**Group 12:**

**Bunch of Coders**

**Scrum Master**

Bobby Buxani

**Team Members**

Chandni Amin

Charlie McCollough

Riley Buckles

Emma Turner

Bobby Buxani

**Table of Contents**

[1.1 Project Team Organization](#_x9i724qrwaug)

[1.1.1 SCRUM Team Biography](#_fs7o76ty5ire)

[1.1.2 SCRUM Skills Matrix](#_sfn2w82g8ozy)

[1.2 System Overview and Visioning](#_g8ur7gh2yysf)

[1.2.1 Client Organization Background](#_po8c9r6w8hy7)

[1.2.2 System Vision Document](#_8q84qcfb5chs)

[1.2.3. For all business benefits identified above, come up with conditions of satisfaction](#_a0sk9464z9i0)

[1.2.4. Use the product-box technique to describe the vision for your system.](#_sxb2meiv66ks)

[1.3 System Analysis](#_fmn4n9j1yyp2)

[1.3.1. Information Gathering Techniques](#_ei61p05xvhza)

[1.3.2. Process Requirements](#_f8iw7pqkpamf)

[1.3.3. Use Case Diagram](#_hmlt6gds2do)

[1.3.4. Brief Descriptions of your Use Cases](#_l5zdvmhhiryh)

[1.3.5. User Interface (UI) Flow](#_tow2q4f21njn)

[1.3.6. Domain Class Diagram (DCD)](#_coc9g7s0afsx)

[1.3.7. CRUD Matrix](#_m7kla3cf61fx)

[1.4 Identification of Prototype User Stories](#_mom9j7otdc)

[1.4.1. Break your Use Cases (Features) into User Stories](#_u9b7l97xkzsy)

[1.4.2. Use Case Description](#_nog1r2eojngp)

[1.4.3. Prioritize User Stories](#_jyye3v64qegn)

[1.4.4. Estimate Effort](#_9jq8e2hjuk5h)

[1.4.5. Release Planning: Plan your Sprints](#_7wozeehqboa)

[2.1. Structure of Sprint](#_4l8ixlsuehp)

[2.1.1 Prioritized Product Backlog](#_mjz6qshhv5nz)

[2.1.2 An updated Use Case Diagram, brief Use Case Description table, and Domain Class Diagram](#_jwjsqqchy9gl)

[Revised Use Case Diagram](#_3otvozf3udwg)

[Revised Use Case Description Table](#_xutsgp7qbh80)

[Revised Domain Class Diagram](#_bp0n47e3t42u)

[2.1.3 Starting Sprint Backlog](#_m3torpbkowmz)

[2.1.4 A Brief Description of all User Stories Developed in the Sprint](#_nwdruk19saca)

[2.1.5 Acceptance Criteria for each User Story in the Sprint](#_e2ba06j2029)

[2.1.6 Sprint Review - Tutorial Screencast](#_ye4njwuy1ems)

[2.1.7 Sprint Retrospective](#_jwy5i13ygt72)

[3.1. Final Project Details](#_wn2kx8nal5b1)

[3.1.2. Executive Summary](#_s1kjd9eyjm1k)

[3.1.3. Use Case Diagram and Domain Class Diagram](#_fyotn0fqiy5b)

[Use Case Diagram](#_la5vkm15a9l0)

[Domain Class Diagram](#_d7cm3l8sy246)

[3.1.4. System Prototype (Demo)](#_qtx7lsgqy0la)

[3.1.5. User Manual (Video)](#_fnqr4aoeulk3)

[3.1.6. Screenshots](#_4fro4xql30cg)

[Properties Screen](#_jgn92snwzxoq)

[Contacts Screen](#_5h8691sfnmce)

[Maintenance Requests Screen](#_8cycsas415e1)

[New Maintenance Request Form](#_j8wp3v908wxb)

[Maintenance Requests by Properties Report](#_h6ecwlvv0hac)

[Maintenance Requests by Type Report](#_tq1gnlw18b80)

[Tenants by Unit Type Report](#_926uzciq2xaz)

[Tenants by Status Report](#_5twy6ncvsdtx)

[Property Report](#_xy6pukzg4r5r)

[Maintence Status Dashboard](#_bx6zeal4e15q)

[Property Dashboard](#_72ebxodwp3xv)

[Tenant Dashboard](#_wmk96mqgy6nv)

[3.1.7. Project Evaluation](#_kdvwnzdkpe04)

[3.1.8. Project Retrospective: Lessons Learned](#_y0a4vrdqlfu6)

**DELIVERABLE #1**

# **1.1 Project Team Organization**

## **1.1.1 SCRUM Team Biography**

**Charlie McCollough**

My name is Charlie McCollough and I am a third year at UGA studying Management Information Systems. My role for this project is primarily as a developer for the Sales Force application and secondly as a business analyst.

I’m from a small town in southeast Georgia called St. Marys which is located on the St. Marys River. The river is the border between Georgia and Florida, so much of my free time is spent on the river fishing in our brackish water. Along with the river on Cumberland Island, I spend time exploring old mansions from the Carnegie era along with backpacking in the south Georgia barrier islands.

In my free time at college, I play many IMs including soccer, tennis, ultimate frisbee, volleyball, and badminton. Outside of Ims I also am a very amateur golfer who strives to hit at least one par every round of golf I play.

I spent my summer being a counselor for a boy's camp in Black Mountain North Carolina. During my time at camp I have spent 9 summers as a camper, and 2 as an employee. During my most recent summer, I was the climbing director, where I took campers and counselors on backpacking trips to crags and taught basic rock climbing techniques and safety on the walls.

**Bobby Buxani**

My name is Bobby (Parvin) Buxani and I am a fourth year at UGA studying Management Information Systems. I am a first-generation student and am about to graduate in May! I have a job lined up after graduation from Ernst and Young as a Technology Consultant in the Atlanta office.

I come from a small town called Cordele, GA. Cordele is called the “Watermelon Capital of the World”. This can be seen in the local Chick-fil-A because the Chick-fil-A has watermelon paintings on the ceiling. Other than that, there is so little to do. My graduating class has 220 people and my whole town has 10,000 people.

In my free time, I love to play pickup basketball at Pound Hall with my roommate and friends. This is probably where most of my free time goes. I also love watching NFL and NBA games. My favorite teams are the Denver Broncos and the Los Angeles Lakers because of my favorite players, Peyton Manning and Lebron James. I am actually going to a Denver Broncos game next year. I also love to work out. My favorite day to work out is Fridays because that is my “Chest and Back” day!

**Emma Turner**

My name is Emma Turner and I am a fourth-year Management Information Systems major at UGA. After graduation, I’ll be working as a Business and Technology Consultant for NTT Data in their Nashville office.

I grew up in Marietta, Georgia in a big family. I’m a triplet with two boys and me, and we also have an older brother. I appreciated growing up in a fun, loud household where we were encouraged to explore our interests and try new things. I also grew up going to our lakehouse in rural Alabama, where I found a love for boating and watersports.

In my free time, I love to spend time listening to music, cooking, and going outdoors. I go to concerts almost every week and some of my favorite artists are Taylor Swift, Brandi Carlile, Tyler Childers, and The 1975. Against all odds, we did in fact get tickets to Taylor’s tour! I also like to hike, camp, and rock climb, and went on my first outdoor climbing trip this weekend. I’m always excited to try new adventures and am going snorkeling with Manatees in February.

**Riley Buckles**

My name is Riley Buckles and I am a third-year Management Information student at UGA. I will be working this summer as a Risk and Financial Advisory intern for Deloitte. Following my internship, I will begin courses for a Master of Accountancy and will graduate in Spring 2024.

I grew up in Peachtree City, Georgia, which is about 45 minutes south of Atlanta. This area is very unique as it is made up of hundreds of miles of golf cart paths that navigate throughout the city. We use golf carts to drive to school, restaurants, shopping areas, etc. It was a really fun place to grow up and I am so thankful for the experiences it gave me.

In my free time, I enjoy playing sports and hanging out with my friends. I played soccer and volleyball in high school, and still enjoy picking up games whenever possible. I am also a member of the Pi Beta Phi sorority. I have met so many friends through this organization and am so grateful for all it has done for me. I am excited for my last year in college and everything beyond!

**Chad Amin**

My name is Chandni Amin, but all my friends call me Chad. I am a fourth-year Management Information Systems student at UGA. Upon graduation, I will be moving to Chicago to work for Deloitte as a Cybersecurity Analyst in their Risk and Financial Advisory sector. This past summer I interned with Deloitte Atlanta as a Cyber and Strategic Risk intern. I had so much fun working on a real project that makes a real impact and I met some of my best friends this summer.

In my free time, I powerlift for UGA and my private team called Team Fueled. After my running career of eight years came to an end I decided to get into lifting and from that I decided to compete in powerlifting. I have been powerlifting for about two years now! In April I will be competing at Collegiate Nationals for USA Powerlifting and will be representing the University of Georgia. Also in my free time I spend a lot of time with my cat named Gary. His full name is Geralt, named after the Witcher, but I call him Gary for short. I always wanted a dog, but I got a cat instead so I trained Gary like a dog. He knows how to sit, goes on walks, is harness trained, plays fetch, and knows a few other commands.

## **1.1.2 SCRUM Skills Matrix**

|  |  |  |
| --- | --- | --- |
| Group Member | Skills | Roles |
| Name: Charlie McCollough  Email: cam70179@uga.edu | Strengths: Reliable, Communication with SCRUM team, technical skills.    Weaknesses: Organizational skills | Primary: Developer  Secondary: Business Analyst |
| Name: Bobby Buxani  Email: prb04927@uga.edu | Strengths: Communication, Organization skills, Great at meeting deadlines, Good at SQL    Weaknesses: Weak technical skills, can be easily distracted, Horrible at Java | Primary: SCRUM Master  Secondary: Database Expert |
| Name: Riley Buckles  Email: rkb89946@uga.edu | Strengths: Organization, time management and planning, detail oriented    Weaknesses: lack of experience in technical field | Primary: Database Expert  Secondary: Client Liaison |
| Name: Chandni Amin (Chad)  Email: cta28330@uga.edu | Strengths: super detail oriented, communication, time management    Weaknesses: weak technical skills, easily distracted | Primary: Business Analyst  Secondary: Junior Developer |
| Name: Emma Turner  Email: eet39061@uga.edu | Strengths: Organization, communication, research, technical skills    Weaknesses: Work prioritization | Primary: User Experience Expert  Secondary: Business Analyst |

**TEAM STRENGTHS:** Good communication, technical proficiency, schedules work together, committed to project.

**TEAM WEAKNESSES:** Heavy outside workload may interfere with project work.

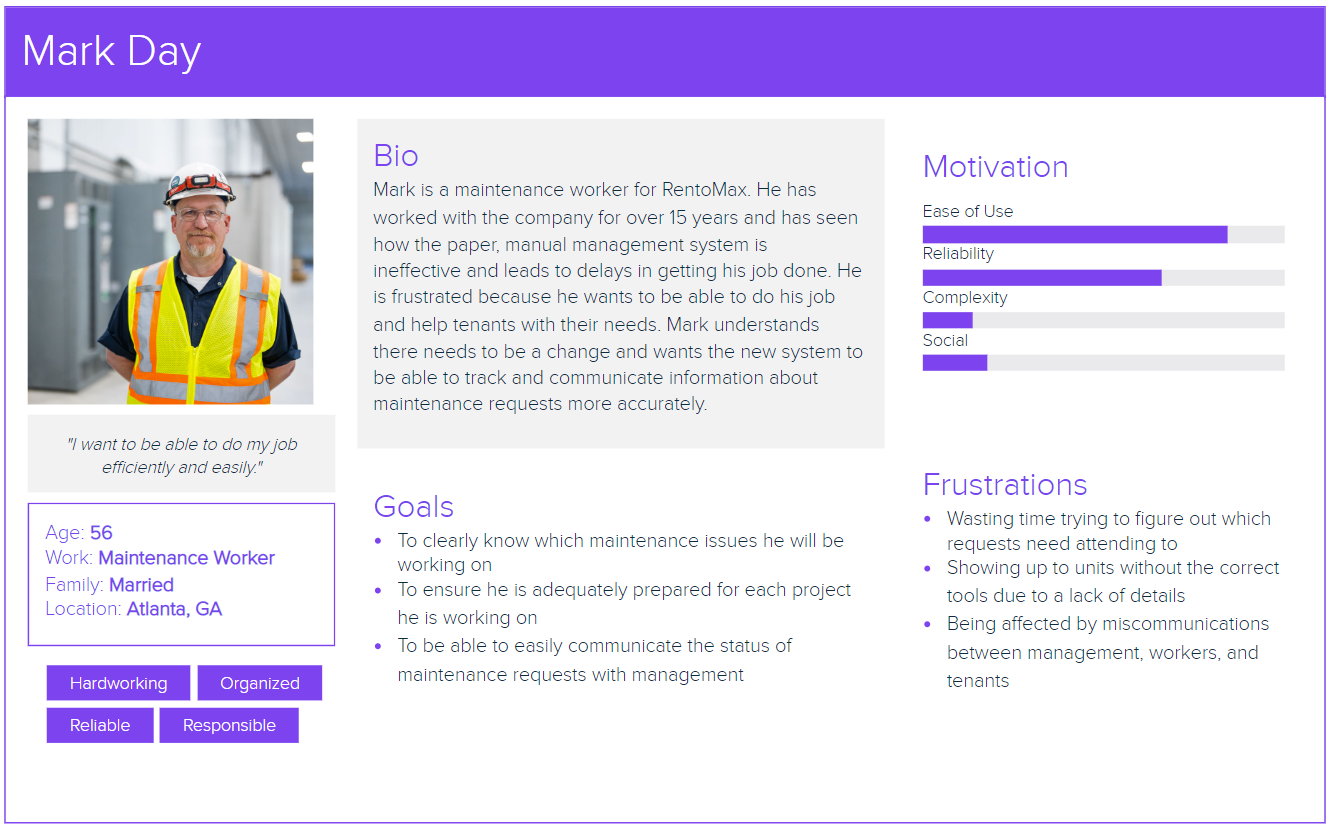
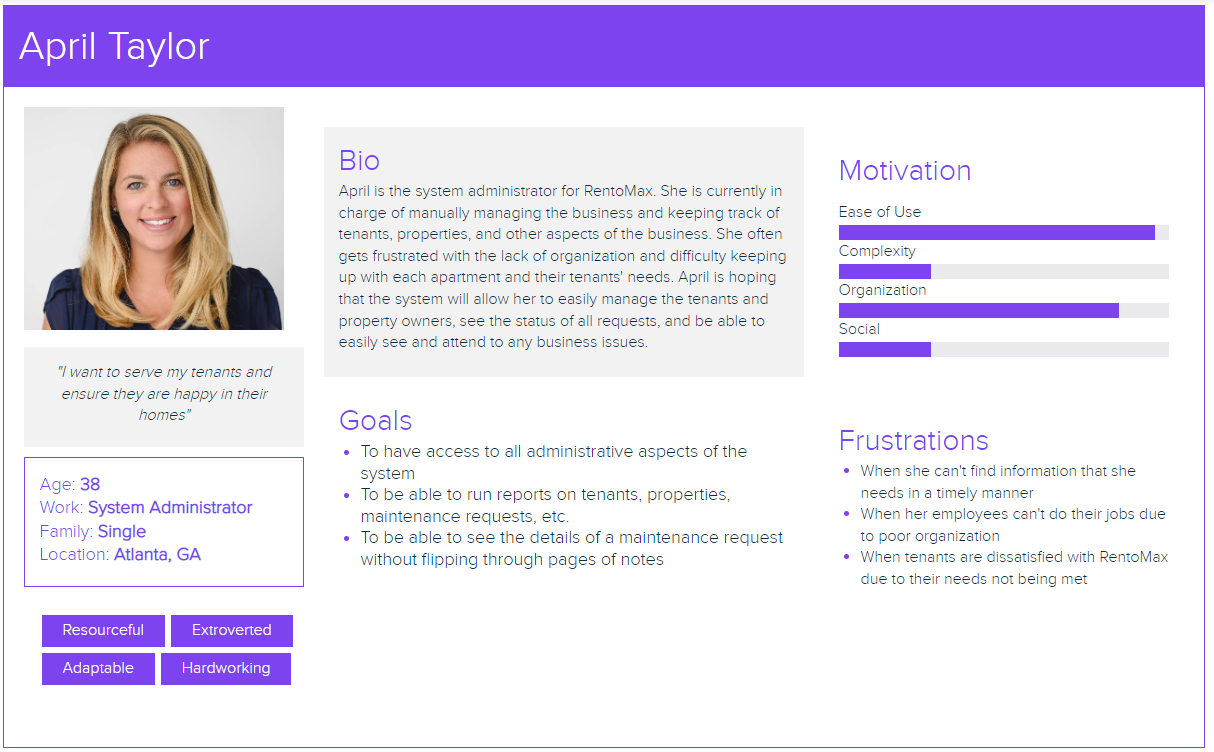
# **1.2 System Overview and Visioning**

