# Sinking Ship Development Studios

## Requirement Specification

Submitted to:
Decima Technologies

2018-12-05

**Purpose of the Project** **3**

**Stakeholders** **4**

**Constraints** **5**

**Terminology** **5**

**Relevant facts and Assumptions** **6**

**Functional Requirements** **7**

**Business Use Cases** **9**

**Business Data Model** **26**

**Scope of the product** **30**

**Non-functional Requirements** **39**

**Project Issues** **42**

**Costs** **46**

# Purpose of the Project

## Problem Statement

There are currently many different sites that students and homeowners must use to find an somewhere to live or to post an apartment to rent. Finding the right apartment can be a difficult task, especially if multiple roommates are required to sign the lease at the same time. The purpose of this product is to create a centralized system that allows students to easily find apartments to rent during the school year and provide homeowners with an easy and useful method to list apartments that are available to rent. Students aren't responsible for finding their own roommates and can easily search for the apartment that best fits their needs.

## Goals of the Project

**Purpose:** To make it easier for students and homeowners to rent apartments
- **Advantage:** To provide convenience to mobile users
  - **Measurement:** Number of students and homeowners using the mobile app
- **Advantage:** Creates one central location to check for renting information
  - **Measurement:** App and website bounce rate

**Purpose:** To make it safer for students and homeowners to rent apartments
- **Advantage:** Landlords can see information about who they're renting to, students who they're renting from, and students who they're renting with
  - **Measurement:** How often tenants or homeowners choose not to pick a tenant to rent to or live with
- **Advantage:** Can be paid using pre-authorized payments
  - **Measurement:** How many students use pre-authorized payments
- **Advantage:** Can be paid through our trusted website
  - **Measurement:** Number of students and homeowners who have used the service to make or receive at least one payment

# Stakeholders

Internal stakeholders include:

- People who came through Decima will be involved in the project as they are the ones who will directly receive the project once it is done. As well they are providing $2.8M CAD funding for the project.
- Developers who will be the team involved in creating the application itself. We will require at least 2 of each of the following: A UI designer; Android developers; iOS developers; and web developers
- A customer care team who will be the support team involved with interacting with the customers. This will include the responsibilities of the adjudicating the dispute / claim system as well as selling and advertising the app to different companies, groups, and organizations.

External stakeholders include:

- Users who will be interacting with our system, there will be students who will be renting, as well as a sponsors group who can be responsible for paying the student's rent. Homeowners who will be the ones to post their house ads using the system.
- Peterborough municipal government who Decima may be required to contact to pass through certain bylaws as well as paying municipal taxes.
- Banking companies for processing payment through the app, including: PayPal; Visa; Mastercard; American Express; BMO; and Interac.
- Hackers who pose threats to the safety of user's private information.
- Post-secondary institutes including Trent and Fleming, for providing proof of enrolment, as the system states that it is exclusive to student use.
- Competition coming from other apartment finding applications such as Kijiji, - Craigslist, and Trent Roommate finding groups on Facebook.

# Constraints

- Product must be available as a web application for use on any platform
- Product must be accessible and understandable by the target market Android and iOS devices
  Product must be targeted at university students and homeowners
- Development must be completed by February 2020 (1.5 year development cycle)
- Budget of $2.8 million CAD

# Terminology

Apartment: Any single rental unit that a homeowner lists for rent on the system

Blacklisted Account: The account of a user who has had multiple complaints or disputes. This account is considered inactive, as the account holder can no longer use this system

Complaint / Dispute: Any statement of dissatisfaction or any dispute from either a student or a homeowner during their time renting while using this system

Database System: A database that holds the account information for all users and the apartment information for all locations in the system

Homeowner: A person with an apartment(s) available for rent who wants to rent to students

Payment Schedule: A schedule provided to the student that outlines when they are to make rent payments to the homeowner

Sponsor: A person or entity that is responsible for the rental of an apartment on behalf of a student

Student: A person currently enrolled in post-secondary education looking for a place to rent during the time of their enrollment

# Relevant facts and Assumptions

Relevant Facts

- As of 2017, Trent University has a total enrollment amount of 8,940
- Trent has a large body of students that require temporary living situations
- We are the the first home/location rental application developed specifically for students in Peterborough

