting our own timeline goals before the final launch. We realized that keeping the client updated was important because each time we updated the client, we were provided with feedback on which we could improve on. This approach ensured the client’s vision aligned with the team’s. In terms of efficiency within our own team, we found that establishing clear roles and responsibilities, along with deadlines, greatly contributed to our overall success. We met frequently, even when there were no deadlines coming up, to make sure we were being as efficient as possible. Additionally, we created a team friendship and mutual respect - which ultimately improved our efficiency as well, given we always wanted to provide an extra hand when somebody needed it. By creating a comfortable working environment, we were all motivated to get the job done. To conclude, the lessons we learned from the project truly uncover the importance of being clear and concise, teamwork, and methods of project management. The lessons we have learned will benefit us all in future projects, and we will continue to improve individually to be better employees!

# PROJECT PRESENTATION GUIDELINES

How did the team use the topics discussed in the class to produce the prototype? The presentation should briefly review the context of the project and the business case for it, the various models used to analyze the system, the prototype, and any lessons learned. To placate the demo gods – who always spoil live demos, the team should prepare a brief screencast showing the prototype functionality. The following Project Presentation Template should be used:

## Template for Project Presentation

#### Presentation Time: Approximately 12 minutes maximum (strictly enforced) (Exact time limit will be announced closer to the presentation date)!

1. **Introduce your team**. State the various roles each one of you played (e.g. business analyst, programmer, user experience expert, etc.)
2. **Present the system from a business perspective.** What is the business need that the system addresses? What are the benefits to the business/organization? What are the major pain points and how does your system address these? Be vivid in your description and make a compelling case for the business need and how your system addresses it. E.g., if you are replacing an existing system, show screenshots of the old system and illustrate why it needs redesign (or show a process model of the old process and show why it is inefficient). If you are creating an online ordering system for a restaurant, talk about missing out on business + the time it takes to take phone orders + mistakes people can make in taking orders and how the new system can address all these. If it is a not-for- profit baseball league, talk about the need to communicate information to players and to volunteers and how they have limited resources (1-2 people) to do so, etc. At the end of this, your audience should understand why you need the system from a business perspective.
3. **Talk about your users:** who are they (you should also present your user personas), how did you elicit requirements? From whom? Any issues you had understanding the requirements, getting in touch with the users, etc. How is the system going to impact what they do?
4. **Present the main functionality for the system:** Discuss what functionalities are implemented by the system to address the business need. Discuss what these functionalities are and why they are important vis-à-vis the business needs. Show your *use case diagram*. Talk about any important non-functional requirements. Show which use cases you developed and which ones are left for future sprints.
5. **Analysis and Design techniques:** Talk about how you modeled requirements during analysis and design. Show your *domain class diagram* for the system. Show your *business process model* for the entire system and an *activity diagram for a use case*. State

how you started with your use case diagram and domain class diagram and then understood and modeled the requirements one use case at a time. Talk about how you used Salesforce to implement your system (did you use any coding? Did you have to write queries?).

1. **Demonstrate your system:** Have a short 5-minute video presentation of your system. Make sure you think through how to present the system so that the system helps the user understand how the organization will use the system in their business process (e.g., start with beginning of the business process and show logical progression in terms of use cases).
2. **Feedback from user(s) (if any):** What feedback did you receive from your users? What did they like about your system? Did not like? Suggestions for further improvement?

#### What are YOU proud of? What particular aspects of the system are you particularly proud of? What can you improve upon in the system given more time/resources?

1. **Project Retrospective - Lessons learned:** What did you learn through this project? About *developing systems* and/or analysis & design? About *working in teams*? About project management? etc. What did your team do very well? What did you not do well? How could you have done better? What would you do differently?