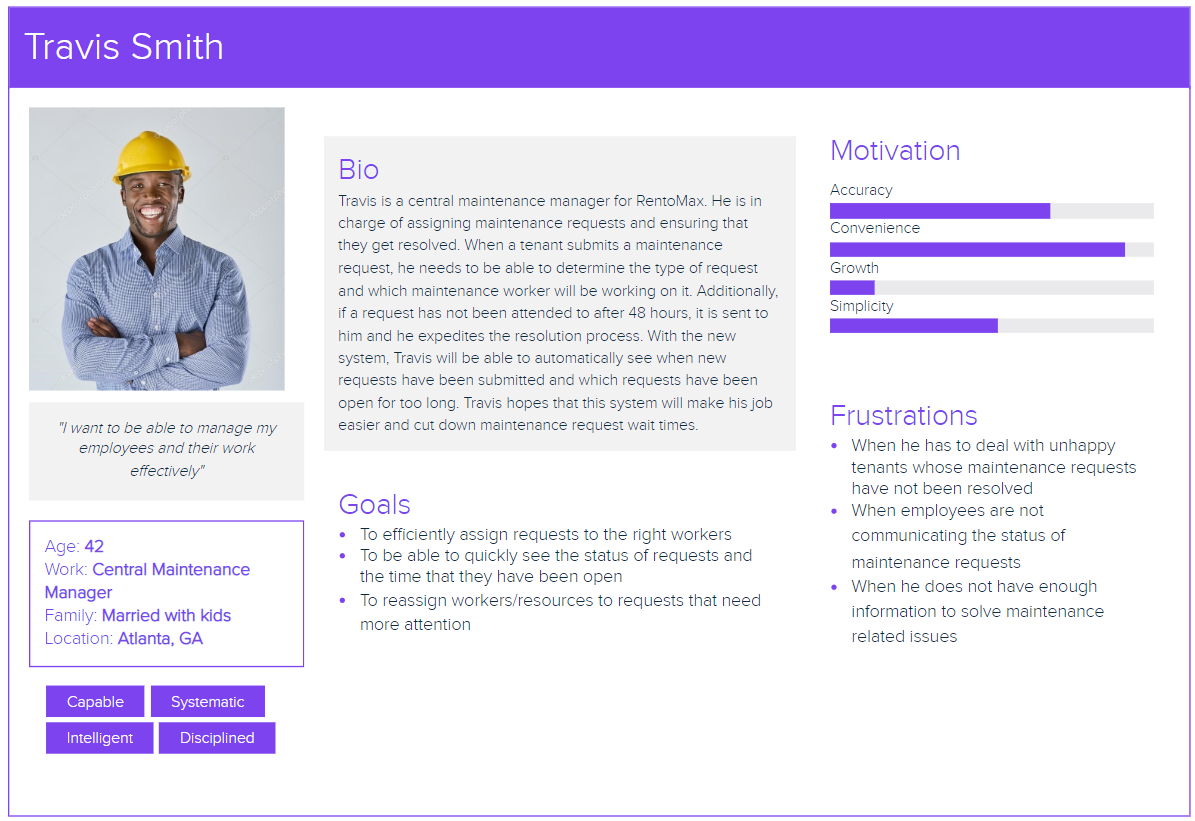
## **1.2.1 Client Organization Background**

* **Description:** What organization/department is the system developed for? Briefly provide a short overview of this organization/department.
  + The system will be developed for RentoMax, an apartment rental company that manages its properties in Atlanta, Georgia. RentoMax currently uses paper to manually track all of the properties, tenants, requests, etc. They are requesting a system to manage all of their properties and any related actions in one place. Specifically, RentoMax wants an easy, convenient way to track maintenance requests. They have 2 types of maintenance requests, electrical or plumbing, and different employees are sent to units depending on the type of request. Additionally, they need to track whether a maintenance request is pending, in progress, or resolved. Our system will aim to generate reports, keep track of request details, and take care of any other actions RentoMax may need.
* **Stakeholders:** Who are the stakeholders and why? Draw a stakeholder   
  context diagram.
  + Property Managers - The property managers are in charge of overseeing an individual RentoMax property and its tenants. The system will help them keep track of which tenants must get notified about maintenance workers entering their units and when this would happen. They will also be able to use the system to generate reports on their tenants and better be able to serve any of their needs.
  + Tenants - The tenants will use the system to submit maintenance requests and receive notifications about those requests. They will be affected by the success of this project as their maintenance requests will be handled more efficiently and the managers will ensure that they get solved in a timely manner with the correct type of worker.
  + Maintenance Workers - The workers will be affected by this project as they will be able to more easily keep track of the units/issues that they have been assigned. They will also be able to change the status of these requests and communicate to the managers when maintenance requests have been completed.
  + Central Maintenance Manager - The central maintenance manager is in charge of assigning any new maintenance requests. When a request gets submitted, they will use the system to determine which type of worker is needed (plumbing or electrical) and which worker is available to resolve the request. Additionally, any requests that have not been addressed after 48 hours are sent to the central maintenance manager to be dealt with. The new system will make it so that once it has been 48 hours, these requests will be automatically sent to the manager and they will be able to quickly address the lapse in response. The central maintenance manager will be able to better complete their job and serve tenants after the implementation of this system.
  + System administrator - The system administrator is in charge of managing all RentoMax properties. They will have access to all administrative aspects of the system. They can use the system to add/remove tenants and assign/change property managers. They will also be able to use the system to generate reports that could be used to see which apartments have the most maintenance issues, which maintenance workers complete work the fastest or most reliably, etc.



* **Users:** Who are the users for the system?
  + The users of this system are the property managers, the central maintenance manager, the system administrator, and maintenance workers. They will be using the system to track and update maintenance requests. The system will also be used to notify tenants of information regarding maintenance requests, however, they will not be direct users of the system.
* Create 2-3 User Personas





## **1.2.2 System Vision Document**

* Problem Description (provide a description of the problem/opportunity the project approvers find compelling; then describe how your solution will address the issue)
  + RentoMax is an apartment rental company that manages its properties across various locations in Atlanta, and they manage all of their processes manually on paper. This especially puts stress on their maintenance processes, where jobs can fall through the cracks and lead to unhappy residents. Our Salesforce-based system will address these weaknesses by giving RentoMax visibility into details of properties and tenants, and providing a simple way for residents to submit requests and for RentoMax staff to manage them. This system improves clarity by sending requests to the proper team (electrical or plumbing) and efficiency by allowing residents to submit requests by phone, web, or email. RentoMax will also be able to see the status of the requests (Pending, In Process, or Resolved) with a 48-hour trigger for unresolved requests to be escalated to the central maintenance manager. RentoMax will be able to view all pending and completed requests on their own dashboard, and they won’t have to keep up with paper records that could easily be lost. After implementation, the system will have transformed RentoMax’s key business processes into the digital sphere, leading to improved service quality, customer experience, stronger resource management, and data collection.
* System Capabilities
  + Phone/web/digital maintenance request submission
  + Stores information on units like the number of bedrooms, bathrooms, balcony, and square footage, tenant name, tenant age, etc.
  + Automatic filtering of requests to electrical or plumbing team
  + Dashboard for RentoMax to view pending and completed requests
  + View into the status of requests (3 tiers -> pending, in process, resolved)
  + Ability to generate reports with details like time to resolve maintenance requests and status of requests
* Business Benefits
  + More efficient labor allocation
  + Supply of information to decision-makers
  + Faster service
  + Improved quality of service
  + Error minimization
  + Cost reductions
  + Better customer service
  + Continuous availability of the systems
  + Growth in communication capabilities and methods

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

(i.e., project-level definition of done).

* More efficient labor allocation: Scheduling the proper number of maintenance resources to an issue and scheduling them in the proper category (electrical or plumbing). Would increase the amount of jobs workers could get done due to the right amount of people in the right situation with the right tools.
* Supply of information to decision-makers: Dashboards for RentoMax would be created for the managers at the company streaming data into a way for managers to visualize and therefore use the data provided in a fast and efficient manner.
* Faster service: 80% faster scheduling of maintenance resources to an issue, 20% faster at resolving each maintenance request.
* Improved quality of service: With faster and better quality service comes lasting impressions on customers and the buildings the team is upkeep. This will improve customer relations with feedback due to the team fixing their problems. This will also affect the upkeep of the buildings due to better quality products while fixing the problems at hand for RentoMax.
* Error minimization: Reduce errors by 30%
* Cost reduction: Cost reductions in loss of labor and loss of materials will be improved upon due to the implementation of the platform. Rentomax will not be expending extra resources on small problems or not providing enough resources to larger problems which will increase bottom-line efficiency in both situations.
* Better customer service: Positive feedback through phone/web/email channels, fewer complaints
* Continuous availability of the systems: Achieve 99.999% System Reliability with a running system at all hours of the day. With a running system, maintenance requests will get resolved faster providing better customer relations, better reviews, better feedback, and less work on the manager and workers' workload.
* Growth in communication capabilities and methods: Contacting the maintenance team has added a new communication medium via the platform and maintenance portal for customers and Rentomax to interact through.

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

*Product Name: RentoMax Maintenance Request System*

**Product Description**

RentoMax Maintenance Request System is the perfect solution for apartment rental companies who need to manage their properties and tenants’ maintenance requests. Our Salesforce-based system automates key business processes, allowing you to add properties, tenants, and units with features such as bedrooms, bathrooms, balconies, and square footage. Our system also allows tenants to submit maintenance requests by phone, email, or web and track their status as “Pending, In Process, or Resolved.” Maintenance staff is specialized in electrical or plumbing-related tasks and each request can be escalated to the central maintenance manager if not resolved within 48 hours. Our system will generate reports to keep track of request details such as the time to resolve maintenance requests and will update the status of maintenance requests in the system. A tenant can view a dashboard to see information such as pending maintenance requests and maintenance requests that are completed.

**Features List**

* Add Properties and Tenants
* Details of each unit including features such as the number of bedrooms, number of bathrooms, and square footage
* Every property has a property manager
* Tenants occupy a unit with a fixed monthly rent
* Submit Maintenance requests by phone, email, or web
* Maintenance requests will be sent to the appropriate staff
* Each maintenance request has a status of “Pending, In-process, or Resolved”
* Generate reports to keep track of request details
* Dashboard to see information such as pending maintenance requests and maintenance requests completed

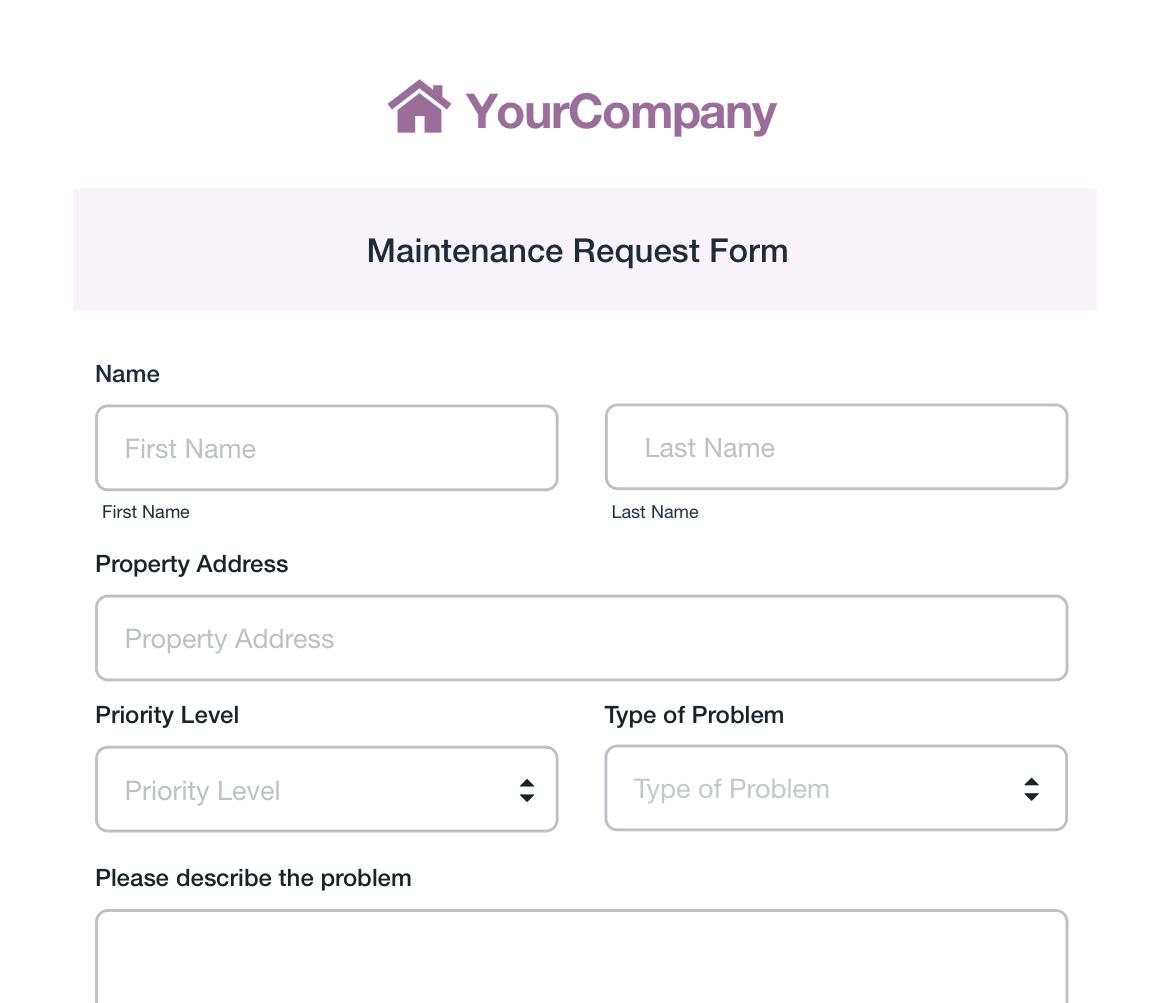
**Operating Requirements**

* Security: The maintenance system should have a secure authentication system in place to ensure that only authorized personnel can access and modify the records.
* Reliability: The maintenance system should be able to handle large volumes of data and requests without compromising performance.
* Scalability: The maintenance system should be able to scale up and down in response to changing needs and demands.
* User Interface: The maintenance system should be easy to use and intuitive for users.
* Reporting: The maintenance system should be able to generate reports for tracking and analysis of maintenance requests.
* Automation: The maintenance system should support automated processes and workflows.
* Monitoring: The maintenance system should monitor performance and provide alerts when necessary.
* Documentation: The maintenance system should have comprehensive documentation and help resources.