Business Rules

- Messaging system must remain secure and confidential
- Maximum number of rental offers on an account is 5
- Only 1 account allowed per registered Trent student
- Follow Google play store, and Apple app store terms of service
- Zero tolerance towards student/homeowner reviews that have proven to be obscene and/or profane

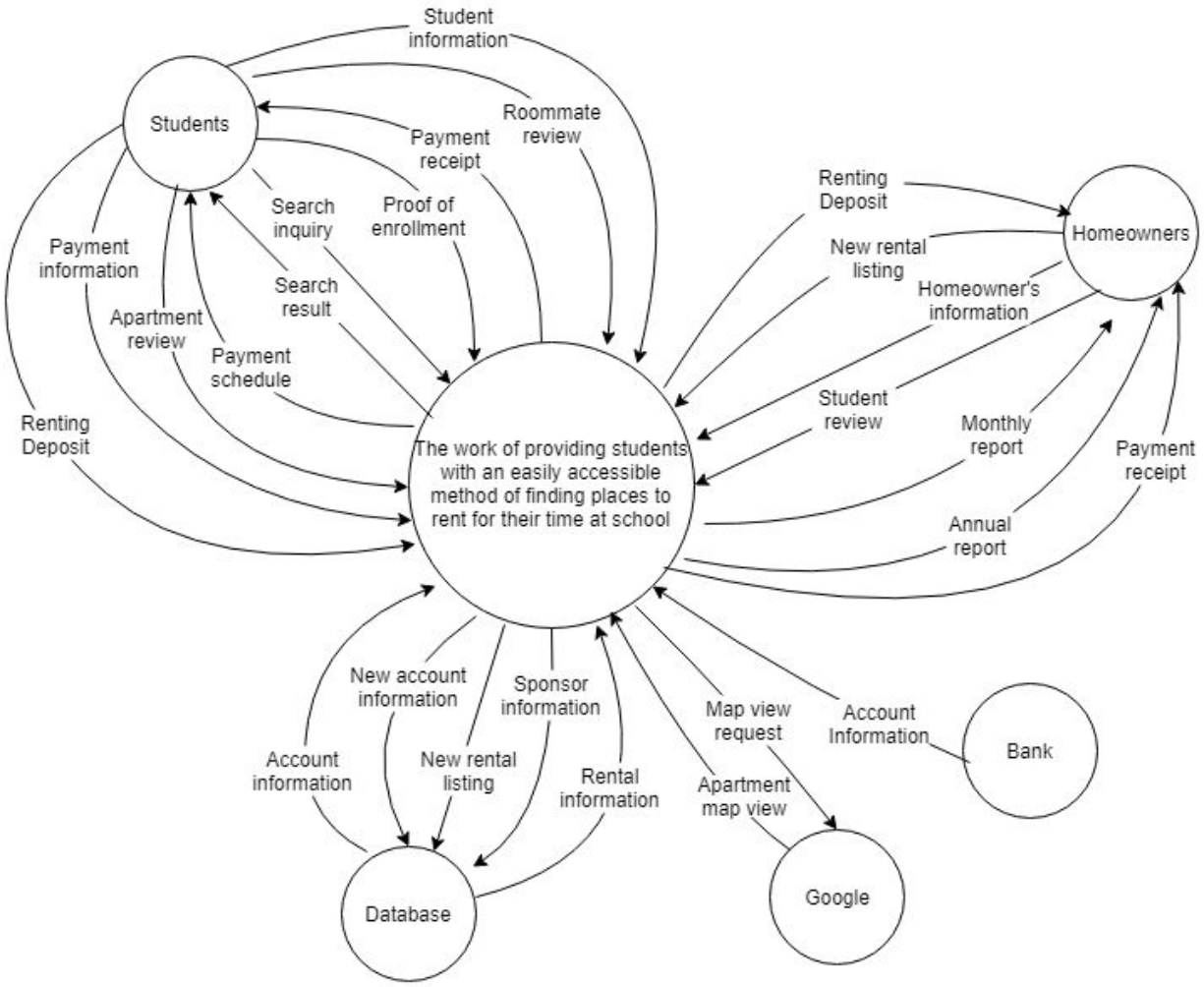
Assumptions

- Users will be fluent in using similar services such as kijiji and facebook marketplace
- Most consumers use a smartphone or device that can run web applications
- We will be able to find and hire somebody for each of the roles required to finish
- Customers will use product properly

# Functional Requirements

Work Context Diagram

The diagram below outlines the scope of the project and the work that needs to be done. It includes inputs and outputs of all business events that will take place within the project scope.



