SW Engineering CSC648 Summer 2021

DormMates

Milestone 1 • Version 2 • 22 June 2021

Team 01

Team Lead and Github master	Andrei Georgescu	ageorgescu@mail.sfsu.edu
Frontend Lead	Meeka Cayabyab	
Backend and Database Lead	Jonathan McGrath	
Engineer	Alexandre Ruffo	
Engineer	Jimmy Yeung	
Engineer	Sayed Hamid	
Engineer	Ahad Zafar	

History Table

Version	Date	Notes
M1V2	07/01/2021	Added database master role to the title page, addressed use cases feedback, addressed functional requirements feedback, and addressed competitive analysis feedback.
M1V1	6/22/2021	Initial submission

Table of Contents

Executive Summary

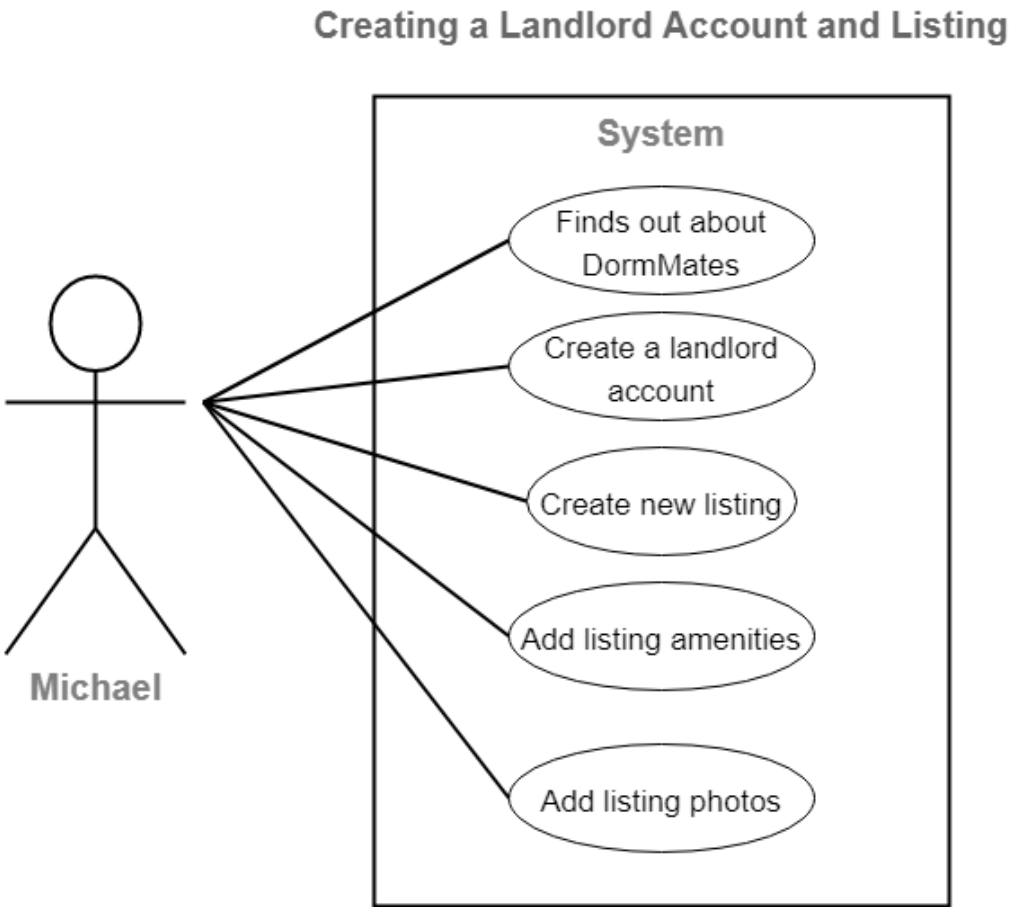
One of the most challenging things to do as a college student is finding the right living situation for you. For example, San Francisco State University only has housing available for less than 12% of registered students. As a third of new incoming freshmen are becoming more concerned with the ever-increasing prices of housing, they must resort to other means to find affordable housing. These students may find themselves utilizing various different websites and tools, making their search for housing inconsistent. Students may use Facebook to find the right group of roommates and Craigslist to find the right apartment. DormMates aims to simplify this entire process as it allows students to find both satisfactory roommates and housing that aligns with their needs in one easy-to-use platform.

Living away from home for the first time can be an overwhelming experience for most college students. The notion of having to live with a random person that you know nothing about, in a location you did not choose can greatly impact a student's emotional and academic success. DormMates helps students get paired with roommates based on their academic major, hobbies, interests, schedules, and lifestyle. In addition to this, DormMates can also be used by landlords to list their housing unit. Landlords can chat with multiple students before deciding on an agreement and they are held to a high standard by our Landlord rating system to ensure students find the best housing option possible.

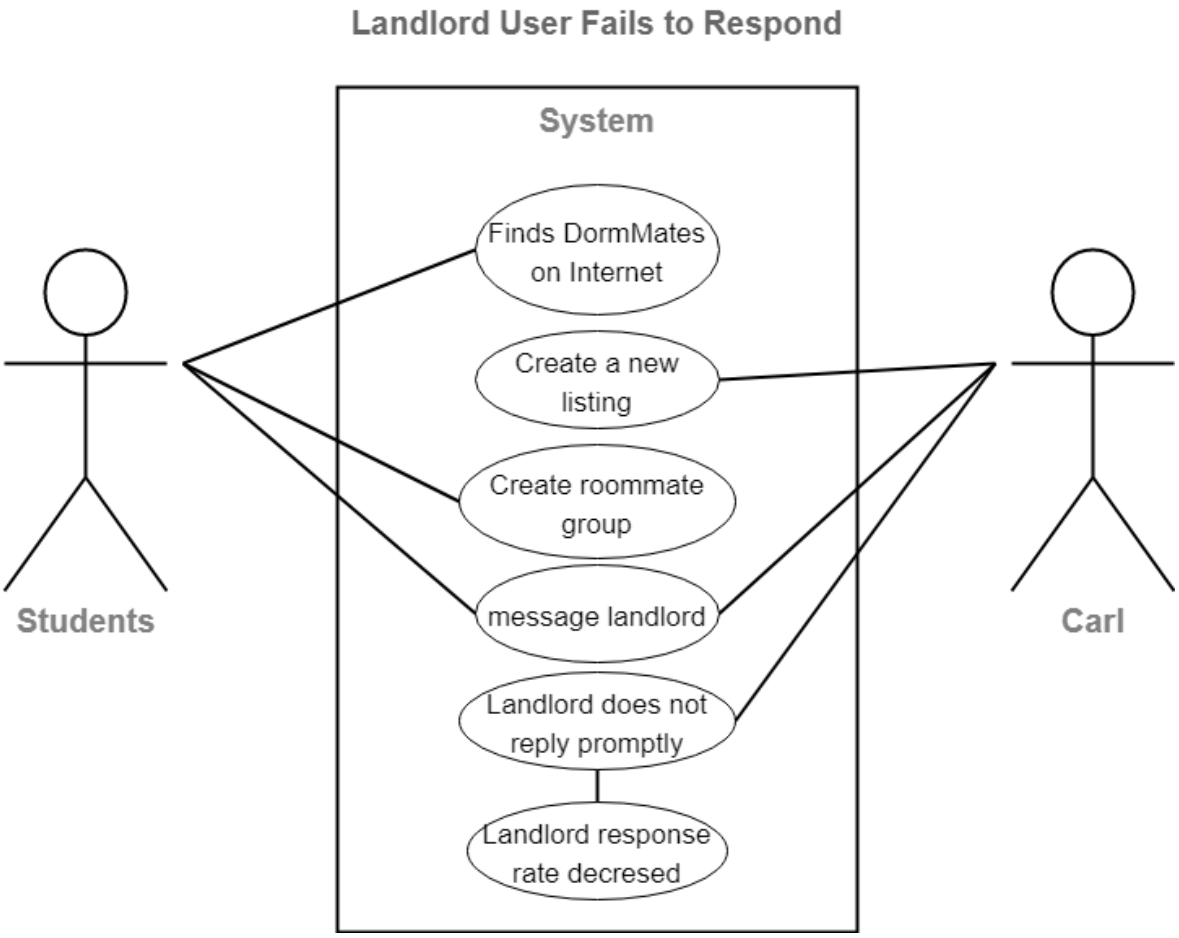
DormMates seeks to empower college students to take their living situation into their own hands by providing them a platform to connect with landlords and other students. We aim to give both students and landlords peace of mind when searching for roommates or listing their units on DormMates through several key features. Firstly, DormMates requires all students to have a valid and verifiable university email address. Next, our roommate matching service will utilize surveys and other techniques to match students with the right roommates for them. Lastly, our landlord rating system will ensure that all landlords provide students with the highest quality of living possible.