**Selling Points/Objectives**

* Automate key business processes
* Easily manage properties and tenants
* Quickly submit and resolve maintenance requests
* Keep track of request details and status
* Monitor information on a dashboard





# **1.3 System Analysis**

## **1.3.1. Information Gathering Techniques**

*How did your group collect the information regarding the proposed system?*

Our group had two gathering methods of collecting information for the proposed system. The first and most important for us was to interview Dr. Minetola. During the group/class interview, we asked many questions that narrowed down the scope and requirements for us. One question we asked was “How will this switch directly affect your tenants and what will they see on their end? Will they be able to manage their own property online (such as paying online, seeing their own status of requests, etc.)?” This allowed us to get information on how the tenants should not see anything on their end except what’s happening with their requests. Other questions we asked were “Is there one tenant per unit?” and “How do you hope this change will affect your overall long-term goals?”.

The second way of gathering information was through collaboration with other groups. While other groups were asking questions, our group was busy writing down the group's questions and the answers Dr. Minetola gave us. Some of the key questions the other groups asked are in the table below:

|  |  |
| --- | --- |
| Questions we asked: | * Is there one tenant per unit? * How will this switch directly affect your tenants and what will they see on their end? * Will they be able to manage their own property online (such as paying online, seeing their own status of requests, etc.)? * How do you hope this change will affect your overall long-term goals? * What are some of the major risks of this project? |
| Questions other groups asked: | * What functions and features are we automating and should prioritize? * What’s the difference between a system administrator and a property manager? * What information should we store on the tenants? * How should the dashboard tell the tenants if the request is 48 hours late? * What is the difference between pending and in- process? * Is there a limit on maintenance requests? |

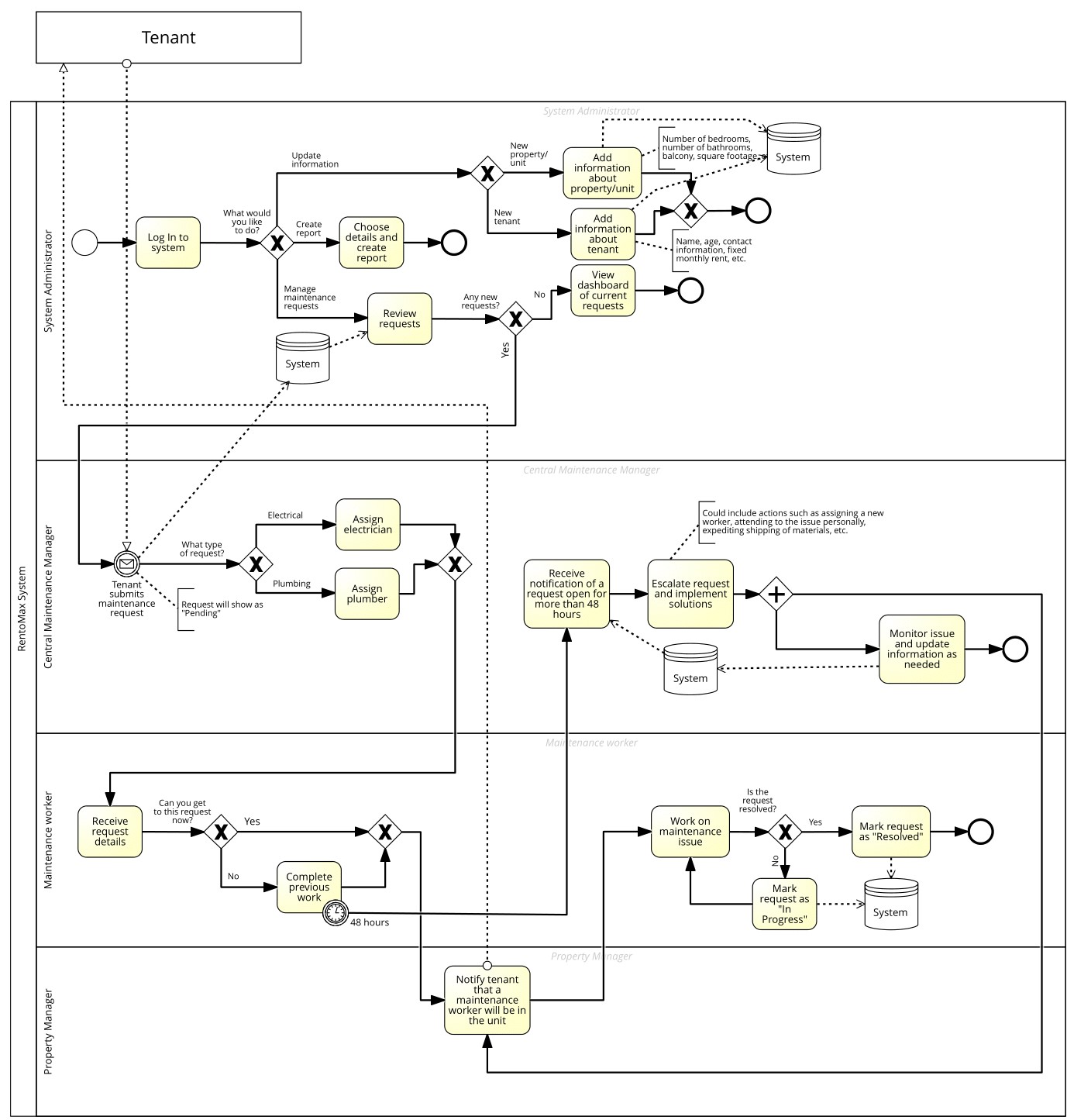
**There are three views of requirements: a process view, a user view, and a user interface**

**view. These are reflected in what is required for you to develop in this deliverable.**

## **1.3.2. 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.

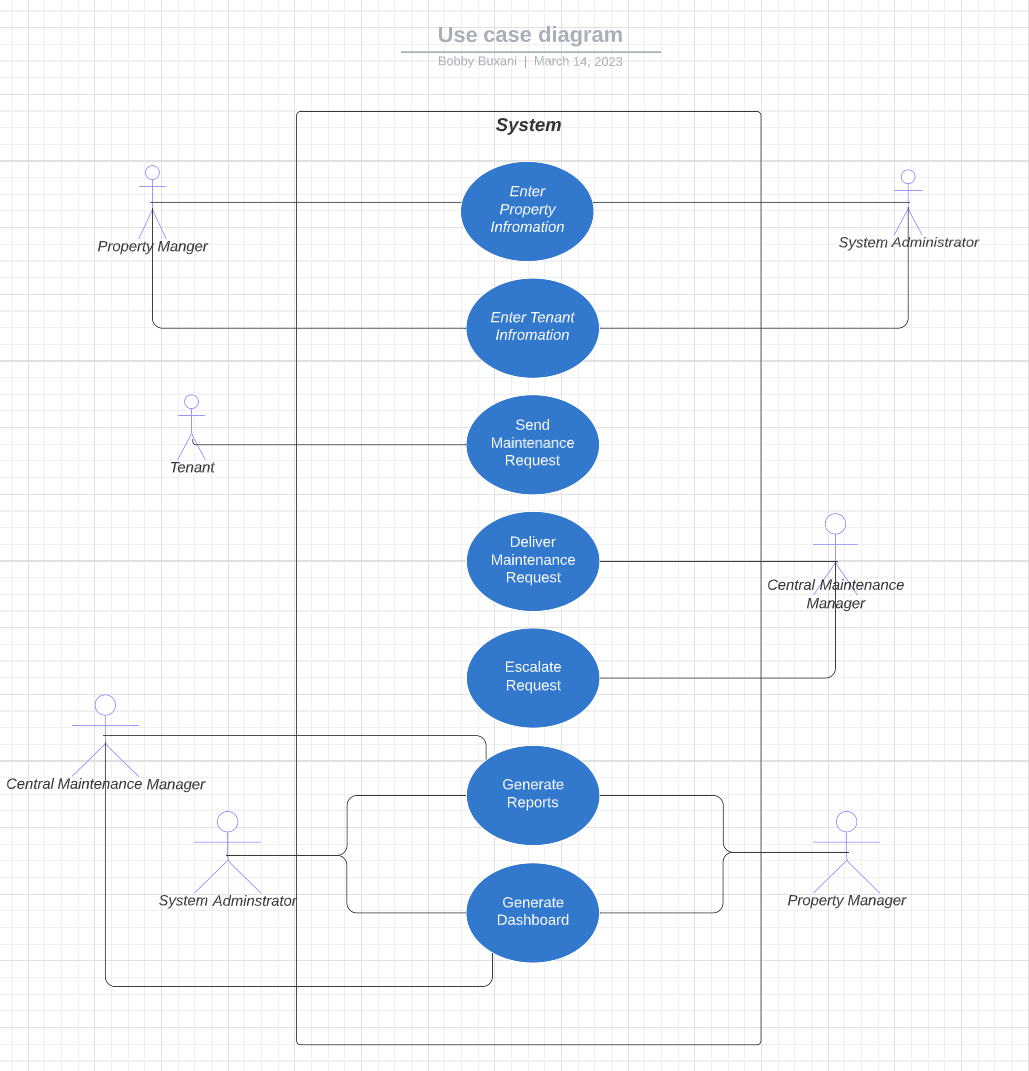


## **1.3.3. Use Case Diagram**

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.



## **1.3.4. 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)

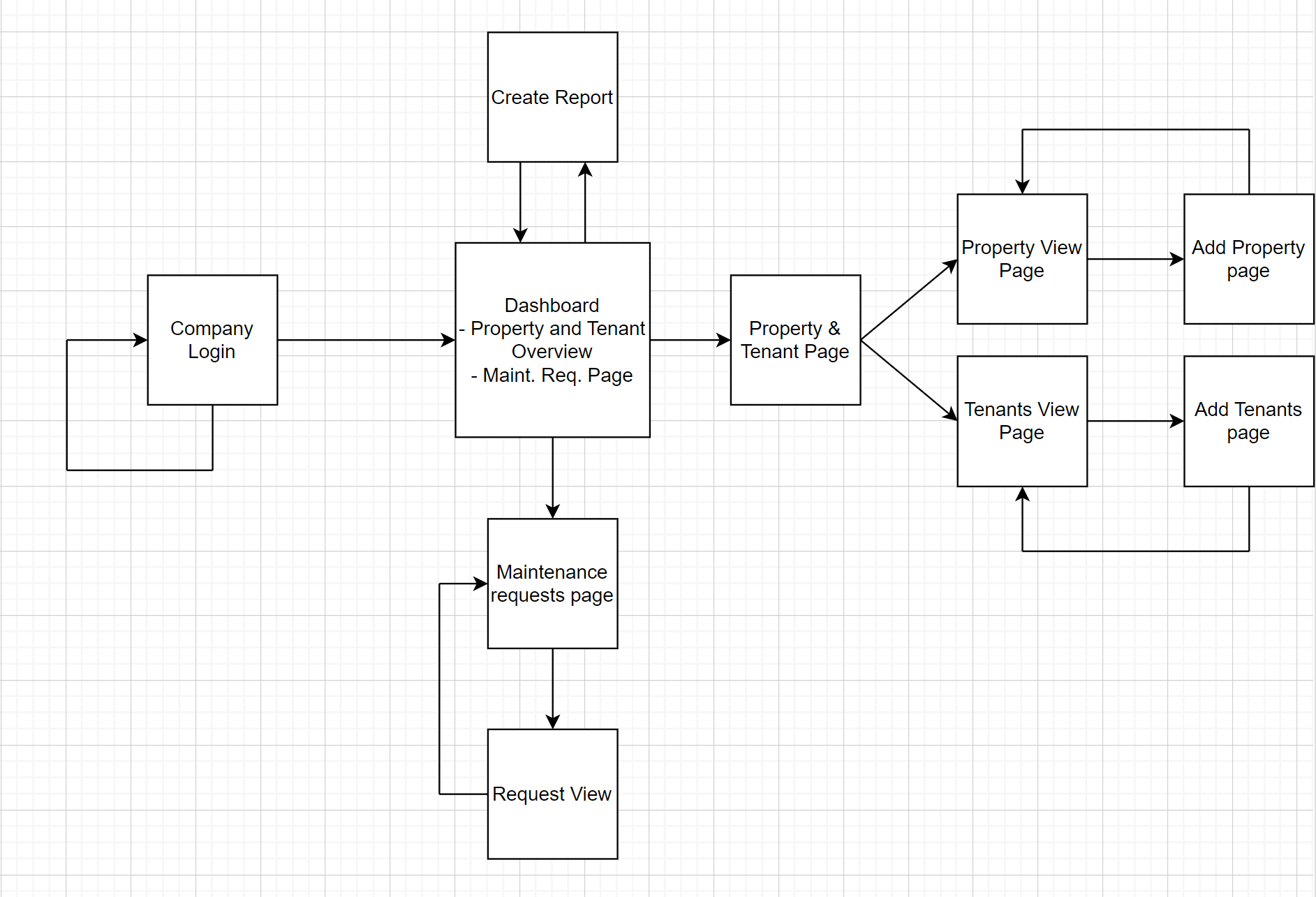
Column 2: A brief but comprehensive description of functionality provided by the use case

Column 3: Actors.

|  |  |  |
| --- | --- | --- |
| Use Case Name | Brief Description | Actors |
| Enter Property Information | The property manager will have the ability to keep track of the properties. The information that will be included is: number of bedrooms, number of bathrooms, whether the unit has a balcony, and square footage. | Property Manager, System Administrator |
| Enter Tenant Information | The property manager will have the ability to keep track of the properties’ tenants. The information that will be included is: name, age, etc. | Property Manager,  System Administrator |
| Send Maintenance Request | Tenants can send two types of maintenance requests: electrical or plumbing. | Tenant |
| Deliver Maintenance Request | The system will figure out if the maintenance request is plumbing or electrical and deliver it to the right staff. | Central Maintenance Manager |
| Escalate Request | If a request is not resolved for more than 48 hours, it needs to be escalated to the central maintenance manager | Central Maintenance Manager |
| Generate Reports | The system should keep track of request details such as time to resolve maintenance requests and status of maintenance requests | System Administrator,  Central Maintenance Manager,  Property Manager |
| Generate Dashboard | The system should also allow RentoMax to see information such as pending maintenance requests and maintenance requests completed. | System Administrator, Central Maintenance Manager,  Property Manager |