Work Partitioning

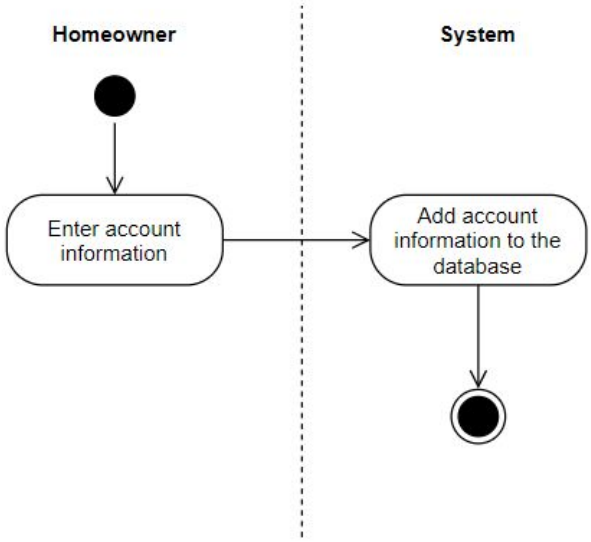
| Event Name | Input and Output |
|---|---|
| 1. Homeowner makes an account on the system | Homeowner information (in)<br>Account information to database (out) |
| 2. Student makes an account on the system | Student information (in)<br>Proof of enrollment (in)<br>Account information to database (out) |
| 3. Homeowner lists an apartment for rent | New rental listing (in)<br>Rental listing information to database (out) |
| 4. Student searches for an apartment to rent | Search inquiry (in)<br>Search results (out) |
| 5. Student pays rent | Payment information (in)<br>Payment receipt (out) |
| 6. Student sets up automatic payments | Desired payment schedule information (in)<br>Payment schedule (out) |
| 7. Automated payment is made (time-triggered event) | Payment receipt (out) |
| 8. Student writes a review for an apartment | Apartment review (in) |
| 9. Student writes a review for their roommate | Roommate review (in) |
| 10. Homeowner writes a review for a student | Student review (in) |
| 11. A complaint is filed by a student or homeowner | Complaint statement (in)<br>Account possibly blacklisted (out) |
| 12. Monthly report generated for homeowners | Rental information (in)<br>Monthly report (out) |
| 13. Annual report generated for homeowners | Rental information (in)<br>Annual report (out) |
| 14. Student adds a sponsor to their account to rent on their behalf | Sponsor information (in)<br>Updated student account information (out) |
| 15. Student wants to see apartment location on Google Maps | Request for map view (in)<br>Map view of apartment (out) |
| 16. Student rents a room | Renting deposit (in)<br>Renting deposit (out)<br>Rent receipt (out) |

# Business Use Cases

**Business Use Case Name:** Create a Homeowner account
**BUC #:** 1
**Trigger:** Homeowner information
**Pre-Condition:** A Homeowner has chosen to create a new account
**Active Stakeholders:** Homeowner (trigger), database