Main Use Cases

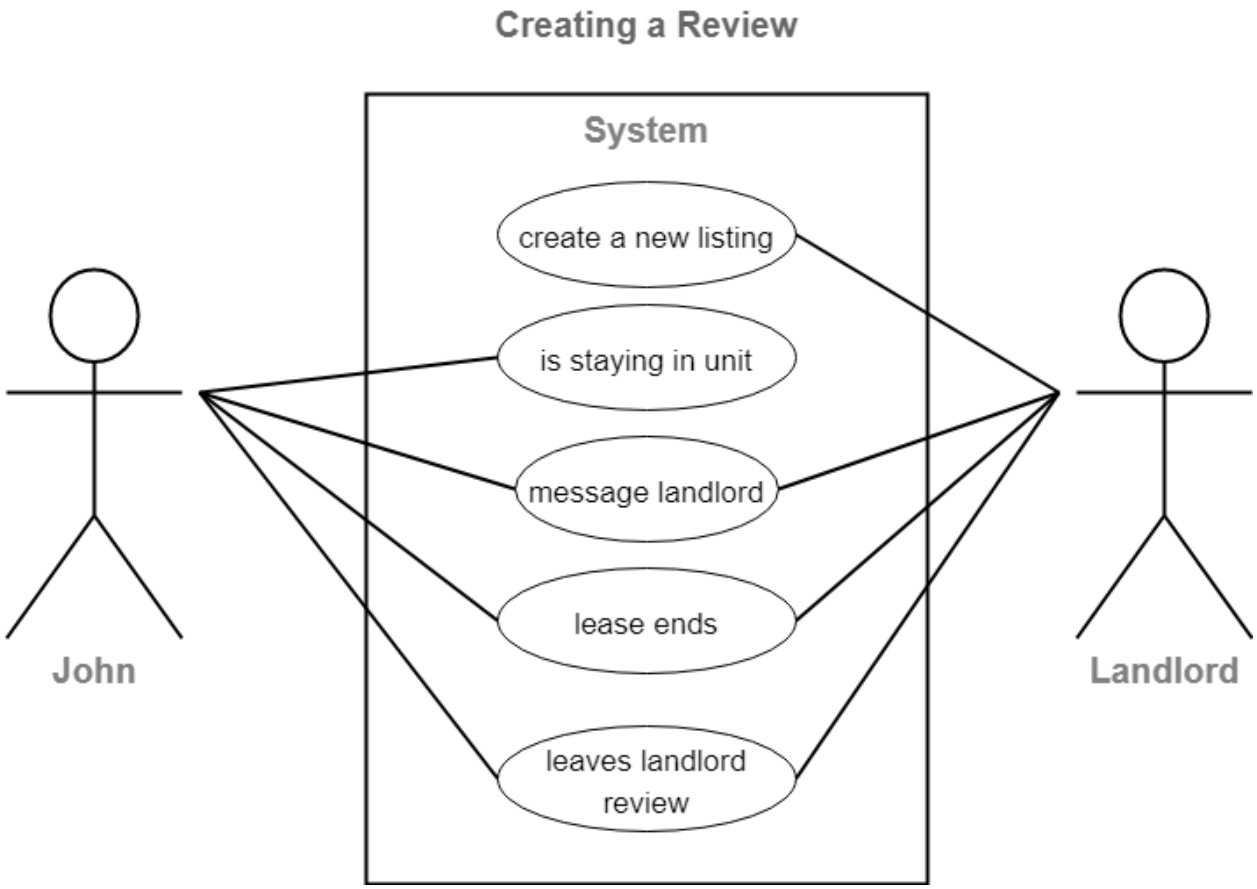
Title:	Creating a Landlord Account and Listing
Actor(s):	Michael
Description:	A landlord, Michael, purchases his first rental unit and is looking for college students to be his tenants. Michael contacts a friend who has been renting his rental unit to students and tells Michael about DormMates. Michael then creates a DormMates Landlord account. He creates a new listing for his unit. Michael also compiles a list of all amenities his unit has to offer and realizes that his unit can comfortably host 4 college students.



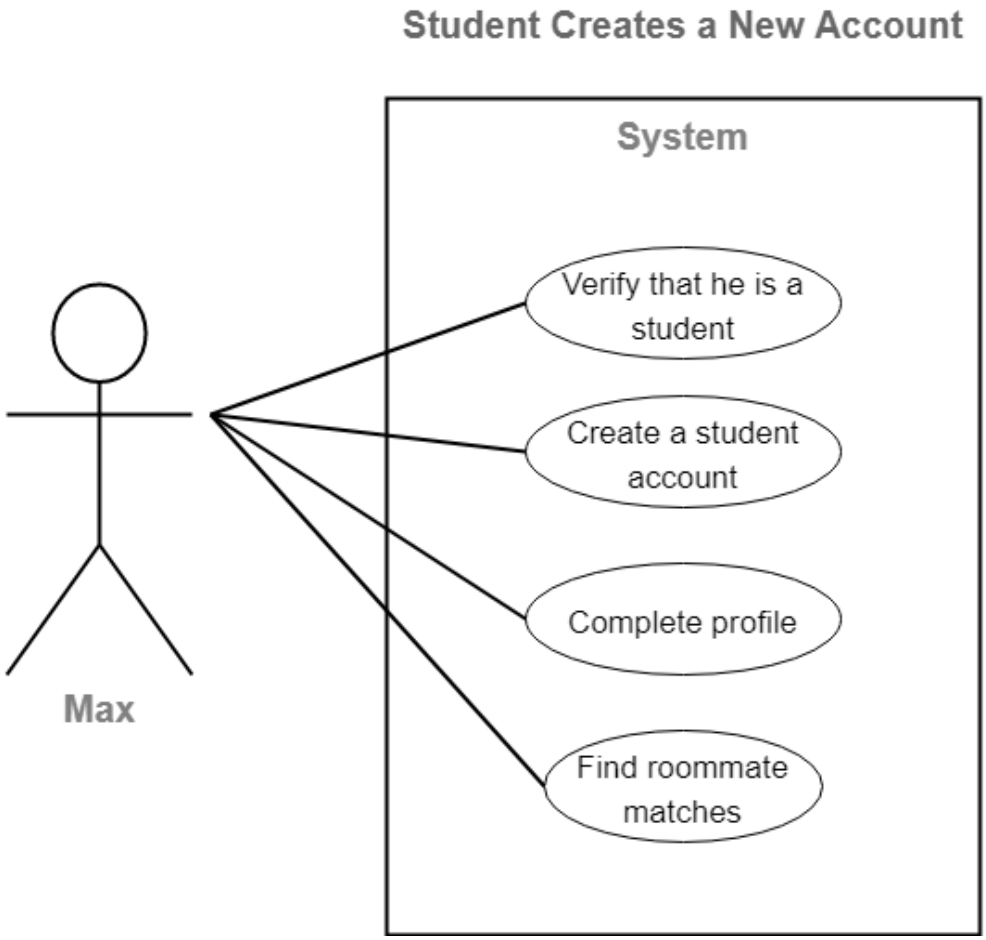
Title:	Landlord User Fails to Respond
Actor(s):	Students and Carl
Description:	A group of students wants to find roommates from the same school and same hobbies as them. After browsing through the internet they find DormMates. They create a student account and search for available housing listings, they find an apartment that suits all of their needs that is owned by Carl, a landlord. The group of students messages Carl through DormMates. When Carl fails to respond to the group within a limited amount of time his response rate rating is decreased.



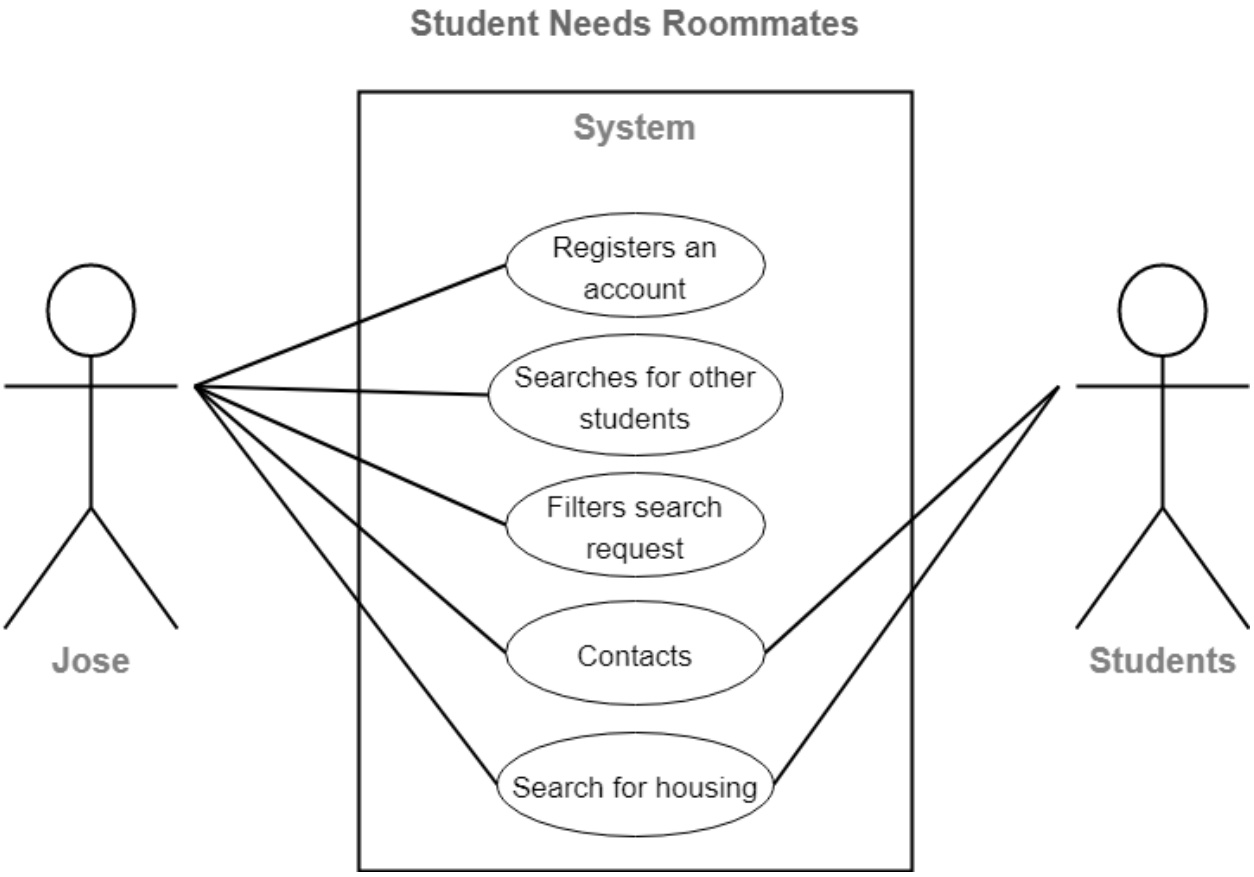
Title:	Creating a Review
Actor(s):	John and Kevin
Description:	John, a student who is renting a unit that is owned by Kevin, a landlord, is having some issues with the in-unit washing machine. In order for John to have the washing machine in his units fixed he had to reach out to Kevin multiple times and felt uncomfortable asking him to resolve the issues multiple times. On completion of his lease agreement, John leaves Kevin a negative landlord review.