## **1.3.5. User Interface (UI) Flow**

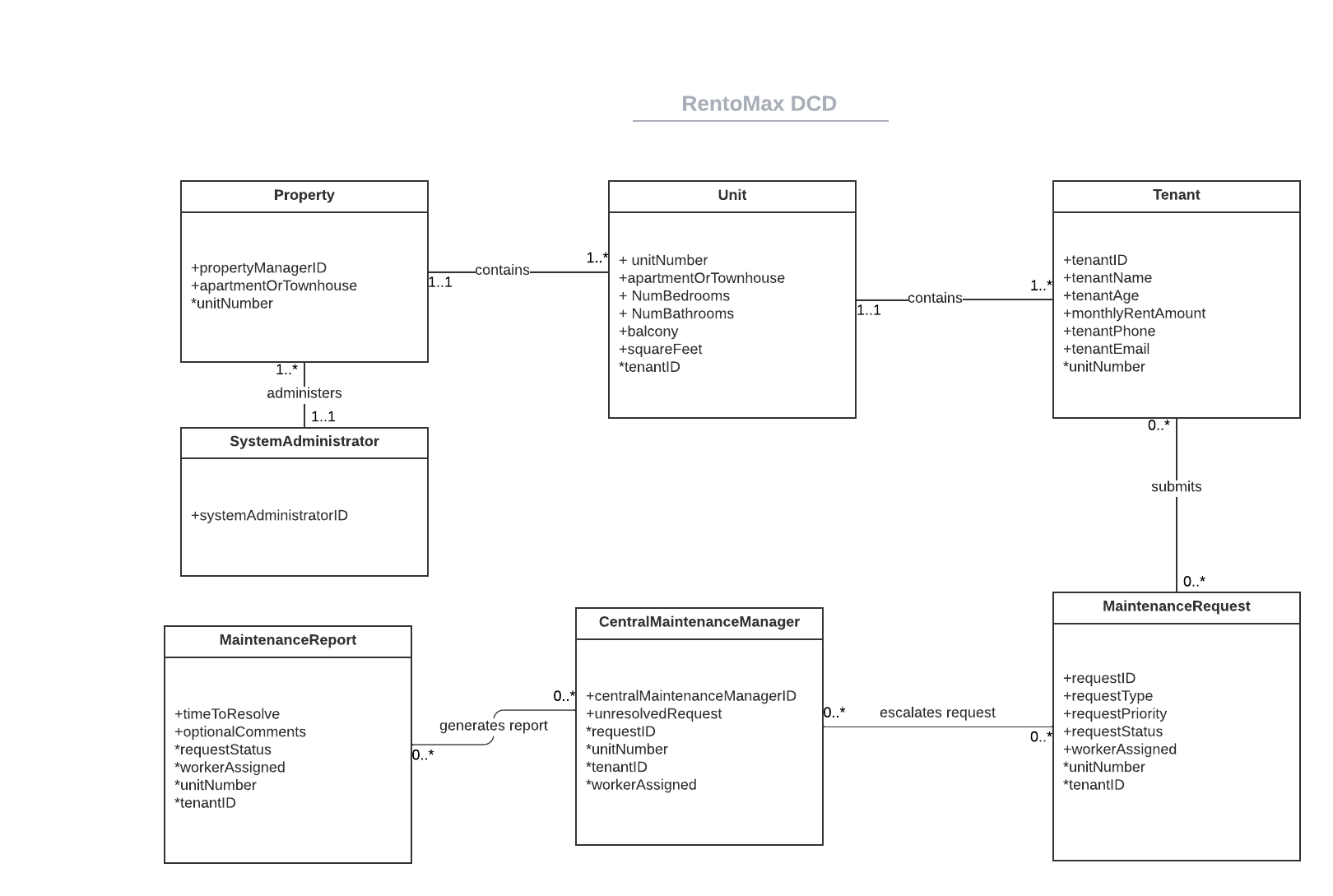
Include a diagram that shows the flow of UI screens.



## **1.3.6. Domain Class Diagram (DCD)**

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.

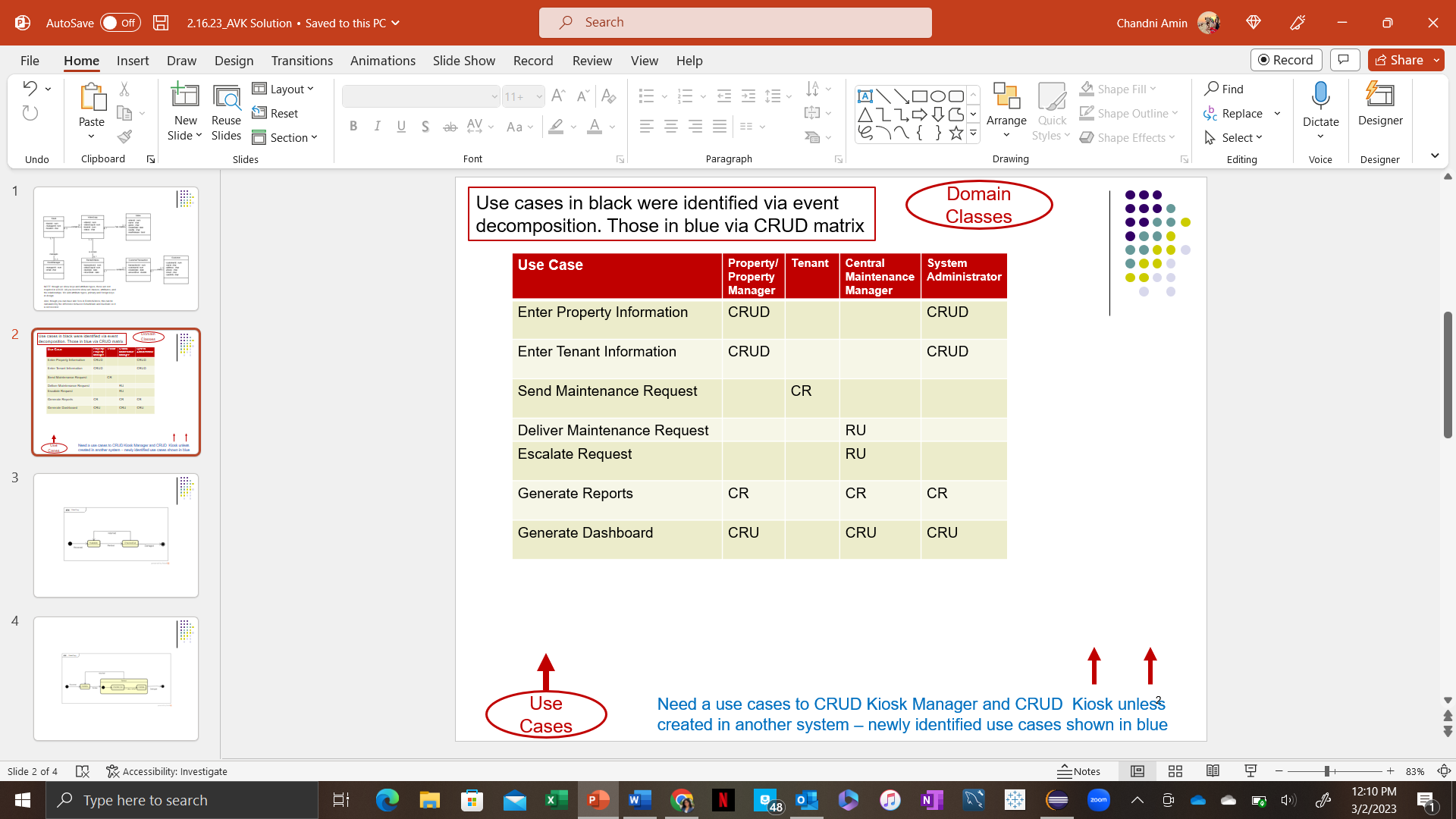


## **1.3.7. CRUD Matrix**

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.

****

# **1.4 Identification of Prototype User Stories**

## **1.4.1. Break your Use Cases (Features) into User Stories**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | User Story | User Story Template | Brief Description |
| Enter Tenant Information | Gather Tenant Information | As a Property Manager, I want to gather a tenant’s information so that I can input it into the system so we will have it on file. | The Property Manager will send out an email to the prospective tenant to ask for their information and enter that into the system |
| Send Maintenance Request | Send a Notice to Maintenance Staff | As a Tenant, I want to be able to send a maintenance notice to the maintenance staff so that I can have my apartment fixed | The Tenant should be able to send a notice quickly and electronically to the maintenance staff |
| Escalate Request | Figure out Solutions | As a Central Maintenance Manager, I want to be able to sort out possible solutions so that I can figure out which solutions are best. | The Central Maintenance Manager needs to know which solutions are best for this request and how to handle it accordingly. |
| Escalate Request | Implement Solution | As a Central Maintenance Manager, I want to be able to do the solution I picked so that I can alleviate the tenant’s maintenance request | The Central Maintenance Manager needs to implement the solution so that whatever issue the tenant is having is gone since the request has been at least 48 hours old. |
| Escalate Request | Monitor Solution | As a Central Maintenance Manager, I want to make sure that the solution works for a certain time after I have implemented the solution so that the maintenance request does not happen again. | The Central Maintenance Manager watches over the solution that he/she implements and sees if it worked or not. |

## 

## **1.4.2. Use Case Description**

|  |  |  |
| --- | --- | --- |
| Use Case Name: | Manage Property | |
| Scenario: | NONE | |
| Triggering Event: | Property manager wants to manage the details of RentoMax’s properties | |
| Brief Description: | The property manager will be able to use the platform to add new properties and record the details of those properties (such as address, name, etc.). After adding a new property, they will also need to add the associated units. They also want to be able to manage existing properties and change the previously recorded details (e.g. if the name of the property changes, if certain units no longer exist, etc.). Finally, they may need to delete a property if it is no longer a part of the RentoMax conglomerate. | |
| Actors: | Property Manager | |
| Related Use Cases: | Might invoke the Manage Unit use case to associate units with a new property | |
| Stakeholders: | System Administrator, Central Maintenance Manager, Maintenance Workers, Tenants | |
| Preconditions: | Property table must exist. Property Manager must exist. | |
| Postconditions: | Property record is created, updated, or deleted. Unit records may be created and associated with property. | |
| Flow of Activities: | Actor | System |
| 1. Property Manager enters the system  2. Property Manager adds a property  3. Property Manager updates a property  4 . Property Manager deletes a property | 1.1 System prompts for user credentials  1.2 System validates user credentials and provides access to user  2.1 System checks if property exists  2.2 If property does not exist, system prompts for property details (name, address, etc.)  2.3 System creates property record  3.1 System checks if property exists  3.2 If property does exist, system displays current information  3.3 System saves new information entered by user and updates property record  4.1 System checks if property exists  4.2 If property does exist, system displays a confirmation page  4.3 If user confirms they wish to delete a property, the property record is removed |
| Exception Conditions: | 1.1 User credentials are invalid  2.1 Property already exists  3.1 Property does not exist  4.1 Property does not exist | |

## **1.4.3. Prioritize User Stories**

**Product Backlog**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number | Use Case | As a… | I want to… | So that I can… |
| 1 | Create Data Structures (non-functional) | System User (Property Manager /Administrator/Maintenance Manager) | Set up data models and tables | Store maintenance and tenant information along with associations between entities |
| 2 | Create Logins (non-functional) | System User (Property Manager /Administrator/Maintenance Manager) | Have a secure login and authentication process | Ensure security of sensitive information |
| 3 | Increase Scalability (non-functional) | System User (Property Manager /Administrator/Maintenance Manager) | Add more properties, units, tenants, and requests to the system | Performance isn’t compromised or slowed down |
| 4 | Manage Property (Must-have) | Property Manager | Create, update, and delete properties | Have accurate knowledge of which properties I’m responsible for and to have property-specific information |
| 5 | Manage Unit (Must-have) | Property Manager | Create units of different types, manage their attributes, and associate them with properties | I can provide unit-specific information like monthly rent and rental status for current and future tenants |
| 6 | Manage Tenant (Must-have) | Property Manager | Create, update, delete tenants, associate them to a unit in a property, and capture tenant information like name, payment information, move in/out dates | Keep track of current residents, be able to contact them, and be aware when new units will open after tenants move out |
| 7 | Manage Maintenance Requests (Must-have) | Central Maintenance Manager | Enter requests that come in by phone, set status of requests, assign workers, and escalate after 48 hours if not resolved in that timeframe | Make sure each request is being assigned to a specific worker and ensure jobs are being completed efficiently |
| 8 | Determine solution to escalated request | Central Maintenance Manager | Brainstorm possible solutions | Complete the escalated request as quickly and efficiently as possible |
| 9 | Implement maintenance solution | Central Maintenance Manager | Execute the best determined solution | Alleviate the tenant’s maintenance request |
| 10 | Monitor solution to escalated request | Central Maintenance Manager | Monitor and assess that the solution worked | The tenant does not have the same problem again |
| 11 | Generate Tenant Report (Must-have) | System User (Property Manager /Administrator/Maintenance Manager) | Generate reports of tenant information (name, email) grouped by unit type and/or rental status | Keep accurate records of tenants residing in specific unit types |
| 12 | Generate Property Report (Must-have) | System User (Property Manager /Administrator/Maintenance Manager) | Generate reports of unit information (type, rental status, rent) grouped by property | Keep accurate records of what types of units exist in which properties, along with pricing and other attributes important to current and future residents |
| 13 | Generate Case Report (Must-have) | System User (Property Manager /Administrator/Maintenance Manager) | Generate reports with case information like type (electrical or plumbing) and date opened grouped by type and/or property/unit | Understand and visualize what types of maintenance issues are appearing in certain units/properties |
| 14 | Create Dashboards (Should-have) | System Administrator/RentoMax | Visualize pending maintenance requests and maintenance requests completed | Determine efficiency of maintenance resources and make any changes as needed |
| 15 | Allow tenants to submit maintenance requests by email (Could-have) | Tenant | Submit maintenance requests by email | Enter my request in a faster, more convenient way |
| 16 | Allow tenants to submit maintenance requests by web (Could-have) | Tenant | Submit maintenance requests by web | Enter my request in a faster, more convenient way |

## **1.4.4. Estimate Effort**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#** | **Use Case Name** | **Points** | **Tasks** | **Total Estimated Time** |
| 1 | Create Data Structures (non-functional) | 5 | Set up data models and tables | 1 hour |
| 2 | Create Logins (non-functional) | 3 | Have a secure login and authentication process | 1 hour |
| 3 | Increase Scalability (non-functional) | 8 | Add more properties, units, tenants, and requests to the system | 2 hours |
| 4 | Manage Property (Must-have) | 5 | Create, update, and delete properties | .5 hour |
| 5 | Manage Unit (Must-have) | 5 | Create units of different types, manage their attributes, and associate them with properties | 2 hours |
| 6 | Manage Tenant (Must-have) | 8 | Create, update, delete tenants, associate them to a unit in a property, and capture tenant information like name, payment information, move in/out dates | 2 hours |
| 7 | Manage Maintenance Requests (Must-have) | 8 | Enter requests that come in by phone, set status of requests, assign workers, and escalate after 48 hours if not resolved in that timeframe | .5 hour |
| 8 | Determine solution to escalated request | 3 | Brainstorm possible solutions | 1 hour |
| 9 | Implement maintenance solution | 5 | Execute the best determined solution | .5 hour |
| 10 | Monitor solution to escalated request | 3 | Monitor and assess that the solution worked | 2 hours |
| 11 | Generate Tenant Report (Must-have) | 5 | Generate reports of tenant information (name, email) grouped by unit type and/or rental status | 1 hour |
| 12 | Generate Property Report (Must-have) | 5 | Generate reports of unit information (type, rental status, rent) grouped by property | 1 hour |
| 13 | Generate Case Report (Must-have) | 5 | Generate reports with case information like type (electrical or plumbing) and date opened grouped by type and/or property/unit | 1 hour |
| 14 | Create Dashboards (Should-have) | 3 | Visualize pending maintenance requests and maintenance requests completed | 2 hours |
| 15 | Allow tenants to submit maintenance requests by email (Could-have) | 2 | Submit maintenance requests by email | .5 hour |
| 16 | Allow tenants to submit maintenance requests by web (Could-have) | 3 | Submit maintenance requests by web | .5 hour |
|  | **TOTAL STORY POINTS:** | **73** | **TOTAL HOURS:** | **17.5 HOURS** |

* The total story points for this selection of stories is 73.
  + We decided to split our total story points in half for two sprints 11 days each.
* Velocity = Total Story Points / Total # of Sprints = 73 / 2 = 36.5 points per sprint
  + The first sprint will be 37 story point
  + The second sprint will be 36 story points.
* Capacity in Hours: 110 hours total available in a one 11-day sprint or 10 hours per sprint
  + 5 group members x 2 hours allocated per group member x 11 days in one sprint = 110
* Velocity = Total Story Points / Available Hours = 36 / 110 = 0.33 for 1st sprint per hour
* Velocity = Total Story Points / Available Hours = 37 / 110 = 0.34 for 2nd sprint per hour
* Our expected velocity is 36.5 story points per sprint. OR expected velocity is 0.33 story points per hour for the first 11-day sprint. Our expected velocity for the second sprint is 0.34 story points per hour. This means that we expect to complete approximately 73 story points together in two separate 11-day sprints, assuming that we have the same team capacity and productivity.