1. Enter required account details
2. Account information is added to the database

**Outcome:** Homeowner now has a homeowner account, and is able to login and access the full website.

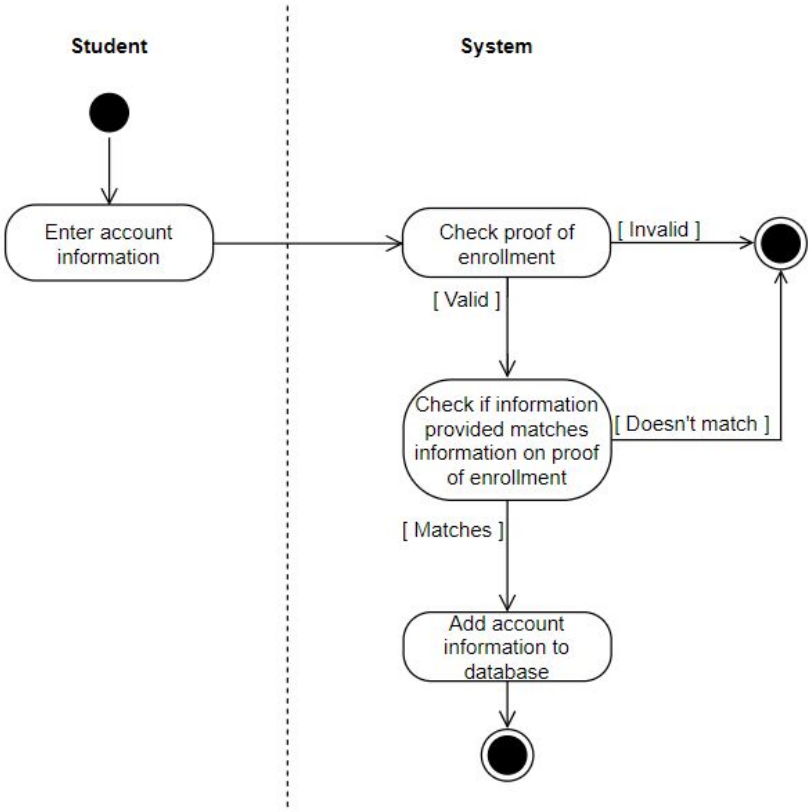



**User Story #1**
**Description:** As a homeowner I want to create an account for this service.
**Rationale:** I want to list my apartment/house.
**Source:** Discussion about BUC 1
**Fit Criterion:** The user provides valid information for their email address, phone number, and address. The account is successfully created.
**Dependencies:** email address, phone number, address
**Supporting Materials:** None

**Business Use Case Name:** Create a student account
**BUC #:** 2
**Trigger:** Student account information, proof of enrollment
**Pre-condition:** A student has chosen to create a new account
**Active Stakeholders:** student (trigger), database

1. Enter required account details, including proof of enrollment and other student information
2. Check that the information provided is valid
   2.1 Proof of enrollment must be current
   2.2 Provided information must match information on the proof of enrollment
3. Account information is added to the database

**Outcome:** The student now has a student account, and is able to login and access the full website.

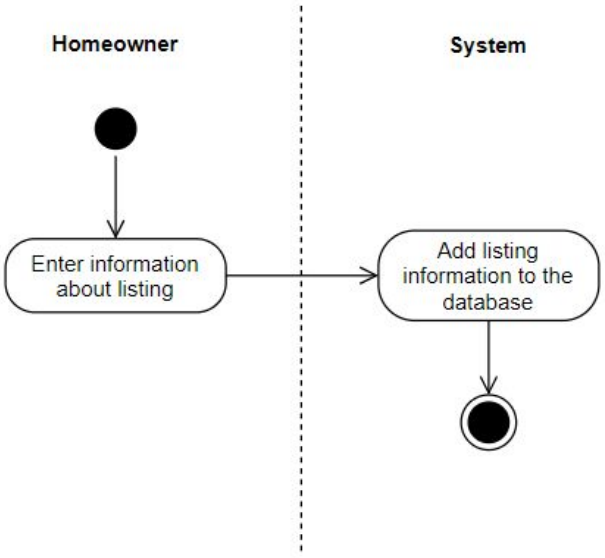



**User Story #2**
**Description:** As a student I want to create an account for this service.
**Rationale:** I want to rent an apartment/house.
**Source:** Discussion about BUC 2
**Fit Criterion:** The user provides valid information for their email address, phone number, address, and proof of enrollment. The account is successfully created.
**Dependencies:** email address, phone number, address, proof of enrollment.
**Supporting Materials:** None

**Business Use Case Name:** Create a new rental listing
**BUC #:** 3
**Trigger:** Rental information
**Pre-condition:** Homeowner making the listing already has an account
**Active Stakeholders:** homeowner (trigger), database