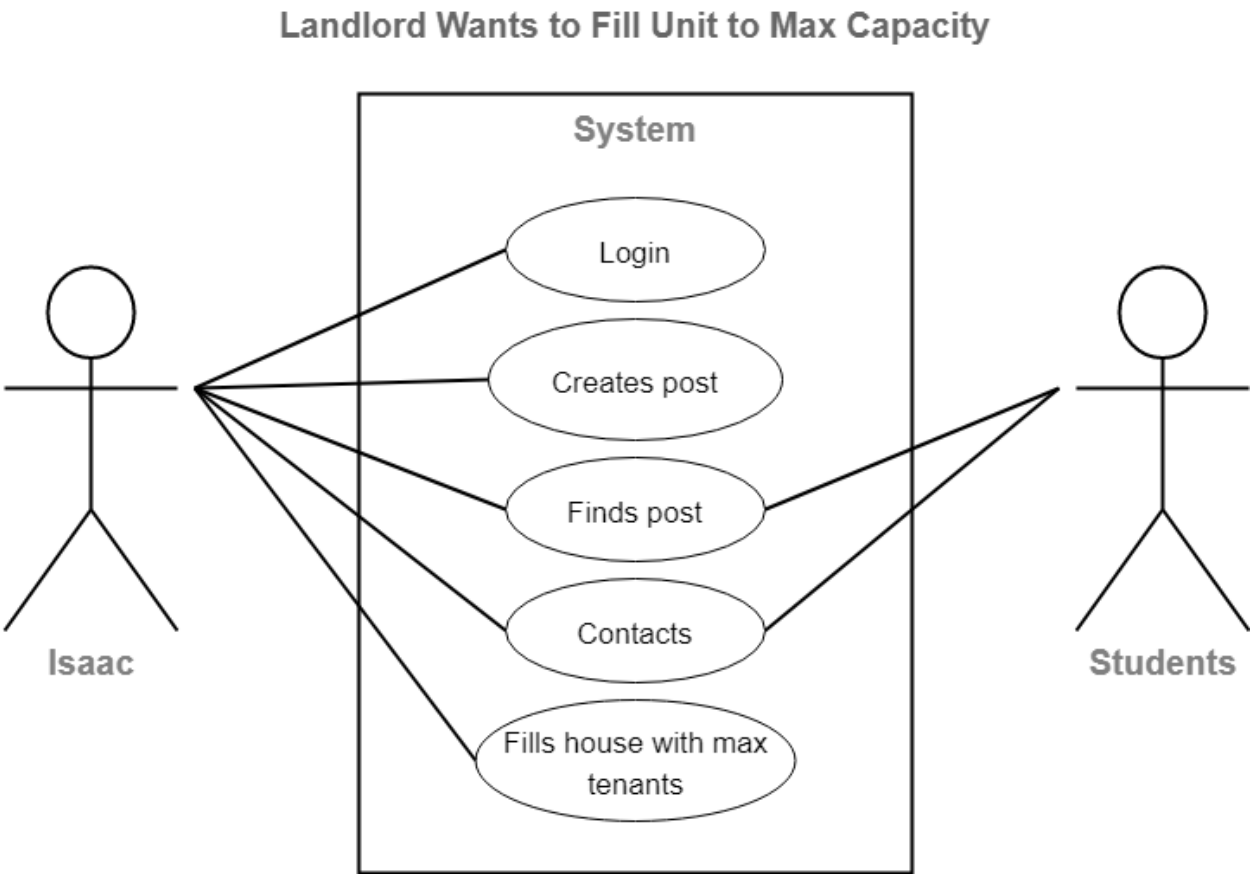
Title:	Student Creates a New Account
Actor(s):	Max
Description:	Max, a computer science student, is unhappy with his current roommate. Max's roommate doesn't have the same interests as him and doesn't even go to university. Max creates an account on DormMates and is able to instantly find other students who are in the same situation as him. In order for him to create a student account he needs to provide his .edu email and verify that he is a student. After that the admin verified his account and proceeded with his search.



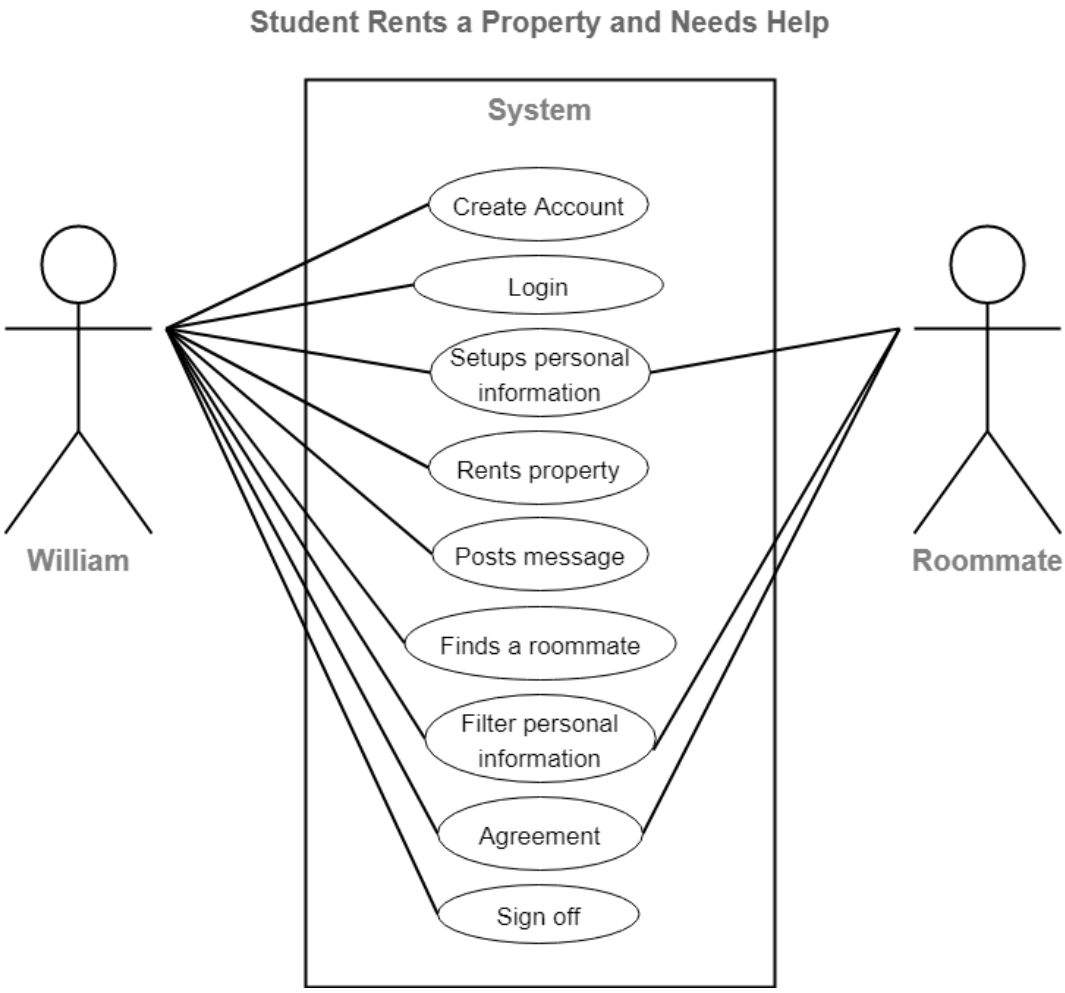
Title:	Student Needs Roommates
Actor(s):	Jose
Description:	An incoming freshmen student, Jose, is searching for a housing option that is within his budget. He is looking for 3 other students to increase the number of available housing options. Jose creates an account on DormMates and searches for roommates that match what he is looking for. He is looking for students that are on a similar schedule as him and have the same major as Jose. He finds and contacts other students that he wants to be roommates with. Jose and the other students search for a housing option within their budget together.