## **1.4.5. Release Planning: Plan your Sprints**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Use Case Name | Points | Sprint 1 | Sprint 2 |
| 1 | Create Data Structures (non-functional) | 5 |  |  |
| 2 | Create Logins (non-functional) | 3 |  |  |
| 3 | Increase Scalability (non-functional) | 8 |  |  |
| 4 | Manage Property (Must-have) | 5 |  |  |
| 5 | Manage Unit (Must-have) | 5 |  |  |
| 6 | Manage Tenant (Must-have) | 8 |  |  |
| 7 | Manage Maintenance Requests (Must-have) | 8 |  |  |
| 8 | Determine solution to escalated request | 3 |  |  |
| 9 | Implement maintenance solution | 5 |  |  |
| 10 | Monitor solution to escalated request | 3 |  |  |
| 11 | Generate Tenant Report (Must-have) | 5 |  |  |
| 12 | Generate Property Report (Must-have) | 5 |  |  |
| 13 | Generate Case Report (Must-have) | 5 |  |  |
| 14 | Create Dashboards (Should-have) | 3 |  |  |
| 15 | Allow tenants to submit maintenance requests by email (Could-have) | 2 |  |  |
| 16 | Allow tenants to submit maintenance requests by web (Could-have) | 3 |  |  |

The sprint backlog breaks up the product backlog into the sprints for our project. To figure out which use cases go into each spring we used the points and velocity to see how to break up the project into two 11-week sprints. The first sprint is more work-heavy by a couple of points to allow for time in the second sprint cycle to finish up the project and the final sales force application.

**End of Deliverable 1**

**DELIVERABLE #2**

# **2.1. Structure of Sprint**

# 

## **2.1.1 Prioritized Product Backlog**

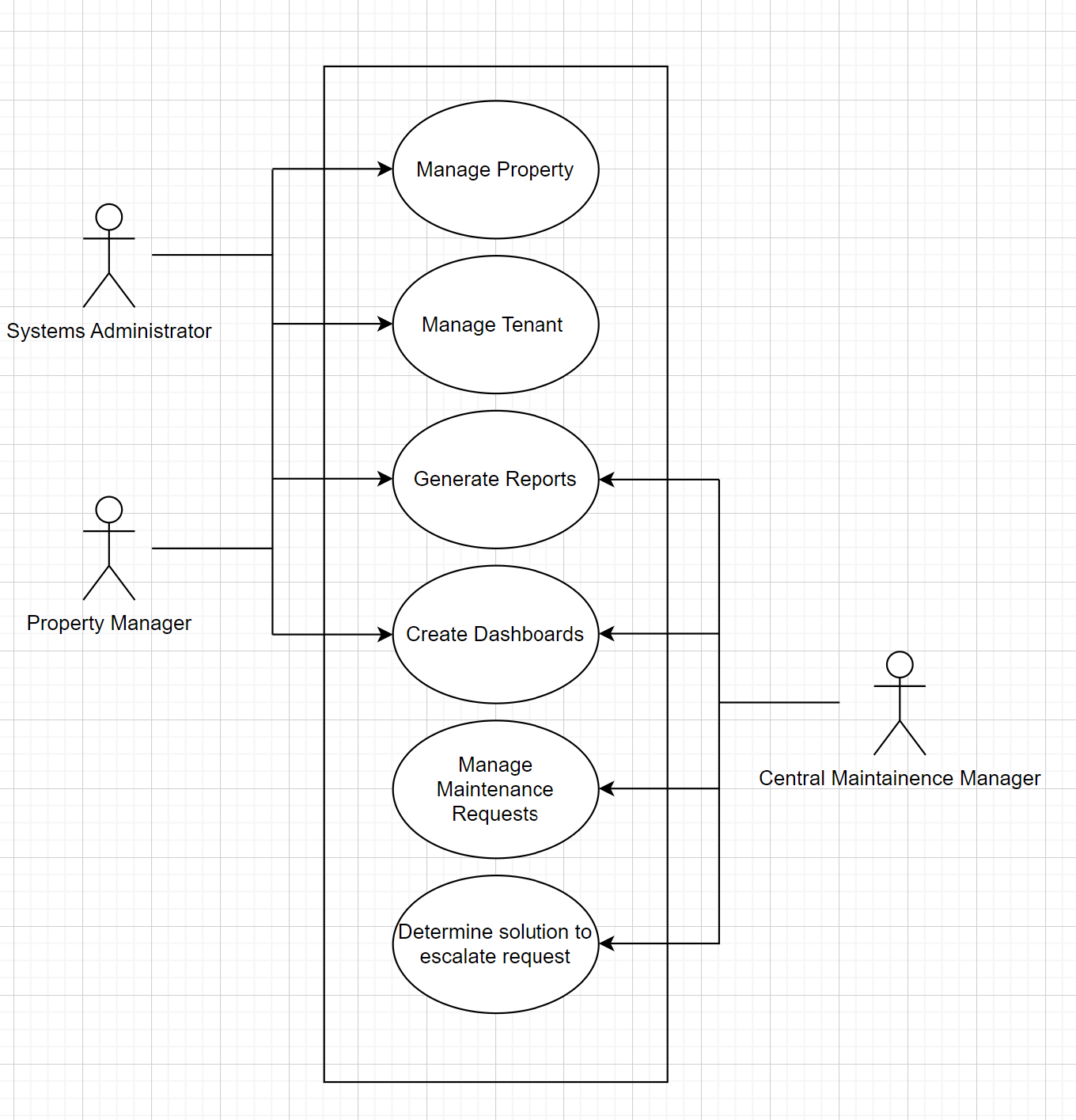
**Product Backlog**

**Sprint 1 Sprint 2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Number | Use Case | As a… | I want to… | So that I can… |
| 1 | Create Data Structures (non-functional) | System User (Property Manager /Administrator/Maintenance Manager) | Set up data models and tables | Store maintenance and tenant information along with associations between entities |
| 2 | Create Logins (non-functional) | System User (Property Manager /Administrator/Maintenance Manager) | Have a secure login and authentication process | Ensure security of sensitive information |
| 3 | Increase Scalability (non-functional) | System User (Property Manager /Administrator/Maintenance Manager) | Add more properties, units, tenants, and requests to the system | Performance isn’t compromised or slowed down |
| 4 | Manage Property (Must-have) | Property Manager | Create, update, and delete properties | Have accurate knowledge of which properties I’m responsible for and to have property-specific information |
| 5 | Manage Unit (Must-have) | Property Manager | Create units of different types, manage their attributes, and associate them with properties | I can provide unit-specific information like monthly rent and rental status for current and future tenants |
| 6 | Manage Tenant (Must-have) | Property Manager | Create, update, delete tenants, associate them to a unit in a property, and capture tenant information like name, **payment information, move in/out dates** | Keep track of current residents, be able to contact them, and be aware when new units will open after tenants move out |
| 7 | Manage Maintenance Requests (Must-have) | Central Maintenance Manager | Enter requests that come in by phone, set status of requests, assign workers, and escalate after 48 hours if not resolved in that timeframe | Make sure each request is being assigned to a specific worker and ensure jobs are being completed efficiently |
| 8 | Determine solution to escalated request | Central Maintenance Manager | Brainstorm possible solutions | Complete the escalated request as quickly and efficiently as possible |
| 9 | Implement maintenance solution | Central Maintenance Manager | Execute the best determined solution | Alleviate the tenant’s maintenance request |
| 10 | Monitor solution to escalated request | Central Maintenance Manager | Monitor and assess that the solution worked | The tenant does not have the same problem again |
| 11 | Generate Tenant Report (Must-have)  DONE | System User (Property Manager /Administrator/Maintenance Manager) | Generate reports of tenant information (name, email) grouped by unit type and/or rental status | Keep accurate records of tenants residing in specific unit types |
| 12 | Generate Property Report (Must-have)  DONE | System User (Property Manager /Administrator/Maintenance Manager) | Generate reports of unit information (type, rental status, rent) grouped by property | Keep accurate records of what types of units exist in which properties, along with pricing and other attributes important to current and future residents |
| 13 | Generate Case Report (Must-have) | System User (Property Manager /Administrator/Maintenance Manager) | Generate reports with case information like type (electrical or plumbing) and date opened grouped by type and/or property/unit | Understand and visualize what types of maintenance issues are appearing in certain units/properties |
| 14 | Create Dashboards (Should-have) | System Administrator/RentoMax | Visualize pending maintenance requests and maintenance requests completed | Determine efficiency of maintenance resources and make any changes as needed |
| 15 | Allow tenants to submit maintenance requests by email (Could-have) | Tenant | Submit maintenance requests by email | Enter my request in a faster, more convenient way |
| 16 | Allow tenants to submit maintenance requests by web (Could-have) | Tenant | Submit maintenance requests by web | Enter my request in a faster, more convenient way |

## **2.1.2 An updated Use Case Diagram, brief Use Case Description table, and Domain Class Diagram**

### Revised Use Case Diagram



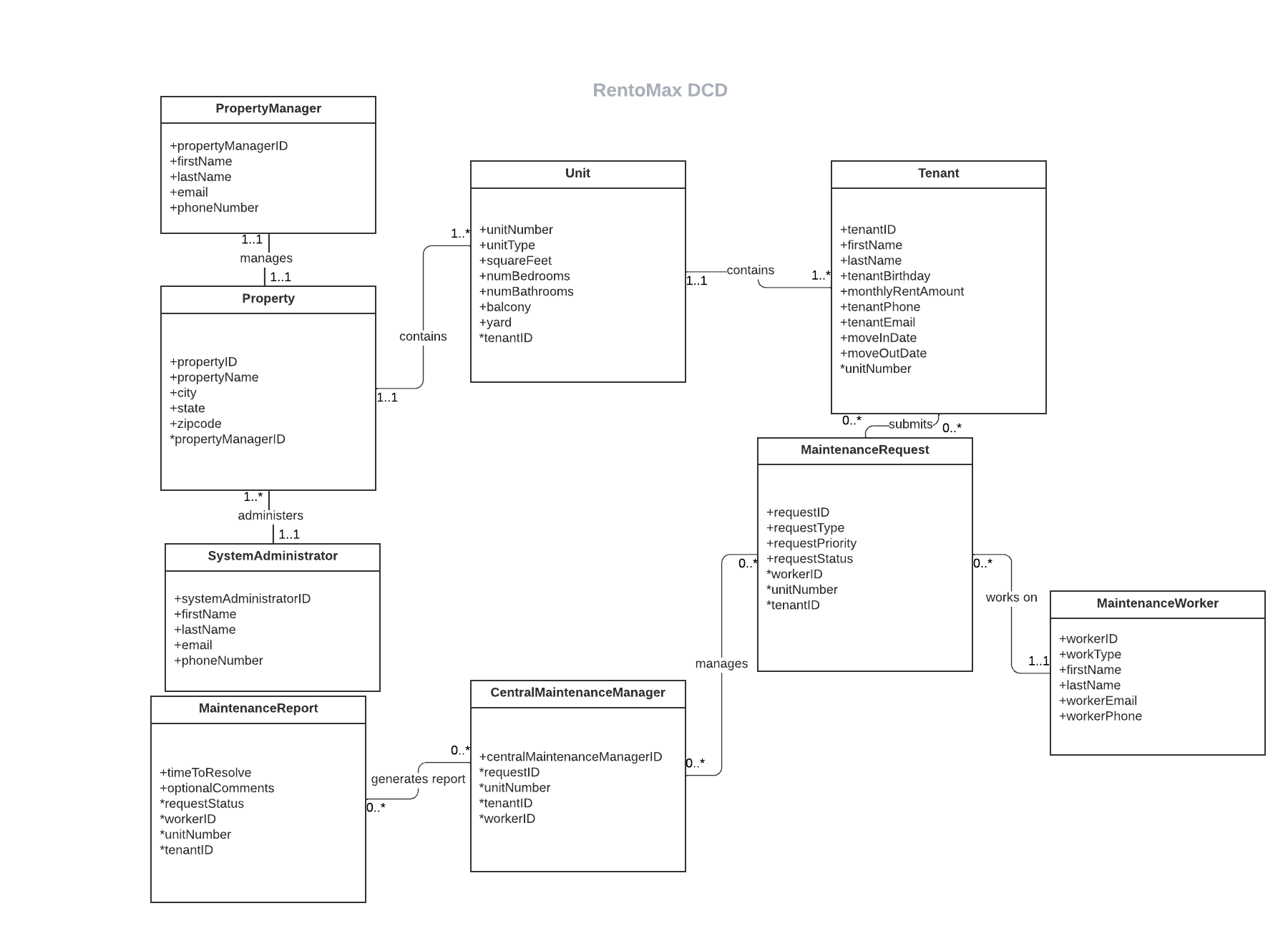
### Revised Use Case Description Table

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | **Brief Description** | **Actors** |
| Manage Property | The property manager will have the ability to keep track of the properties. The information that will be included is: number of bedrooms, number of bathrooms, whether the unit has a balcony, and square footage. | Property Manager, System Administrator |
| Manage Tenant | The property manager will have the ability to keep track of the properties’ tenants. The information that will be included is: name, age, etc. | Property Manager,  System Administrator |
| Manage Maintenance Requests | The central maintenance manager will have the ability to view requests, assign them to workers, update their status, escalate requests after 48 hours, etc. | Central Maintenance Manager |
| Generate Reports | The system should keep track of request details such as time to resolve maintenance requests and status of maintenance requests | System Administrator,  Central Maintenance Manager,  Property Manager |
| Generate Dashboard | The system should also allow RentoMax to see information such as pending maintenance requests and maintenance requests completed. | System Administrator, Central Maintenance Manager,  Property Manager |
| Determine solution to escalated request | The Central Maintenance Manager must be able to deal with escalated requests and determine appropriate solutions. | Central Maintenance Manager |