1. Required information is entered about the new listing
2. Listing information is added to database

**Outcome:** The listing is displayed as an available place to rent and shows up on the search page.

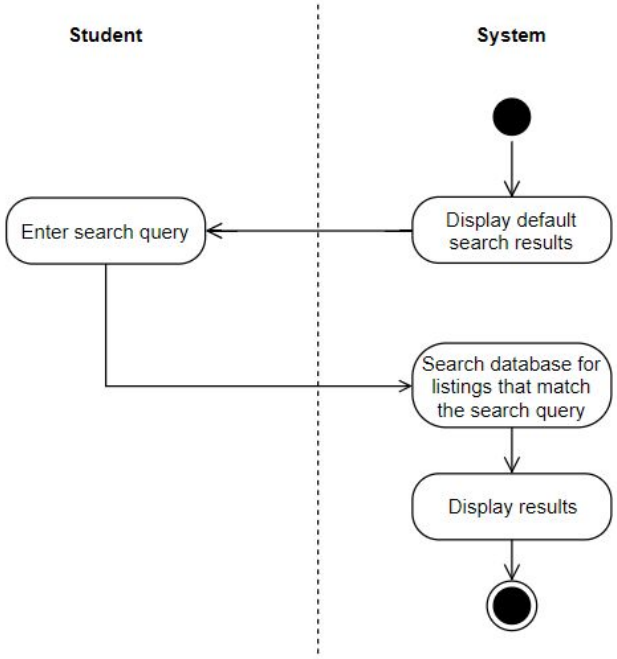



**User Story #3**
**Description:** As a homeowner I want to be able to list my apartment for students to be able to rent.
**Rationale:** I want students to pay rent to live in my house or apartment.
**Source:** Discussion about BUC 3
**Fit Criterion:** The apartment listing is successfully added in the system.
**Dependencies:** Listing information
**Supporting Materials:** None

**Business Use Case Name:** Display search results
**BUC #:** 4
**Trigger:** Students search inquiry
**Pre-condition:** User performing search has an account
**Active Stakeholders:** Student, homeowner

1. Display default search results - most recently posted listings first
2. Student enters search query, including optional filter search choices
3. Search database for listings that fulfill search requirements
4. Display relevant listings

**Outcome:** The student who performed the search can now see the listings that are relevant to what they want, and can choose different listings to look at from this page.

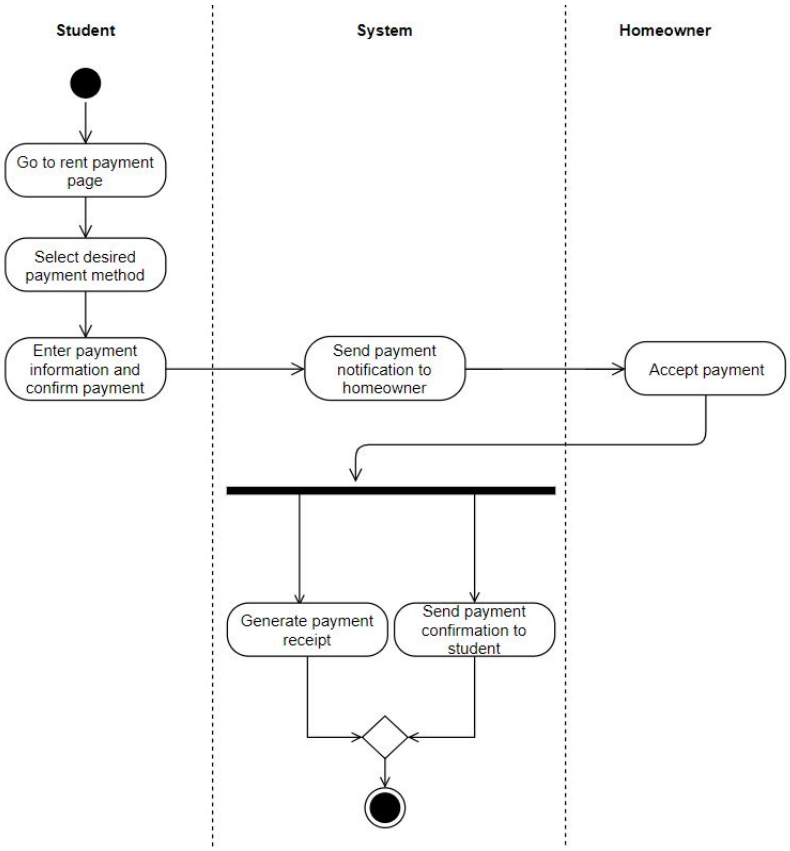



**User Story #4**
**Description:** As a student I want to to find an apartment to rent.
**Rationale:** I need a place to live.
**Source:** Discussion about BUC 4
**Fit Criterion:** The results are returned by the system database for the given filters. They are then shown to the user in a readable format.
**Dependencies:** None
**Supporting Materials:** Search filters