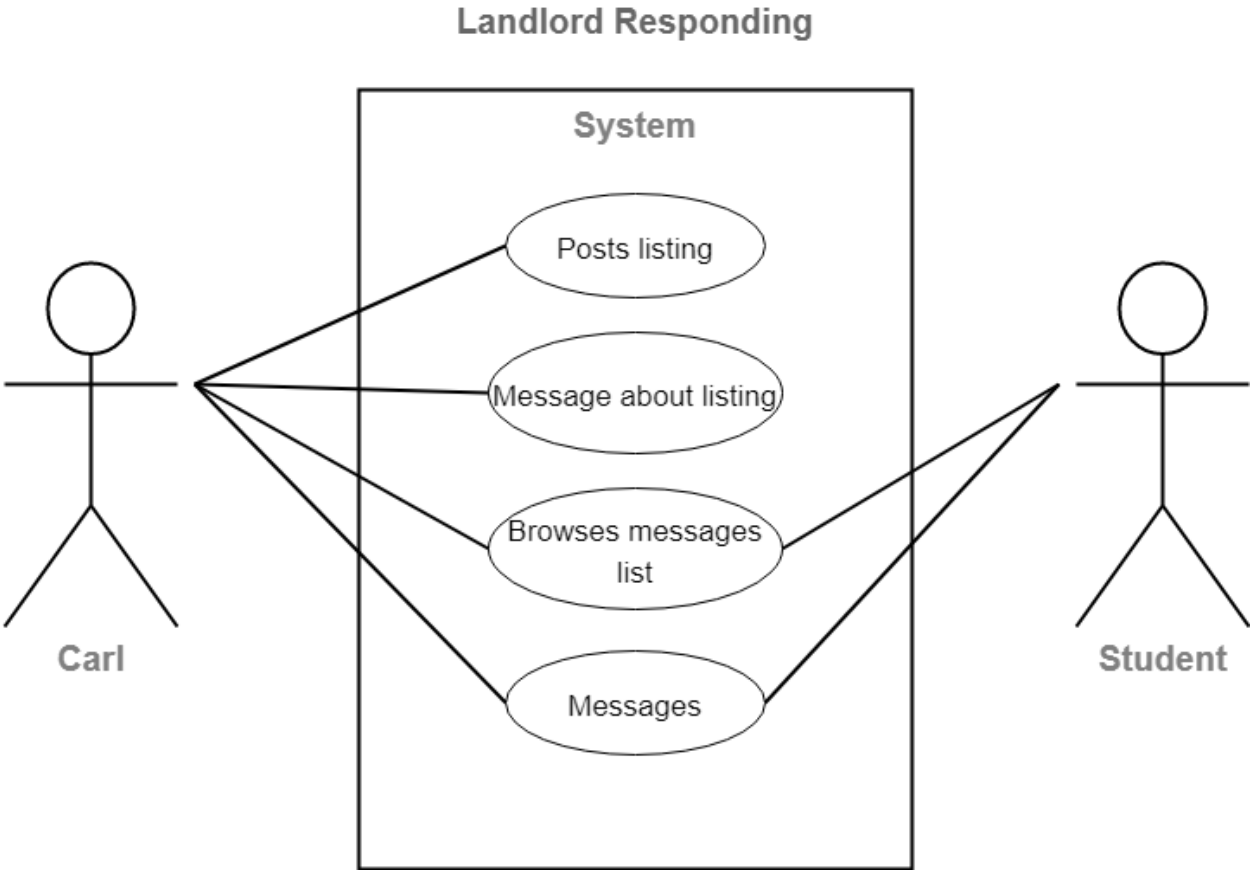
Title:	Landlord Wants to Fill Unit to Max Capacity
Actor(s):	Isaac
Description:	A landlord, Isaac, has logged on to the service and created a listing for his unit with all the available amenities that it offers. It has 3 rooms and can house up to 5 people if 2 students are willing to share a large room. Isaac wants to fill the unit with the maximum number of students that he can. Isaac is contacted by a group of students that found his post and is able to fill out the house with the maximum number of tenants.



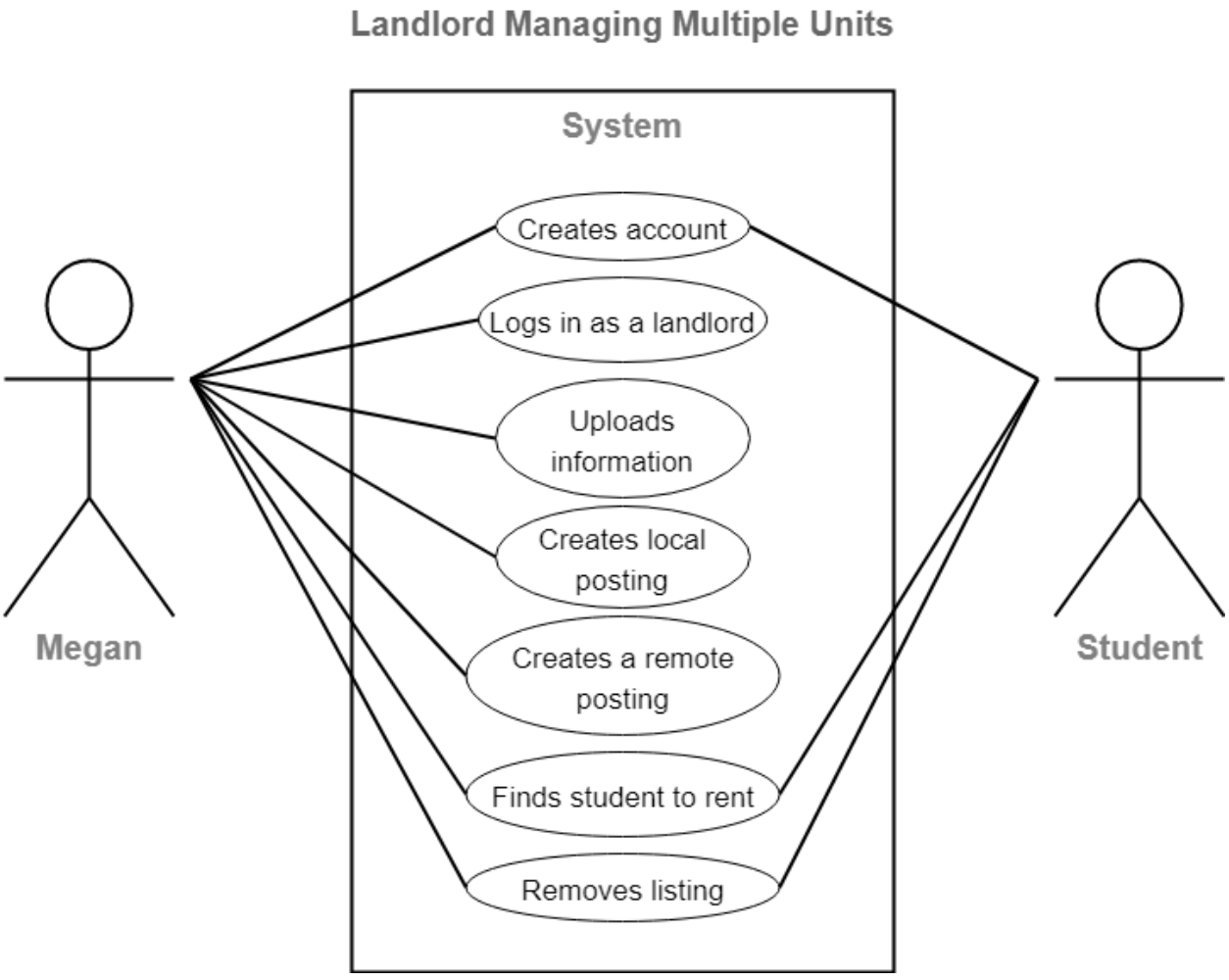
Title:	Student Rents a Property and Needs Help
Actor(s):	William
Description:	The transferring sophomore student, William, went on DormMates to search for housing and a roommate. He needs to find a roommate with a computer science major and attends San Francisco State University. He wants a roommate who can help with his assignments and has a hobby of playing computer games. William then finds a roommate who satisfies his preferences.



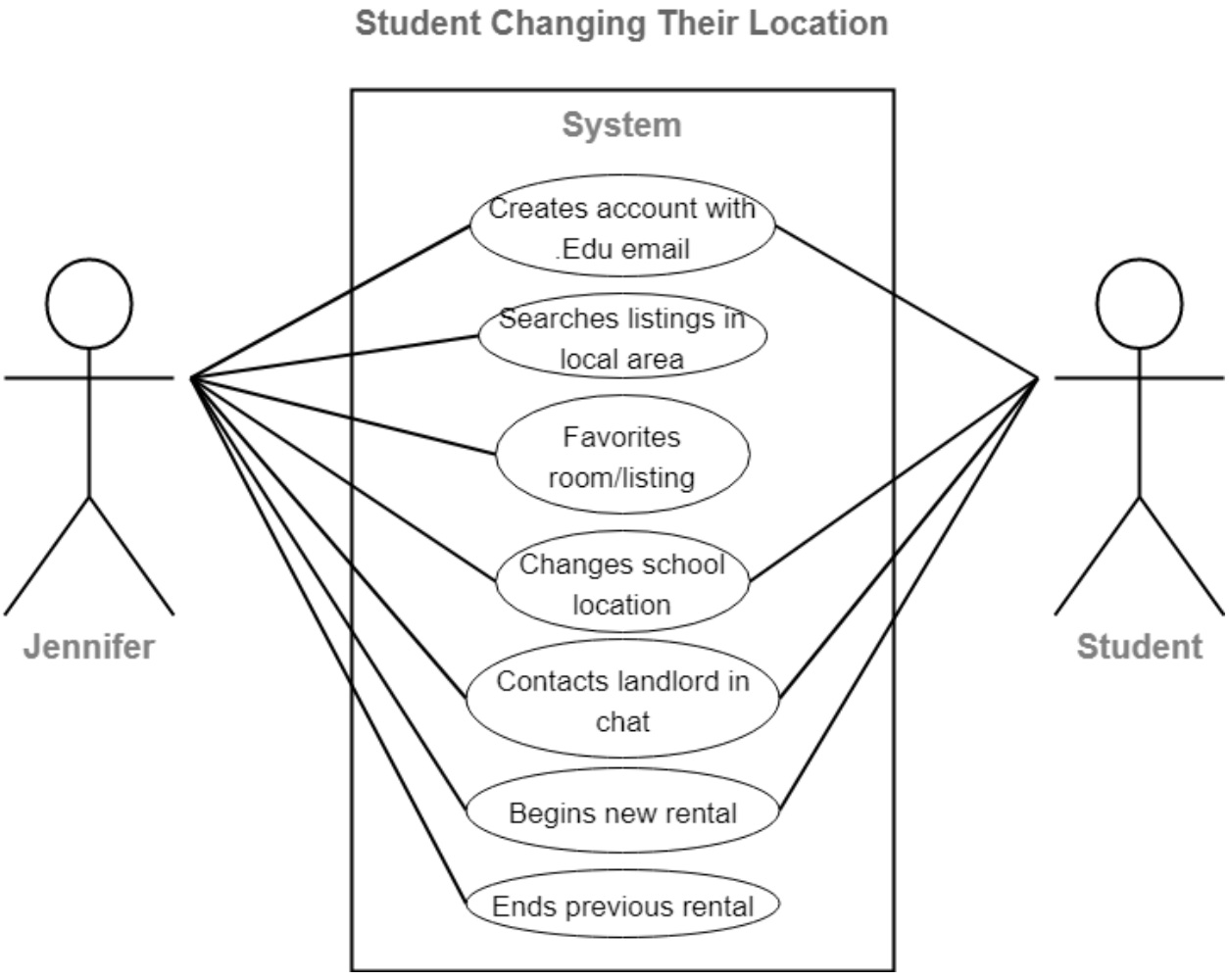
Title:	Landlord Responding
Actor(s):	Carl
Description:	Carl, a landlord who recently listed a rental unit on our application, browses through an array of interested students who messaged him about the listing. Carl is looking for 2 students to rent out his 2 bedroom in-law that are reliable and clean. He takes his time in choosing who to message since he wants students who fit his criteria. Carl decides which students and messages them.