## 

## 

### Revised Domain Class Diagram



## **2.1.3 Starting Sprint Backlog**

|  |  |  |  |
| --- | --- | --- | --- |
| **User Story** | **Tasks** | **Team Member** | **Time** |
| Scrum Artifacts | Create sprint backlog | Riley | 20 min |
| Create Data Structures | Create entities for properties, units, and tenants | Everyone | 1.5 hours |
| Create relationships between entities | Everyone | 1.5 hours |
| Create Logins | Create contacts for all system users | Chandni | 30 min |
| Add information about each contact | Chandni | 20 min |
| Create logins for all system users | Chandni | 20 min |
| Increase scalability | Ensure that more properties, units, and tenants can be added as needed | Everyone | 30 min |
| Manage Property | Create properties (objects) that are under the RentoMax conglomerate | Bobby | 45 min |
| Add information about each property (name, address, etc.) | Bobby | 30 min |
| Manage Unit | Create custom objects for each unit | Charlie | 45 min |
| Add information about each unit (type, number of rooms, etc.) | Charlie | 30 min |
| Add ability to change unit status (Rented, Available, Unavailable) | Charlie | 20 min |
| Manage Tenant | Add contact information (phone, email, etc.) | Riley | 20 min |
| Be able to connect tenants to their unit and property | Riley | 30 min |
| Manage Maintenance Requests | Create form to enter maintenance requests | Emma | 2 hours |
| Create record types for maintenance issues (electrical or plumbing) | Emma | 20 min |
| Add ability to change request status (pending, in-process, resolved) | Emma | 20 min |
| Create escalation rules for requests open > 48 hours | Emma | 30 min |
| Determine Solution to Escalated Request | Brainstorm | Everyone | 20 min |
| Scrum Artifacts | Tutorial screencast | Chandni, Bobby, Charlie | 30 min |
| Sprint retrospective write-up | Riley | 30 min |

# 

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

## **2.1.5 Acceptance Criteria for each User Story in the Sprint**

|  |  |  |  |
| --- | --- | --- | --- |
| **User Story Name** | **Brief Description** | **Acceptance Criteria** | **Met?** |
| Create Data Structures | Create the necessary entities and their relationships | -Must have all necessary entities and their attributes  -Must be able to link entities through relationship | Yes |
| Create Logins | Create login credentials for each user | -Must have all necessary types of system users  -Each user must have a unique and secure login | Yes |
| Increase Scalability | Be able to add new properties, units, tenants, etc. as needed | -Must have ability to add new objects whenever it is needed | Yes |
| Manage Property | Be able to view, create, update, and delete properties and their attributes | -Must be able to create a new property  -Properties should have relevant attributes (e.g. address, name)  -A property consists of multiple units.  -Must be able to update a property  -Must be able to delete a property (and associated units) | Yes |
| Manage Unit | Be able to view, add, update, and delete units and their information | -Must be able to create a new unit and associate it with its property  -Units can be one of two types - apartment or townhouse  -Townhouses may have a yard and yard area. Apartments do not have yards.  -Each unit should have relevant attributes (number of bedrooms, number of bathrooms, square footage, etc.)  -Must be able to update an existing unit  -Must be able to delete a unit | Yes |
| Manage Tenant | Be able to view, add, update, and delete tenants and their information | -Must be able to create a new tenant. Each tenant needs to have one person’s name. Each tenant should be assigned to a unit in a property.  -A tenant should have relevant attributes (name, date of birth, etc.)  -Must be able to update an existing tenant  -Must be able to delete a tenant | Yes |
| Manage Maintenance Requests | Be able to view, create, update, and delete maintenance requests as well as escalate requests when needed | -The system user must be able to enter information for maintenance requests that come by phone from tenants  -Maintenance requests should have a status of “pending, in-process, or resolved”  -Maintenance requests should have relevant attributes (type, reason, tenant name, etc.)  -Once a new maintenance request arrives, and email notification should be sent to the respective maintenance person  -If a maintenance request is not resolved for more than 48 hours, it needs to be escalated to the system administrator | No  We were unable to set up email notifications once requests have been assigned due to insufficient Salesforce licenses. We will continue to work on this and plan to implement a solution in sprint 2. |
| Determine Solution to Escalated Requests | Brainstorm possible solutions to maintenance request escalations | -Must be able to understand and implement solutions for maintenance requests that have not been resolved | Yes |

## **2.1.6 Sprint Review - Tutorial Screencast**

* Here is the URL to our salesforce sprint one video.
  + <https://youtu.be/Ez6aMcgRD5w>

# 

## **2.1.7 Sprint Retrospective**

During our sprint retrospective, our team discussed our performance throughout the sprint, including any issues we faced as well as all of the positive observations about our work. By doing this, we were able to pinpoint what we want to continue doing throughout the next sprint, what we should start doing, and what we need to stop doing.

|  |  |
| --- | --- |
| **Continue doing** | |
| Communication | We did a great job communicating as a team. Our members were very responsive in the group chat and were open to discussing any issues or questions that were brought up. |
| Time management | We started our sprint early in the week which helped us not be as stressed when we ran into issues using Salesforce. We also planned our meetings around when office hours were being held so that we would have questions to ask the Salesforce professionals. |
| Delegation | We split up the work and delegated it to members of the team so that we would be able to work on the project whenever each member had time as well as so that no work accidentally got done twice. We created a shared document where we could see the work that everybody had gotten done and share any issues that we were having. |
| Respect | Our team had great respect for each other which helped us avoid any clashes. We understood each other’s time commitments and made sure that we were all contributing to the project. We also felt comfortable with each other and respected others’ opinions even if they differed from our own. |

|  |  |
| --- | --- |
| **Start doing** | |
| Better preparation | We felt as though we could have prepared for Salesforce better. We spent a lot of time trying to figure out the basics of using Salesforce when we should have been working on the specific aspects of our project by that point. For the next sprint, we want to dedicate time to research exactly what we need to get done in Salesforce and the basics of how to do it before we try to implement any solutions. |
| Check-ins | Our team had an idea of what we needed to get done from the beginning and we focused on that idea without much flexibility. We ran into some frustrating issues with how we were trying to use Salesforce that we spent a lot of time trying to work out without much progress. For the next sprint, we want to do more check-ins with the developers and be more flexible when it comes to issues we face. |
| Discuss requirements | We ran into issues when some of the requirements were confusing or we did not understand exactly what we needed to do. Additionally, because this is an agile project, the requirements sometimes changed and we were not aware of this without talking to stakeholders. For the next sprint, we want to be better about discussing the requirements with our stakeholders and asking for clarification when needed. |

|  |  |
| --- | --- |
| **Stop doing** | |
| Zoom meetings | Our team found that we are much more productive in person rather than virtually. We attempted to conduct some of our meetings via Zoom but realized that it is more difficult to communicate and work together when we are not physically in the same room. For the next sprint, we will ensure that we find time to meet in person whenever we need to work on the project. |
| Making assumptions | We ran into some issues where team members would make assumptions about what to do in Salesforce without everyone being on the same page. This caused us to have to do extra work to get back on track. For the next sprint, we want to have a clear outline of what work needs to get done so that nobody has to make assumptions that may hinder our progress. |

**End of Deliverable 2**

**DELIVERABLE #3**

# **3.1. Final Project Details**

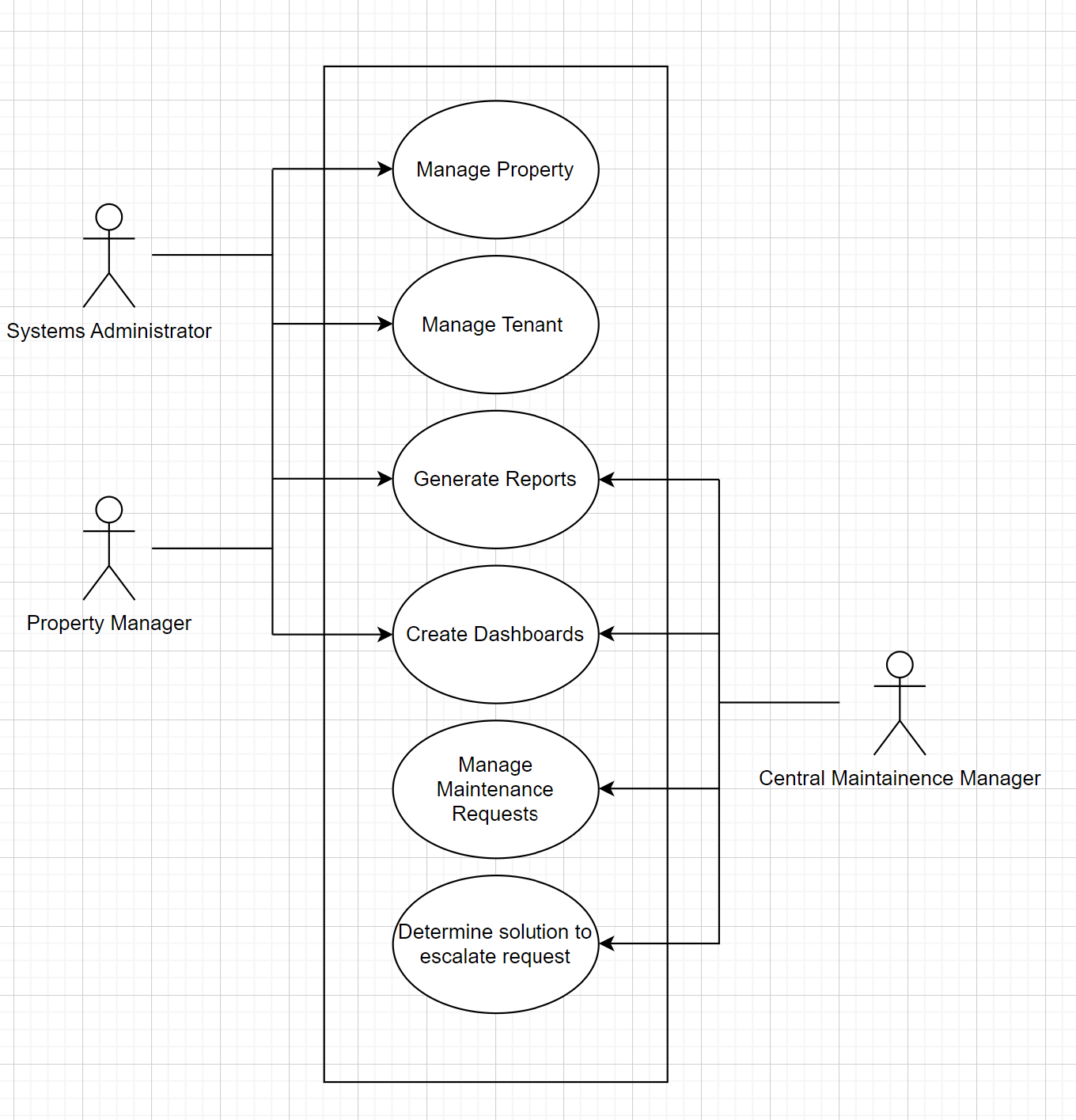
## **3.1.2. Executive Summary**

RentoMax is a property management company that is currently using a manual process to track all their tenants and maintenance requests from their different properties throughout Atlanta, Georgia. Attempting to keep track of all of this data manually has lead to inefficiencies and errors that reflect on the maintenance crew and ultimately can harm the tenant. RentoMax has requested our team, Bunch of Coders, to create a centralized system that will track all maintenance requests for electrical and plumbing-related issues that are sent in by tenants via phone, web, or email. Tracking the maintenance request information and storing related data on tenants (like unit number, phone, email. etc.) can minimize data errors and data loss by providing one place to enter and store that data while supplying key information necessary to manage the RentoMax properties. Since all of the essential data a property manager, maintenance manager, and worker needs is always accessible in the system, issues can be serviced faster and with more efficiency. RentoMax also requested the system to track maintenance requests that are pending, in-progress, or resolved along with automatically notifying maintenance workers on their assigned units. These status and notification features will provide more visibility into the maintenance process and improve labor and resource allocation for each maintenance issue that arises. Our system will address all weaknesses in their process by providing a single platform to generate reports, keep track of request details, and take care of any other actions RentoMax may need.

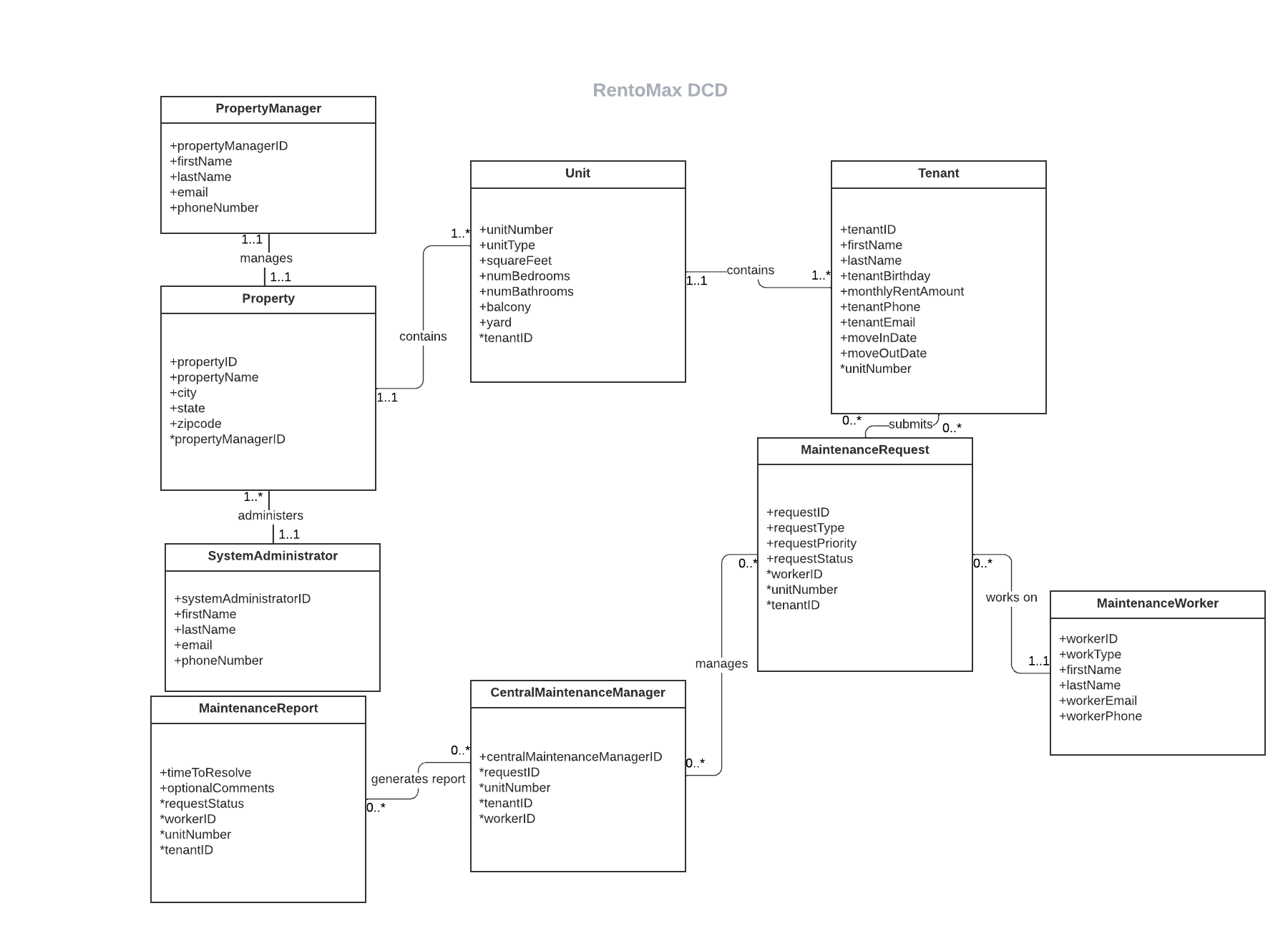
In our Salesforce-based system, we have created a maintenance request page where property managers can create a maintenance request based on information sent in by the tenant. This page captures all relevant data including the type of issue (electrical or plumbing), and tenant information, as well as a space to assign a maintenance worker to the request and update its status. If a maintenance request has not been resolved within forty-eight hours, our email alert feature escalates the request to the central maintenance manager. These features aid in more efficient maintenance scheduling and ensure no issue is left unresolved. Outside of our maintenance request page, our team has also created a dedicated page for properties, units, and contacts. On each page, system users can view details such as the type, the manager of each property, the units within those properties, and tenants who reside in each unit. Specific details like contact information of tenants and system users, the square footage of a unit, unit occupation status, monthly rent, yard size, and more can be found in the “Details” section of each page. These pages are also dynamic, so users can create, update, and delete properties, units, contacts, and all corresponding details. Accessibility of this information in one place helps every member of the RentoMax team provide faster and better quality service to tenants. In the reports page system, users will find generated tenant reports that keep an accurate record of tenants residing in a specific unit type along with all the tenant’s information grouped by unit type and rental status. On this page, there is also a property report that keeps an accurate record of the types of units that exist in each property, along with pricing and important attributes to current and future residents. System users can also generate specific case reports that include maintenance request information like the property, unit, type, and any additional comments from the tenant or assigned maintenance worker. The dashboards page provides visibility into the maintenance request process as a whole and can be a tool for management to analyze what types of maintenance issues are occurring in certain locations, as well as how long they take to resolve. The reports and dashboards features are key for the team to focus on continually raising the standard of their service by analyzing weaknesses and adjusting their tactics as needed.