**Business Use Case Name:** Rent Payment
**BUC #:** 5
**Trigger:** Beginning of a new month
**Pre-condition:** Student is currently renting from a homeowner
**Active Stakeholders:** Student, homeowner

1. Open payment page
2. Select desired payment method
3. Enter payment information and confirm payment
4. Homeowner receives a notice that payment has been sent
5. Homeowner accepts the payment
6. Send confirmation to the student that their payment has been received
7. Generate a receipt and send it to the student

**Outcome:** Required rent payment for that month is now paid

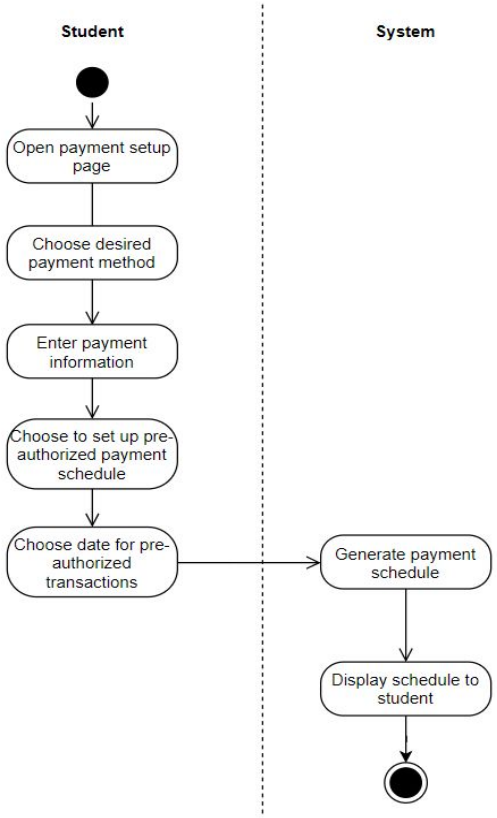



**User Story #5**
**Description:** As a student and a student I need to be able to pay my rent.
**Rationale:** I need to pay rent to avoid having a dispute filed against me.
**Source:** Discussion about BUC 5
**Fit Criterion:** The payment system tells us that payment has successfully been received. The system tells the user that their payment has been accepted.
**Dependencies:** Student needs to be actively renting a listing, payment information.
**Supporting Materials:** None

**Business Use Case Name:** Pre-authorized payment setup
**BUC #:** 6
**Trigger:** Student decides to set up this feature offered by the system
**Pre-condition:** Student is currently renting from a homeowner
**Active Stakeholders:** Student (trigger), Associated Bank

1. Open payment page
2. Select desired payment method
3. Enter payment information
4. Choose to set up a pre-authorized payment schedule
5. Choose date for pre-authorized payments to go through
6. Generate payment schedule and display it to the student