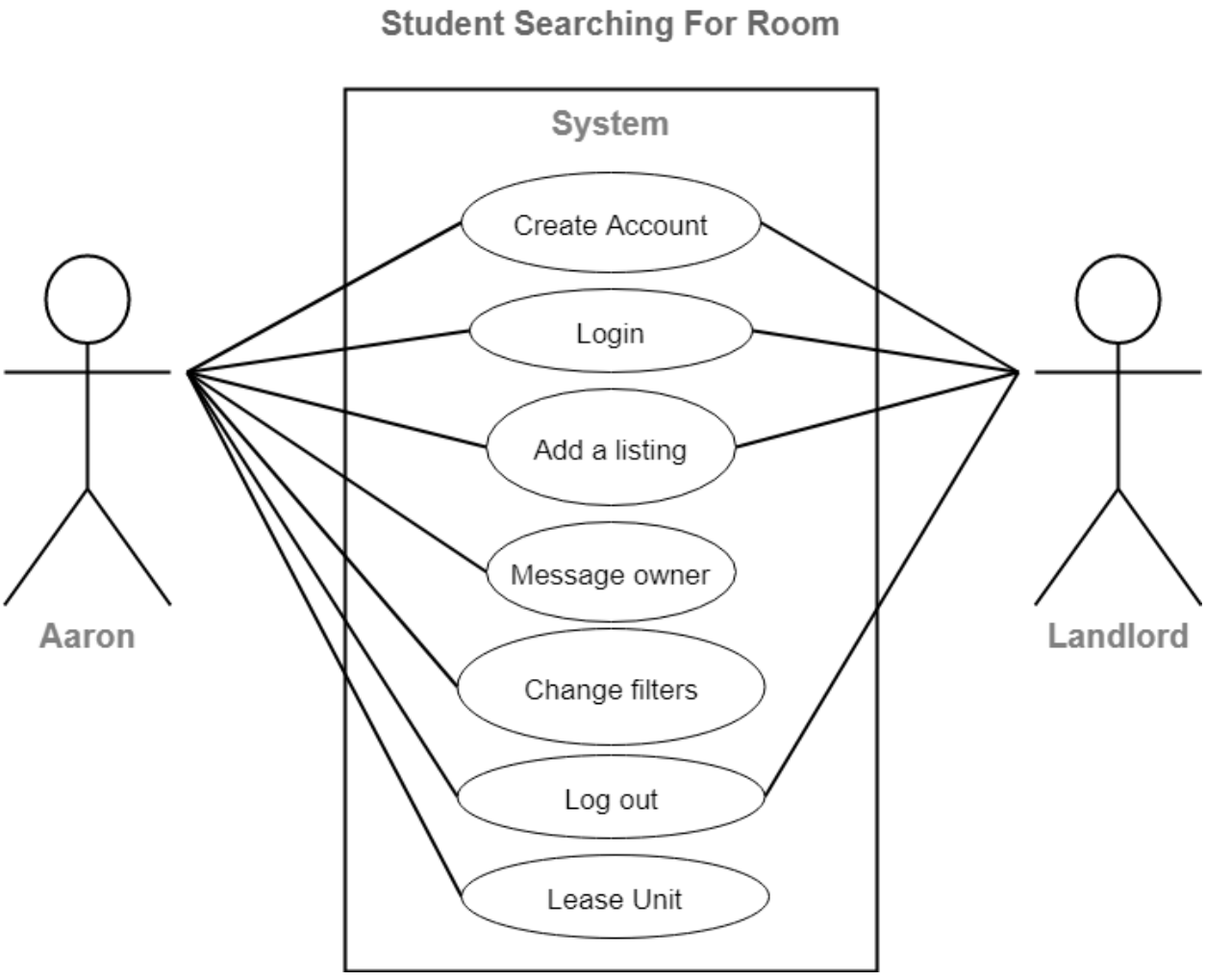
Title:	Landlord Managing Multiple Units
Actor(s):	Megan
Description:	Megan owns a multi unit dorm style building near the downtown area of her city, with two major colleges down the road from her units and more units in another large city in the same state. She creates multiple listings in each city and is able to house students she's housed in different areas and keep connections.



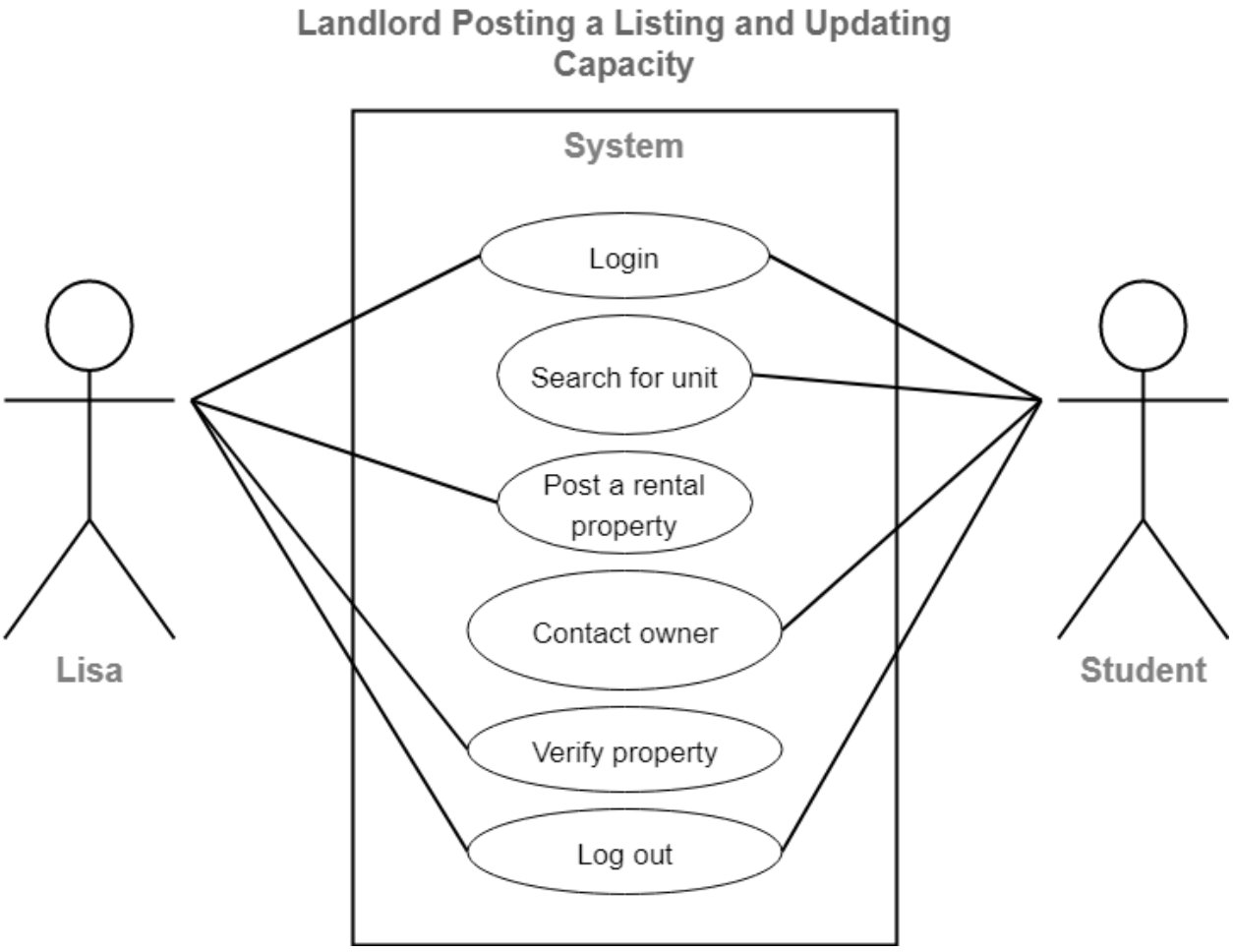
Title:	Student Changing Their Location
Actor(s):	Jennifer
Description:	Jennifer is moving to LA to start school, she is transferring from FAU. She signs up through the portal with her UCLA edu email and is immediately able to see listings in her area according to her student email.She then favorites the ones she likes. This connects her to more roommates that share her interest and degree plan. She was able to meet other UCLA students and live with them.