## **3.1.3. Use Case Diagram and Domain Class Diagram**

### Use Case Diagram



### Domain Class Diagram



## **3.1.4. System Prototype (Demo)**

* Salesforce Login:
  + Link:<https://universityofgeorgia72-dev-ed.develop.lightning.force.com/lightning/page/home>
  + Username: [eet39061@uga.edu](mailto:eet39061@uga.edu)
  + Password: Mist4620S
* Link to Demo: <https://drive.google.com/file/d/11NLiV_EaTBS8ZrDqmlTNITfUrbLqpuff/view?usp=share_link>

## **3.1.5. User Manual (Video)**

* Link to User Manual Video: <https://drive.google.com/file/d/1jREGMtOtPpItt7NFqVCFpKZfpNTm8JJL/view?usp=share_link>

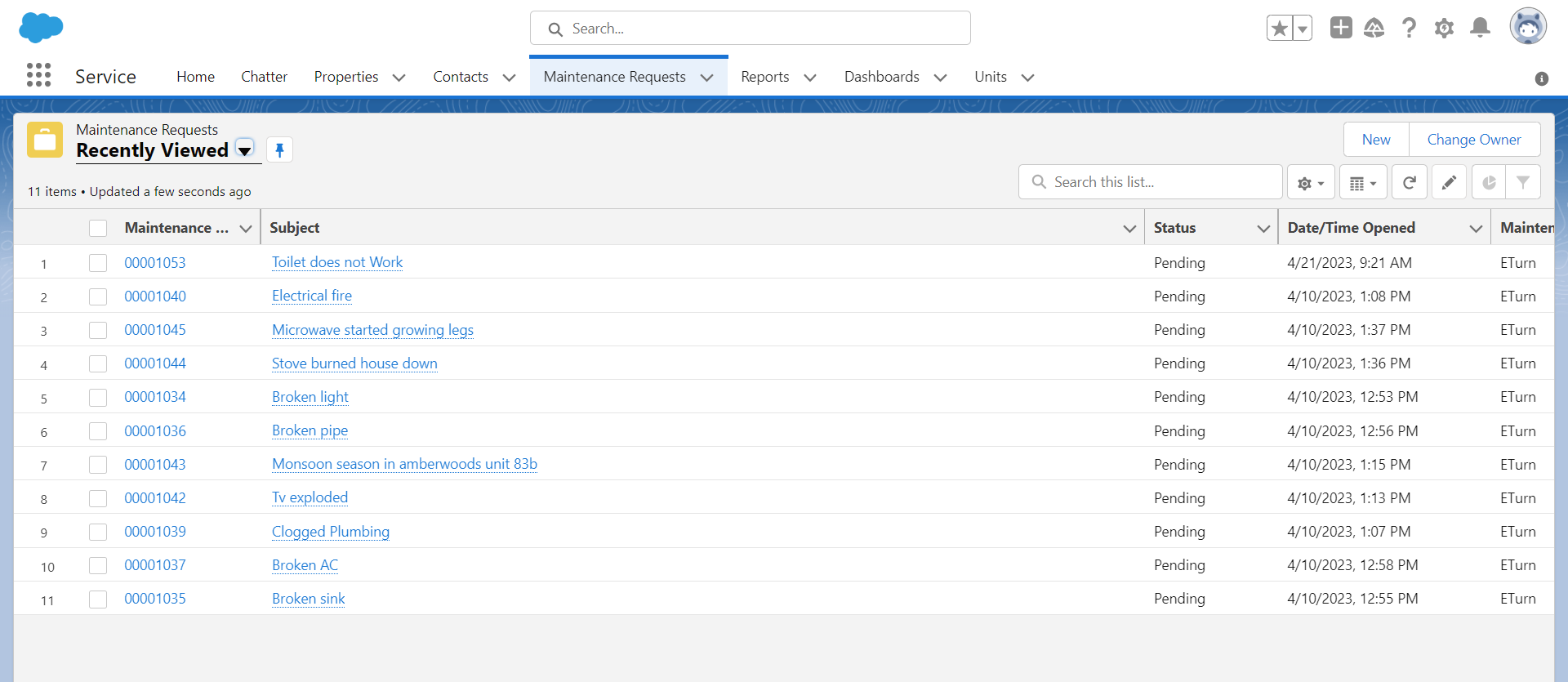
## **3.1.6. Screenshots**

### Properties Screen

### 

### Contacts Screen

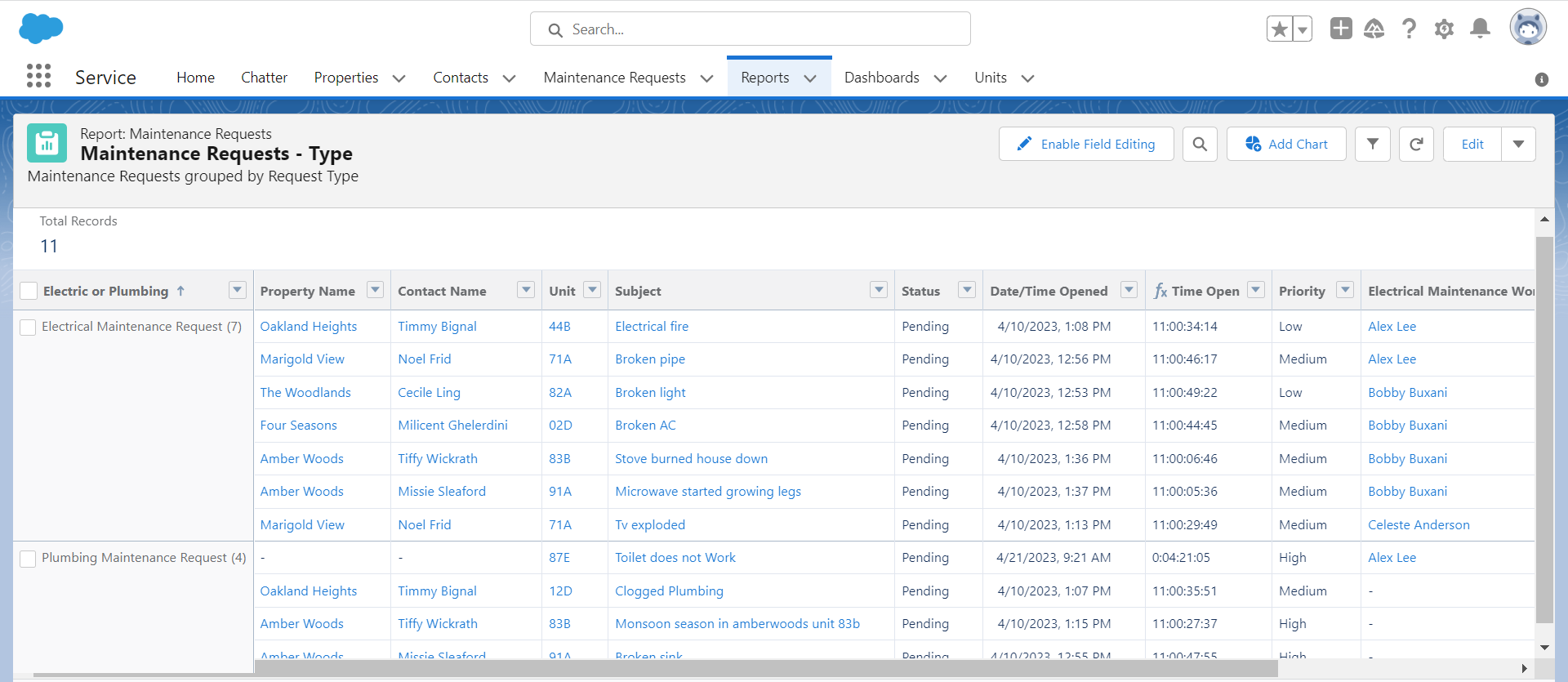
### Maintenance Requests Screen



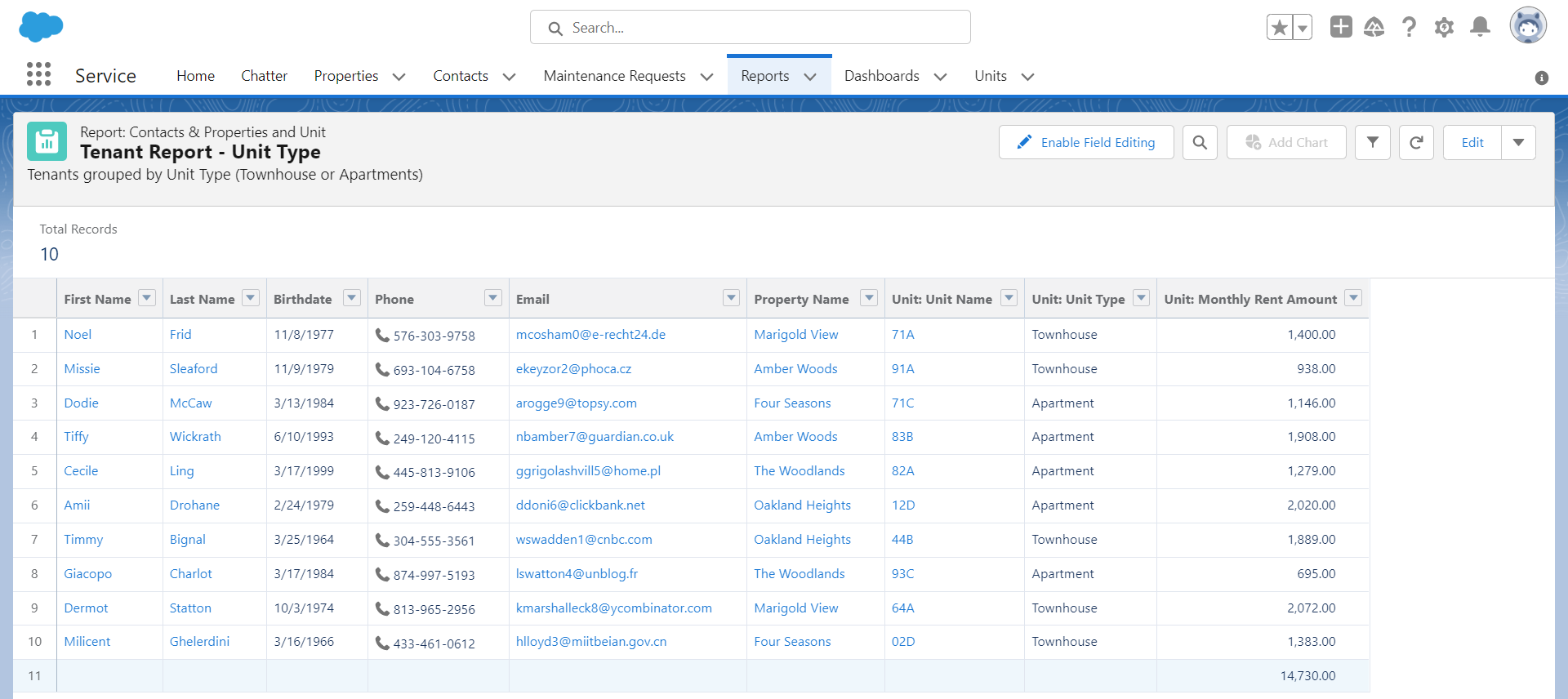
### New Maintenance Request Form

### Maintenance Requests by Properties Report

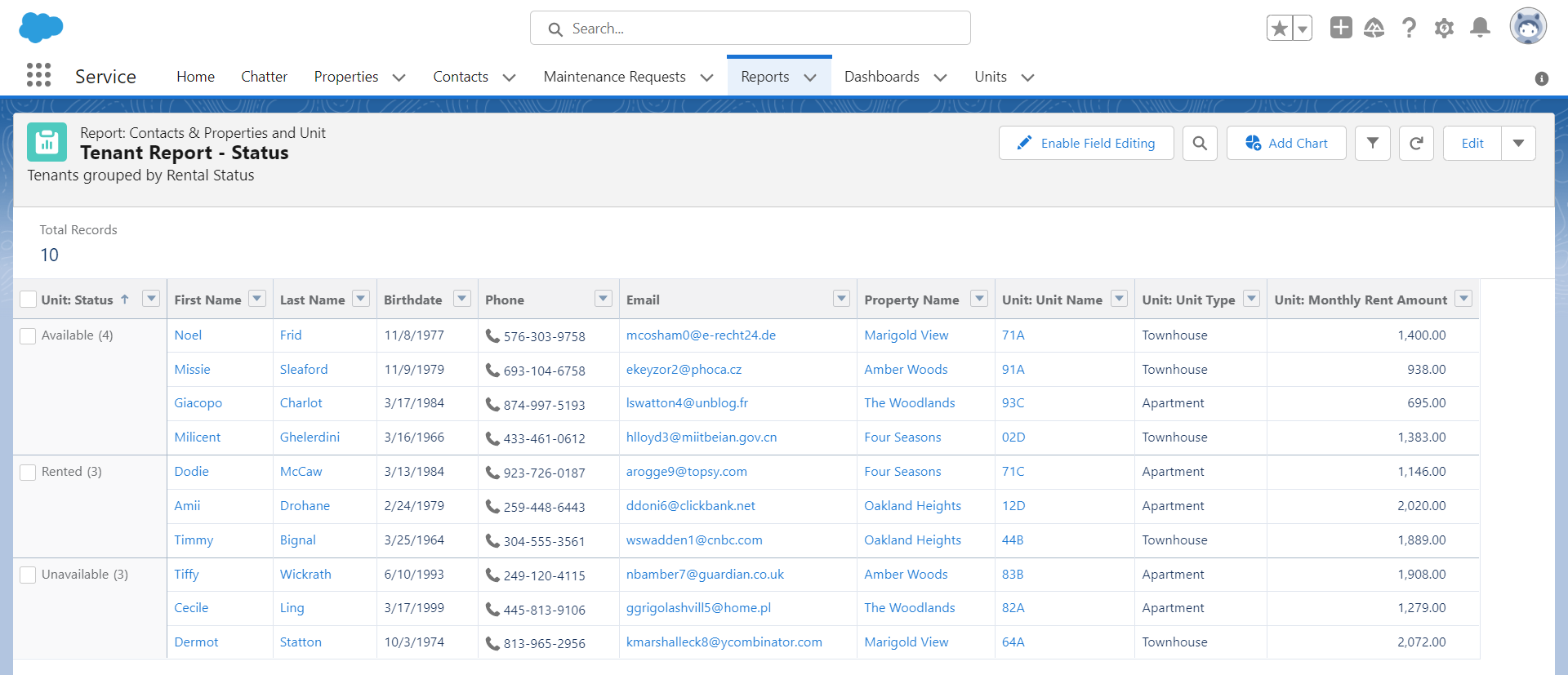
### Maintenance Requests by Type Report



### Tenants by Unit Type Report



### Tenants by Status Report



### 

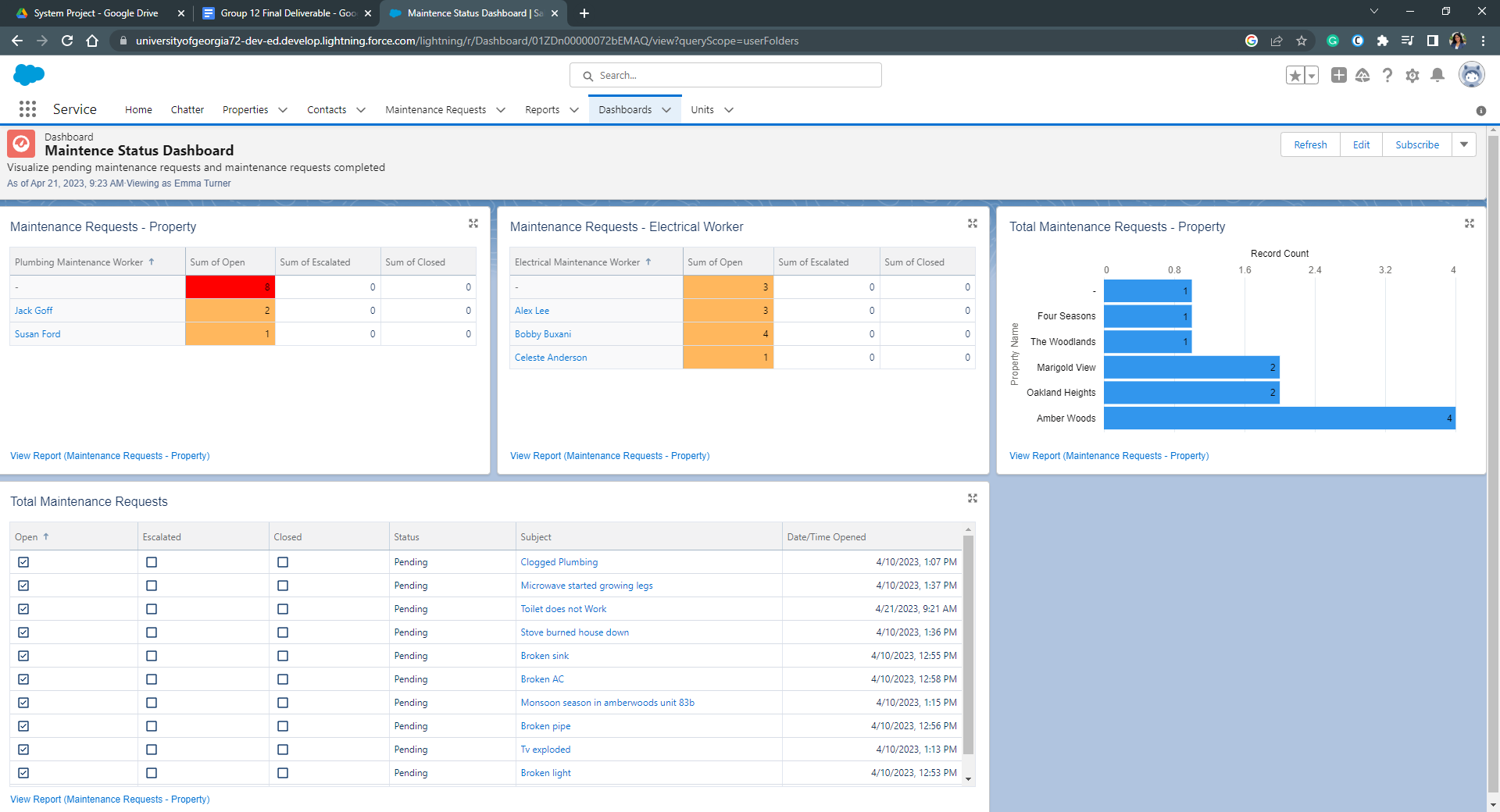
### 

### 

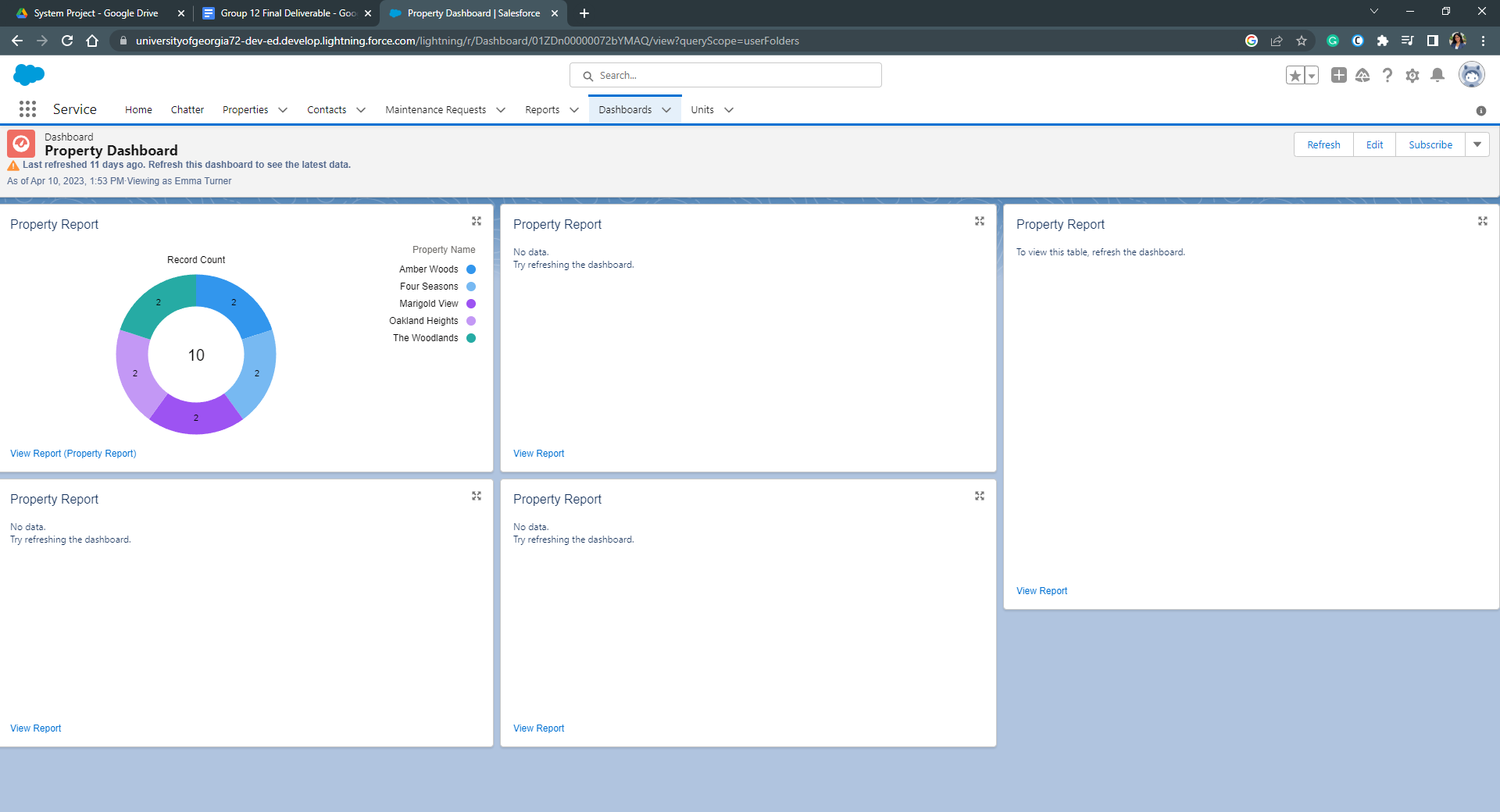
### 

### Property Report

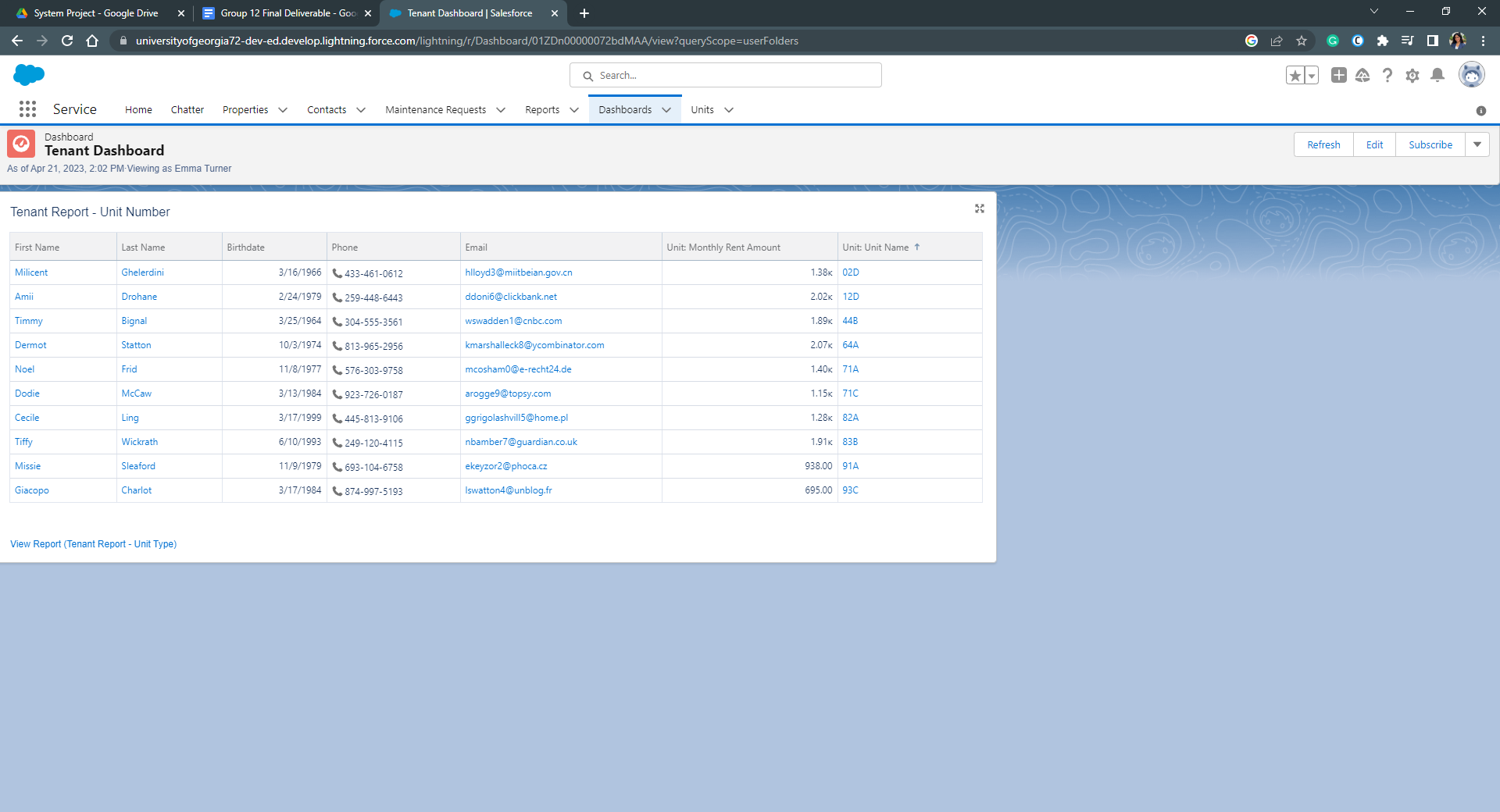
### Maintence Status Dashboard



### Property Dashboard



### Tenant Dashboard

****

## **3.1.7. Project Evaluation**

After my team completed building the system we are most proud of is how easy it is to file a maintenance request. We also made it very easy to see all information on a property, unit, tenant, or worker through our Contacts and Units page. The one thing my team could have improved is to name our page RentoMax instead of Service. We have met all the requirements that were in scope, such as creating a maintenance request form with all relevant attributes, a property page, units page, contacts page, etc… Building our system would have gone much smoother if my team and I were more proficient in using Salesforce from the start. We should have taken more time on learning how to use Salesforce before we started building the system, but the Trailmixes and guest lectures helped us start out We spent a lot of time in the early stages trying to figure out the basics of creating the application when we could have been working on the specific aspects of our project by that point.