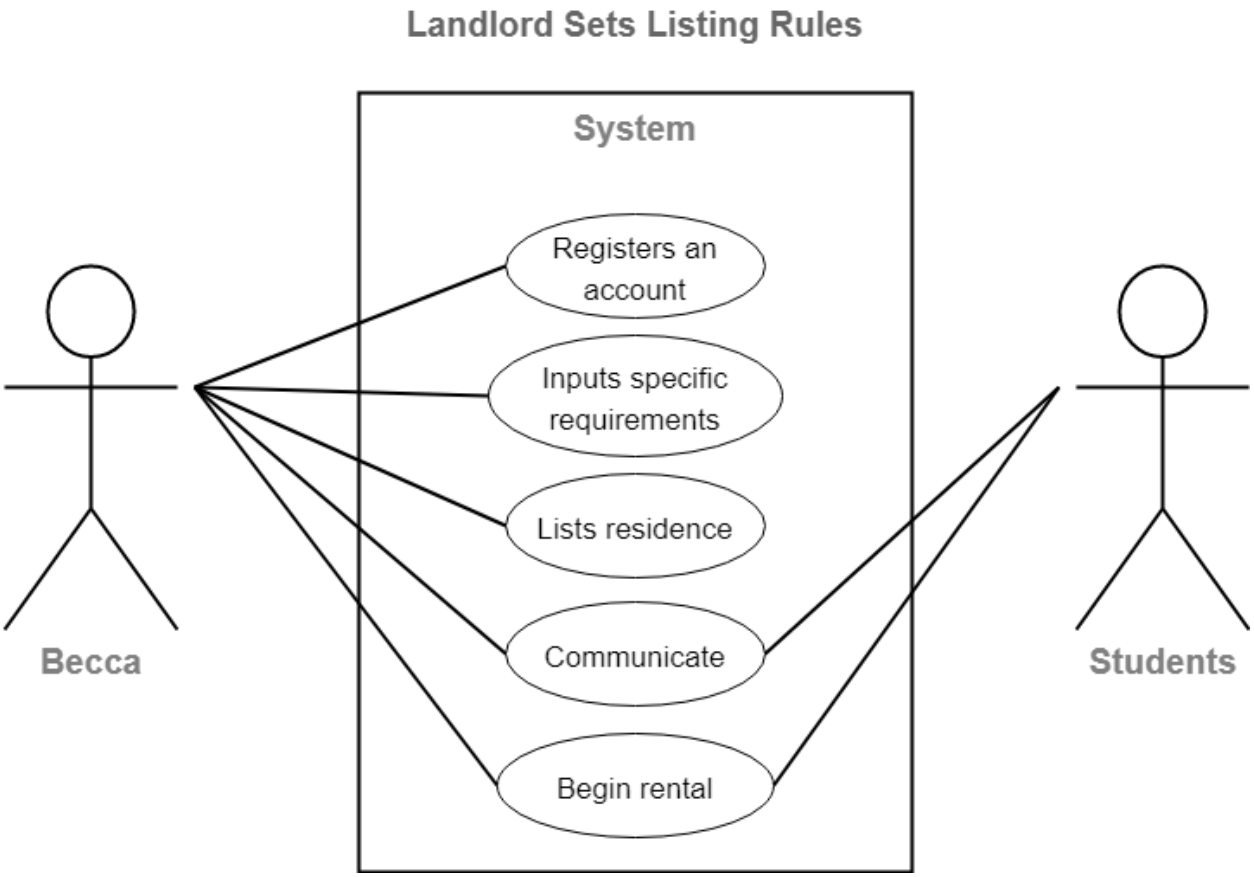
Title:	Student Searching For Room
Actor(s):	Aaron
Description:	Aaron recently moved to San Diego to start school and he is looking to rent a room with one more student near the city. He signs up through the portal with his UCSD account and he can see some listing near him and through his filters, he can look up available rooms in his price range and connect with more students from UCSD.



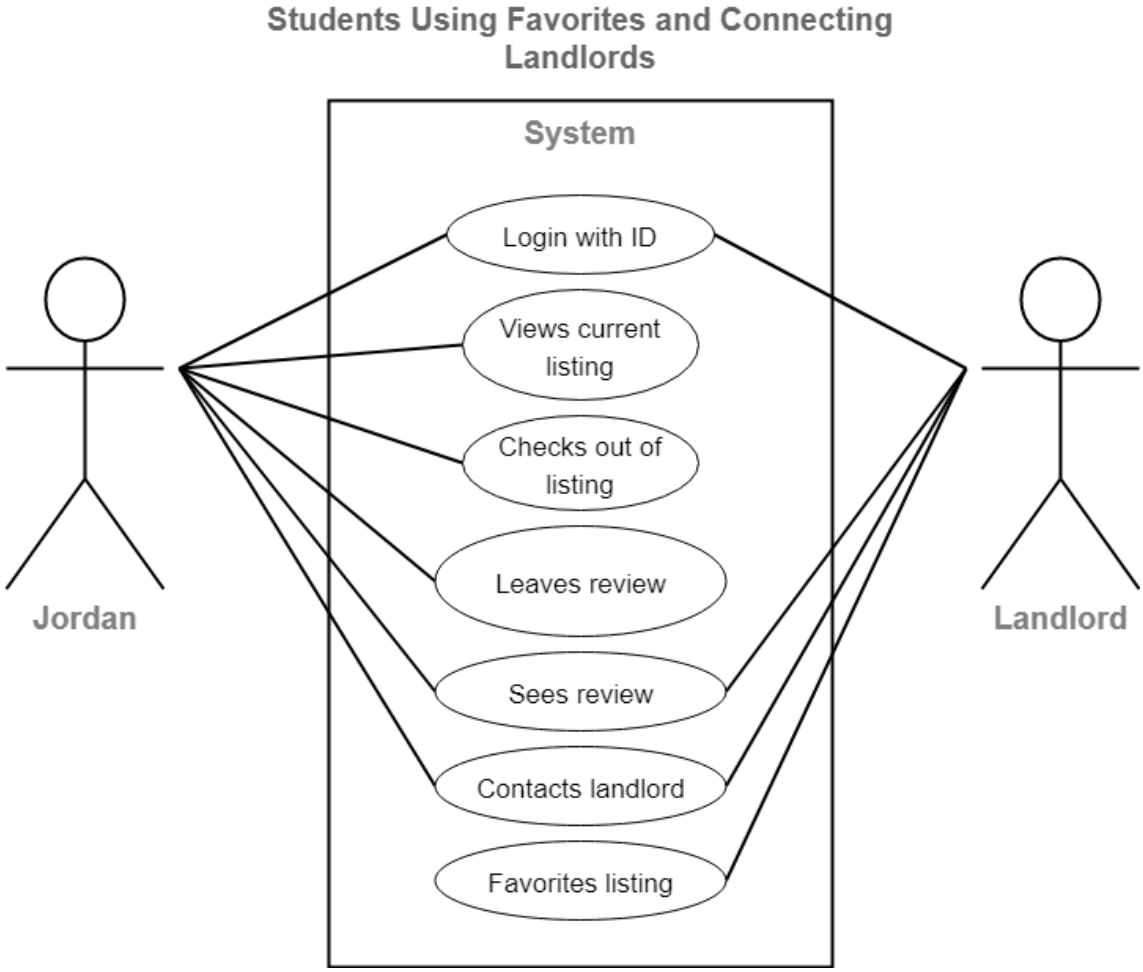
Title:	Landlord Posting a Listing and Updating Capacity
Actor(s):	Lisa
Description:	Lisa owns a small two floor building near the city and she has one room available for rent for two students only. She creates an account through the portal and lists her room and price for each student. After one student messages her and she agrees that the student can move in she can update the listing and change from 2 students to 1 student needed.



Title:	Landlord Sets Listing Rules
Actor(s):	Becca
Description:	Becca is renting out two bedrooms of her three bedroom house to students. Becca also enjoys a peaceful life with her own schedule and does not like a lot of noise around the house. She has had problems with finding two students who can coexist and agree to her rules. Becca lists her property through the portal, with specific requirements such as no guests, no parties; she then communicates with each student to ensure they agree to her rules. The students also talk to each other to make sure they can coexist peacefully.



Title:	Students Using Favorites and Connecting Landlords
Actor(s):	Jordan
Description:	Jordan is moving out of his current apartment, which was full and not posted. He logs in to update information about his living situation. Jordan then leaves a really good review about the landlord. His friend Greg really liked the area Jordan lived in and saw the review about the landlord. He is able to contact the landlord and get information on the listing and favorites for later.



List of Main Data Items and Entities

- Logged in user
    - Either a student or landlord user that is logged in to the website.
- User Type
    - Users can be either Student, Landlord, or Admin.
- Student
    - Users who signed up with a .edu email are denoted as students.
- Landlord
    - Users who signed up without a .edu email are denoted as landlords.
- Admin
    - User that has the ability to moderate the service.
- Guest
    - Users that have not registered to the website.
- Profile
    - Displays key attributes about a user.
- Listing
    - A representation of a housing unit that was posted by a landlord.
- Favorites
    - Specific listings that a user is interested in and wants to bookmark.
- Institution
    - The higher level education that a student is attending.
- Landlord rating
    - A compiled list of reviews and other attributes that rank a landlord.
- Match
    - Represents two users who may be compatible roommates.
- Chat
    - Students can form chats with other students who they are matched with.
    - Students can form chats with landlords of listings they are interested in.
    - Students can share listing links.
- User activity
    - Users have a history of login and chat.
- Map
    - The map is used to display listings.
- Location
    - Specified region that determines which listings are shown to a user.
    - Represents the physical location of a listing.

- Lease
    - Represents an agreement between one or more students and a landlord which signifies that a lease between the parties exists.
- Price
    - Users have to pay for the rent.
- Amenities
    - Represents the key features of a listing.

Functional Requirements

User

Unregistered User

1. An unregistered user can view listings near an institution of their choice.
2. An unregistered user can not view other student users on the platform.
3. An unregistered user can not view other landlord users on the platform.
4. An unregistered user can create a new account.
5. An unregistered user can create a student account.
6. An unregistered user can create a landlord account.
7. An unregistered user can view the Home page.
8. An unregistered user can view the FAQ page.
9. An unregistered user can view the About page.
10. An unregistered user can view the Terms of Service page.
11. An unregistered user can view the Privacy Policy page.

Registered User

12. A registered user should be able to login to with their email and password.
13. A registered user should be able to login to with their username and password.
14. A registered user should be able to logout of their account.
15. A registered user should be able to submit a forgotten password form.
16. A registered user should be able to submit a forgotten email form.
17. A registered user should be able to change their username.
18. A registered user should be able to change their profile photo.
19. A registered user should be able to change their email address.
20. A registered user should be able to change their password.
21. A registered user should be able to change their university.
22. A registered user should be able to change their location.
23. A registered user should be able to change their password.
24. A registered user should be able to change their email.
25. A registered user should be able to change their name.
26. A registered user should be able to change their age.
27. A registered user should be able to change their gender.
28. A registered user should be able to delete a chat.
29. A registered user should be able to favorite a chat.
30. A registered user should be able to verify their account via email.

Student User

31. A student user should be able to edit their personality.
32. A student user should be able to edit their schedule.
33. A student user should be able to edit their hobbies.
34. A student user should be able to edit their major.
35. A student user should be able to edit their social media links.
36. A student user must have an edu email.
37. A student user must have a completed profile.
38. A student user should be able to rate landlords.
39. A student user should be able to message another user.
40. A student user should be able to add other student users to a chat.
41. A student user should be able to search for listings.
42. A student user should be able to view a general location of a listing on a map.
43. A student user should be able to interact with a map to view more details about a listing.
44. A student user should be able to refresh the listings shown on a map.
45. A student user should be able to sort listings by price.
46. A student user should be able to sort listings by the date they were posted.
47. A student user should be able to sort listings by available rooms.
48. A student user should be able to filter listings by distance from university.
49. A student user should be able to filter listings by price.
50. A student user should be able to filter listings by room numbers.
51. A student user should be able to filter listings by washer and dryer being available or not.
52. A student user should be able to filter listings by wifi being available or not.
53. A student user should be able to filter listings by closet space.
54. A student user should be able to filter listings by bathroom count.
55. A student user should be able to filter listings by whether or not they are furnished.
56. A student user should be able to filter listings by whether or not they have a whiteboard.
57. A student user should be able to filter listings by whether or not they have a living room.
58. A student user should be able to filter listings by whether or not they have a kitchen.
59. A student user should be able to filter listings by whether or not they have parking.
60. A student user should be able to filter listings by whether or not they have a patio.
61. A student user should be able to filter listings by whether or not parking is included.
62. A student user should be able to filter roommate selections by personality.
63. A student user should be able to filter roommate selections by schedule.
64. A student user should be able to filter roommate selections by hobbies.
65. A student user should be able to filter roommate selections by major.
66. A student user should be able to filter roommate selections by gender.
67. A student user should be able to filter roommate selections by age.
68. A student user should be able to filter roommate selection by budget.
69. A student user should be able to view other student user profiles.

70. A student user should be able to favorite a listing.
71. A student user should be able to report a listing.
72. A student user should be able to report another student user.
73. A student user should be able to report a landlord user.
74. A student user should be able to view the student dashboard.
75. A student user shall be able to match with another student

Landlord User

76. A landlord user should be able to post listings.
77. A landlord user should be able to edit a listing.
78. A landlord user should be able to delete a listing.
79. A landlord user will be able to add descriptions to listings.
80. A landlord user will be able to view student profiles.
81. A landlord user will be able to respond to student users.
82. A landlord user will be able to see their own reviews.
83. A landlord user will be able to repost their listing once expired.
84. A landlord user will be able to change a listing's price.
85. A landlord user will be able to change a listing's photos.
86. A landlord user will be able to change a listing's location.
87. A landlord user will be able to change a listing's available rooms.
88. A landlord user will be able to change a listing's description.
89. A landlord user will be able to view historical listing data.
90. A landlord user should be able to view the landlord dashboard.
91. A landlord user should be able to pay to promote a listing.
92. A landlord user should be able to pay to post a listing.

Administrator User

93. Admin users should be able to approve listings created by landlords.
94. Admin users should be able to deny listings created by landlords.
95. Admin users should be able to disable landlord user accounts.
96. Admin users should be able to disable student user accounts.
97. Admin users should be able to view user reports.
98. Admin users should be able to view listing reports.
99. Admin users should be able to respond to user reports.
100. Admin users should be able to respond to listing reports.
101. Admin users should be able to edit user profiles.
102. Admin users should be able to feature a listing.
103. Admin users should be able to alter a landlord's rating.
104. Admin users should be able to alter a student's rating.

Non-Functional Requirements

Functionality

1. The website should utilize all tools and frameworks approved by the CTO.
2. The website should be easy to use and intuitive.
3. The website should have a simple and non-cluttered interface.
4. The website's interface should be uniform across all pages.
5. The website should be responsive across all modern devices.
6. The website will use Amazon Web Services for deployment.
7. The website will use Amazon Web Services for its database.
8. The website should use HTTPS for all requests.

Security

9. Users must authenticate themselves before accessing any protected pages.
10. Users must authenticate themselves if their cookie is expired.
11. The student dashboard page must only be available to verified student users.
12. The landlord dashboard page must only be available to landlord users.
13. Registered users should be able to view their own chat messages.
14. Registered users should be able to send messages only within their group.
15. All sensitive information must be encrypted before stored in the database.

Privacy

16. Only registered users will be able to view all listings.
17. Only registered users will be able to view students.
18. Landlords will not have access to viewing other landlords.
19. Landlords will not be able to search student data.
20. Landlords will not be able to search landlords.
21. Registered users' chat messages should remain private.

Legal

22. All users must accept the terms and service policy before creating an account.
23. All users must accept the privacy policy before creating an account.
24. All landlords must prove ownership of a listing.
25. The website must have a copyright notice.
26. The website must have a privacy policy notice.

27. The website must have a terms and conditions notice.
28. The website must have a cookie notice.
29. All content uploaded to the site must be owned by the user who is uploading it.

Performance

30. The frontend must have processes in place that prevent it from being offline.
31. The backend must have processes in place that prevent it from being offline.
32. The website load time should be within industry standard requirements.

System Requirements

33. The website shall work up to Version 91.0.4472.106 Google Chrome.
34. The website shall work up to Version 14.0.03 Safari.
35. The website shall work up to Version 91.0.864.48 Microsoft Edge.
36. The website shall work up to Version 85.0 of Mozilla Firefox.
37. The website shall work up to Version 11.0 of Android.
38. The websites shall work up to Version 14.6 of IOS.
39. The website will be supported in English language.

Marketing

40. The website should follow SEO best practices.
41. Each page on the website shall have the logo next to the navigation bar.
42. Each page will be clear and easy to navigate for new visitors.
43. Each user shall be able to connect their account with their social media platforms.

Content

44. The website will have a navigation bar.
45. The website navigation bar will direct users to different pages on the website.
46. The website pages will have a footer.
47. The website will have a scalable map.
48. The website should give registered users the option to private message.

Scalability

49. The website should be capable of handling a large number of listings.
50. The website should be composed of a frontend and backend which are separate codebases.
51. The website shall be able to handle a large number of users.
52. The chat rooms shall be able to handle a large number of users.

Capability

53. The website should process all requests as expected by the user.
54. The website should respond with a descriptive error if one occurs.
55. The website should alert users when they are about to leave the site.

Look and Feel

56. The navigation bar should have a logo.
57. The navigation bar should have a plain dark color background.
58. The navigation bar should have a light shade hover color button.
59. The footer should have a logo.
60. The footer should have a sitemap with all site pages.
61. The website should have a plain color layout.
62. The website should have a simple layout.
63. The website will have a readable font.
64. The website elements fonts will be uniform.
65. The website elements will be continuous.
66. The website's pages will be scrollable in the vertical axis.
67. The font should be roman new times.
68. The feeling should be friendly.
69. The website should not be repetitive.
70. The website should be easy to traverse.
71. Pages should be instant loaded.
72. The private account page should be easy to find.
73. The private chat should be easily identifiable.
74. The private chat font will be easy to read and uniform.
75. The map should be easily identifiable.
76. The map key will be easily identifiable.
77. Profiles will clearly display a user’s role.
78. The post should be easily identifiable.
79. The forum should be easily identifiable.
80. The buttons should be easily identifiable.
81. Listing filters should be easily identifiable.

Coding Standards

82. All code must be reviewed before it is merged with any of the three main branches.
83. All code must be submitted via pull requests.
84. All code must be pushed to proper branches.
85. All code should be documented.
86. All code should be organized.

87. There should be no repetitive code.
88. There should be no unused code.
89. All code should have in-line comments when needed.
90. The code should have a uniform formatting style.
91. The backend code should be an object-oriented programming paradigm.
92. The backend must implement methods to prevent SQL injection.

Availability

93. The frontend must be online at all times.
94. The backend must be online at all times.
95. The website is updated if and only if code is pushed to the master branch.
96. The website will resync if a loss of connection occurs.
97. The website shall display error messages when errors occur.
98. The website will be managed on a PST time zone.

Cost

99. Amazon web service's server is free.
100. Amazon web service relational database is free.
101. Server must not exceed the free tier.
102. Server maintenance is free.

Storage

103. Store users profile data on the database.
104. Store landlords listings on the database.
105. Remove listings from the database after it has been deleted by the user.
106. Store up to 60 days of inactive listings(incase user wants to repost)
107. Remove listings from the database after 60 days of inactivity.
108. Repost will restart the 60 day clock of storage time.
109. Store students' chat history on the database.
110. Store usernames on the database.
111. Store emails on the database.
112. Store passwords on the database.
113. Store landlord information on the database.
114. Store landlord photos on the database.
115. Store Students photos on the database.
116. Store error logs on the database.