We ran into issues when some of the requirements were confusing or we did not understand exactly what we needed to do. Additionally, because this is an agile project, the requirements sometimes changed and we were not aware of this without talking to stakeholders. We resolved this by discussing the requirements with our stakeholders and asking for clarification when needed. Overall, we are proud of our system and all of its functions.

## **3.1.8. Project Retrospective: Lessons Learned**

Lessons my team and I learned during this project was how important it is to communicate with your team members and client. This project would not have been as well done and successful as it is without communication. Next, we had to realize that requirements can change and we would have to adapt. Nothing ever goes according to plan, and we had so many roadblocks while working in Salesforce. It took patience, teamwork, and problem-solving skills to keep moving forward with building the system. Having a well-done use case diagram came in handy when we were building our system. We also utilized the sprint backlog as a checklist for the system to make sure we met all the requirements.

The next lesson we learned was that meeting in person instead of over Zoom is key. Our team found that we are much more productive in person rather than virtually. We attempted to conduct some of our meetings via Zoom but realized that it is more difficult to communicate and work together when we are not physically in the same room. We were more efficient working together in person and we produced better quality work.

The last lesson we learned was to not make assumptions. We ran into some issues where team members would make assumptions about what to do in Salesforce without everyone being on the same page. This caused us to have to do extra work to get back on track. We overcame this by creating a clear outline of what work needs to get done so that nobody has to make assumptions that may hinder our progress.

**End of Deliverable 3**