Expected Load

117. The website will be able to handle as many users as Amazon Web Services can support.
118. The website will be able to handle as many listings as Amazon Web services can support.

Competitive Analysis

	Craigslist URL	**Roomiematch** URL	**Roommates** URL	**Facebook** URL	**Forrentuniversity** URL
Strengths	• Intuitive • Simple • Many listings • Many filters • Speed • Save listings • Cross platform	• Security and safety focus • Human moderation	• Identity verification • Profile matching • Easy to use • Cross platform	• Popular • Organized • Easy to use • Data integrated Groups • Dedicated community	• Easy to use • Strong search system • Lists apartments and houses
Weaknesses	• Unregulated • No internal chat • Repetitive listings • Lacks design • Not visual based	• Locked behind paywall • Only focused on roommates • Website not intuitive	• Unclear listings • Listings locked by paywall	• Forums posts from top to bottom • No specific search for forums on page	• Generic landing page • Only focused on listings • No chat or student features • Not moderated • Unsatisfying user experience
Pricing	• $5 apartment listings in Boston, Chicago, and NYC areas	• $19.95 per year for pro roommate search	• $6/3 day trial • $20/month • $30/2months	• Free	• Free to post listings
Target Market	• Anyone	• Anyone searching for roommates	• Anyone searching for roommates	• Anyone	• University students
Onboarding experience	• Simple and fast	• Complicated • Too many questions	• Lots of steps • Too many questions	• Simple and fast	• None

Feature	**Craigslist** URL	**Roomiematch** URL	**Roommates** URL	**Facebook** URL	**Forrentuniversity** URL	**DormMates**
Shows user activity	+	-	++	++	-	++
Requires .edu email	-	-	-	-	-	++
Chat	-	+	+	++	-	++
Landlord rating system	-	-	-	-	-	++
Roommate matching	-	+	+	-	-	++
Moderated listings	-	++	+	++	-	++
Map	+	-	+	++	++	++
Listing filter	+	-	-	++	+	++

Legend: - Feature does not exist, + Feature exists, ++ Feature is superior

Summary of Competitive Analysis

When researching current companies in this market, we were not able to find any service that caters directly to university students. While some services had some of the features DormMates seeks to offer, none had all of them in one easy-to-use platform. According to our research, we found three common themes across all of our competitors: usability, price, and student-focused features. While some competitors do prioritize usability, it does not seem that it is a main focus for our competitors and most did not have a simplistic and intuitive interface. In addition to this, competitors also placed a large emphasis on turning a profit. All but one of our competitors offered some type of paid service or locked key features such as verified listings and roommate matching behind a paywall. Lastly, none of the competitors on the market cater directly to students through student-focused features and we believe that this opens up a huge opportunity for us.

DormMates believes that by focusing on the three themes of student-focused features, usability, and price we can create a service that has a substantial edge over our competitors. Students will be at the core of our service and everything they see will be based on the university they are attending. All roommates that student users see on our platform will go to the same university as them and have similar interests. Similarly, all listings that students see will be near their university. In terms of usability, we will place the user-experience above all else by creating a simple and intuitive website that allows our users to find exactly what they are looking for. When it comes to price, we will not lock down any “pro” features behind a paywall. We believe that by allowing students to use all of our features we can generate a profit by retaining the most amount of users possible. The thing that allows us to generate a profit are the landlords. We will allow landlord users to pay us a fee in order to bring more attention to their listings through the user of featured listings. In addition to this, as our student user-base grows, we plan on taking a fee for all listings that are posted on our platform.

High Level System Architecture and Technologies

Hosting

- Server Host: AWS EC2 1vCPU 1GB RAM
- Database Host: AWS RDS T2 Micro (MySQL 8.0.20)
- DNS: Cloudflare
- Email API: Mailjet

Server Software Stack

- Operating System: Ubuntu Server 20.04 LTS
- Web Server: NGINX 1.20.1
- Server-Side Language: Javascript
- Server Framework: Node.js 10.19.0
- Server Process Manager: PM2 5.1.0

Frontend Technologies

- Web Framework: Express 4.17.1
- CSS Framework: Bootstrap 5.0.0
- Templating Engine: Pug 3.0.2
- Interactive Maps Library: Leaflet 1.7.1
- Icons Library: FontAwesome 5.15.3

Backend Technologies

- Backend Framework: Express 4.17.1
- SSL Cert: Cloudflare
- Encryption: Bcrypt 5.0.1
- Sessions: express-session 1.17.2
- Sessions MySQL Store: express-mysql-session 2.1.6
- Image Middleware: Multer 1.4.2
- Real-time Communication: pusher-js 7.0.3
- Academic Email Middleware: Academic-Email-Verifier 3.0.3
- Email sending library: node-mailjet 3.3.4
- Authentication: Passport 0.4.1 (Local Strategy)

Additional Technologies

- IDE: vsCode, Sublime Text
- Web Analytics: Google Analytics

Checklist

Item	Status
Team found a time slot to meet outside of class	DONE
GitHub master chosen	DONE
Team decided and agreed together on using the listed SW tools and deployment server	DONE
Team ready and able to use the chosen back and frontend frameworks and those who need to learn are working on learning and practicing	DONE
Team lead ensured that all team members read the final M1 and agree/understand it before submission	DONE
GitHub organized as discussed in class (e.g. master branch, development branch, folder for milestone documents, etc...)	DONE

List of Team Contributions

Andrei

- Scheduled and conducted 5x meetings where the team worked on M1together.
- Worked with M1 editor by giving advice on how to format certain areas of the document.
- Worked with the team to create the executive summary.
- Worked with the team to create main use cases.
- Worked with the team to list maindata items and entities.
- Worked with the team to create initial list of functional requirements.
- Worked with the team to create list of non-functional requirements.
- Worked with the team to create the competitive analysis.
- Worked with backend lead to list the high-level system architecture and technologies.
- Created the checklist and made sure everything on the checklist was taken care of.
- Compiled the list of team contributions and complaints.
- Made sure that the entire team looked over the final M1 document and agree/understand it.

Meeka

- Attended all meetings that were scheduled by the team lead.
- Participated in each task discussed and written during the meetings such as the executive summary, main use cases, main data items & entities, functional requirements, non-functional requirements, and competitive analysis with the team together.
- Revised and edited the draft documents with assistance from the team lead.
- Created necessary diagrams and tables for the final document.
- Created and formatted the final document as the M1 editor.

Jonathan

- Task 1: Executive Summary:

The group together contributed to this, we all wrote it together and edited the paragraph. I contributed roughly 1/5 of this part along with the team. I edited some sentences and contributed some. We all collectively did this.

- Task2: Main Use Cases:

The team was assigned multiple use cases, I contributed 3 use cases and diagrams for the cases. I added a use case regarding 3 features. Landlord listings, Student use cases and Favorite feature. Roughly a paragraph each.

- Task 3: List of main data items and entities:

This task we worked as a team to finish. I contributed about 6 entities with the team as we brainstormed. Users/ Students/Landlords/Locations. The team collectively edited them too.

- Task 4: Initial list of Functional Requirements:

This task was a group effort during our meeting. We were all tasked with making a list of items and then compiled them together. I contributed roughly 30 functional requirements. These requirements were also edited later in the document.

- Task 5: List of Non functional requirements

This task was also a group effort during meetings. I contributed another 20~30 non functional requirements for this document. We were all tasked with coming up with a list. I made about 40 items but only 20 made it onto the list.

- Task 6: Competitive Analysis:

This task was assigned to the team to compile and come up with an analysis. This task was mostly done by the team. I contributed verbally as they edited the document. I did not contribute/ write on the document as I was assigned Task 7. We all did talk about each item and column.

- Task 7: High level Architecture

I was tasked with this, I met up with the Team Leader and we discussed these technologies and researched on how to implement them. These were decided before hand and Andrei and I talked more in detail about what items we wanted to use and organized them. I did discuss some features we will need to consider, and later on Andrei found some more. So I'd say this task was done 50/50 by Andrei and myself.

Alexandre

- Helped brainstorm ideas to be used for Task 1 Executive Summary
- Helped complete Task 1 Executive Summary
- Created Main Use Cases to help complete Task 2 Main Use Cases
- Helped determine the Data Items and Entries for Task 3 Data Items and Entries completion
- Helped figuring out what would the Functional and Non-Functional requirements are to complete Task 4 Functional Requirements and Task 5 Non-Functional Requirements
- Researched competitors to compare our product for Task 6 Competitive Analysis
- Helped create explanation of wh

Jimmy

- I met the team leader's agenda everyday by responding with emoiji on discord.
- I usually prepare ideas before the tasks so that I can talk to my group.
- I do respond to my group when they ask who did what before.
- For task 1, I wrote a paragraph and bullet points for brainstorm's executive summary.
- I had never missed a meeting and respond to my team leader's who contribute what.
- I wrote the entire non-functional's idea and some ideas for functional's idea for brainstorm.
- For task 2, I wrote a use case in a paragraph and a case diagram.
- I went to checked with team leader if my case diagram is acceptable and appropriate.
- For task 3, I went to wrote user activity and map for main data items and entities.
- I came up idea about agreement for lease.
- For task 4, I contribute one fourth of the functional requirement.
- For task 5, I contribute one third of the non-functional requirement.
- I went to question backend code format and front end looks.
- For task 6, I wrote the almost entire important points for facebook for competitive analysis.
- I went to talk about what the competitor does.

Sayed

- Contributed on non-functional requirements
- Contributed on competitive analysis doing research forrentuniversity.com
- Contributed on functional requirement
- Helped on topic brainstorm executive

Ahad

- Worked on rewriting and rewording the executive summary
- Worked with the team and contributed multiple functional requirements
- Worked with the team and contributed multiple non-functional requirements
- Worked on researching and completing the competitive analysis for all five competitors
- Researched and provided data for competitor features
- Tested competitors website processes and account creating flow