**Outcome:** Pre-Authorized Payment Set up complete. Students rent is now automatically paid based off payment schedule.

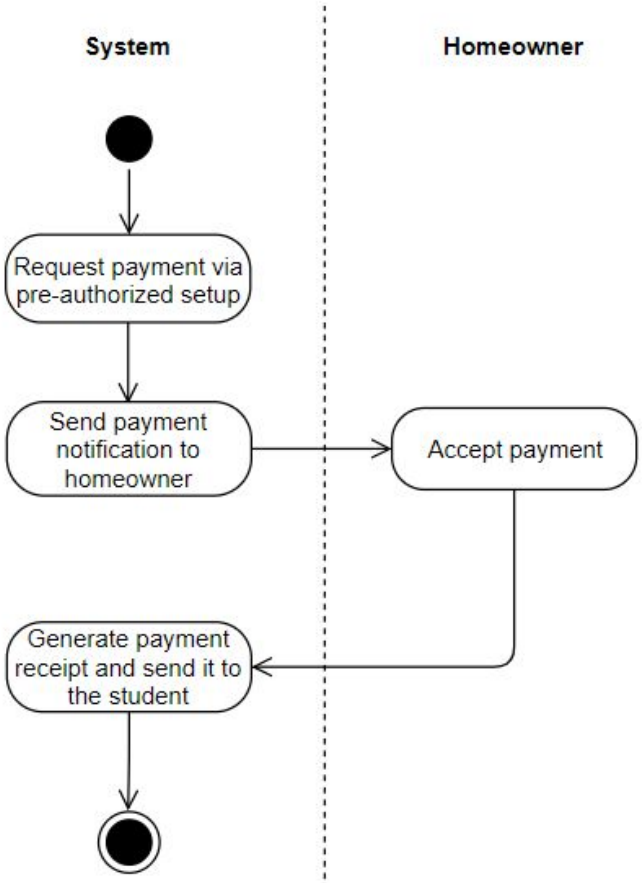



**User Story #6**
**Description:** As a student I want to be able to set up a pre-authorized payment to automatically pay rent each month.
**Rationale:** I want the ease-of-use associated with not having to log in to the system each month to manually pay rent.
**Source:** Discussion about BUC 6
**Fit Criterion:** The pre-authorized payment is approved by the user's bank.
**Dependencies:** Bank account.
**Supporting Materials:** None

**Business Use Case Name:** Automated Rent Payment
**BUC #:** 7
**Trigger:** Time triggered, (When the beginning of a new month starts)
**Pre-condition:** Must have pre - authorized payments set up
**Active Stakeholders:** Student, Associated Bank

1. Students bank information is saved to system
2. Bank automatically submits Students payment for rent when scheduled time is reached
3. Payment receipt confirmation sent via email to both Student and homeowner

**Outcome:** Students rent is now paid for using pre-authorized payment

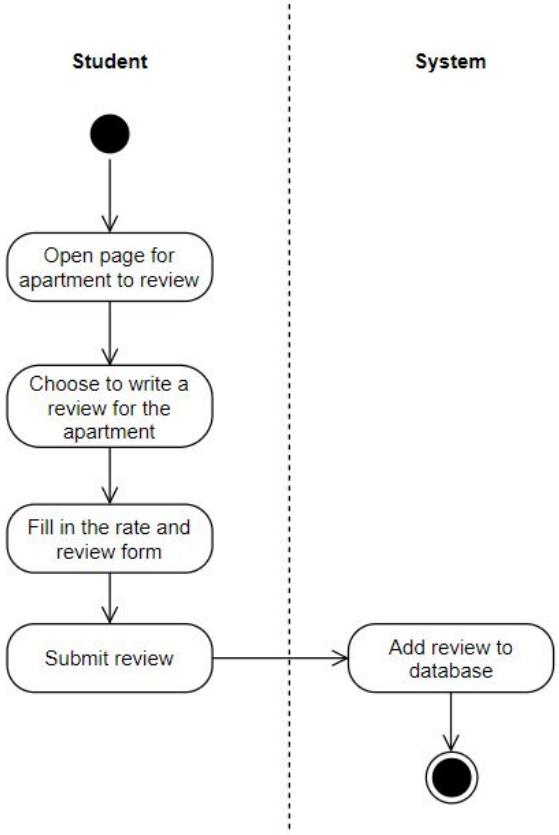



**User Story #7**
**Description:** An automated pre-authorized payment is processed by our system.
**Source:** Discussion about BUC 7
**Fit Criterion:** The pre-authorized payment is approved by the users bank.
**Dependencies:** BUC 6, having a pre-authorized payment set-up
**Supporting Materials:** None

**Business Use Case Name:** Student completes apartment review
**BUC #:** 8
**Trigger:** Student chooses to review apartment
**Pre-condition:** Student has lived in apartment being reviewed
**Active Stakeholders:** Student (trigger), homeowner

1. Student chooses apartment to rate and review
2. Rating is selected by student
3. Student includes review of apartment to help explain chosen rating
4. Rating and review are added to database

**Outcome:** Student rating and review of apartment is now posted on the apartment's page.

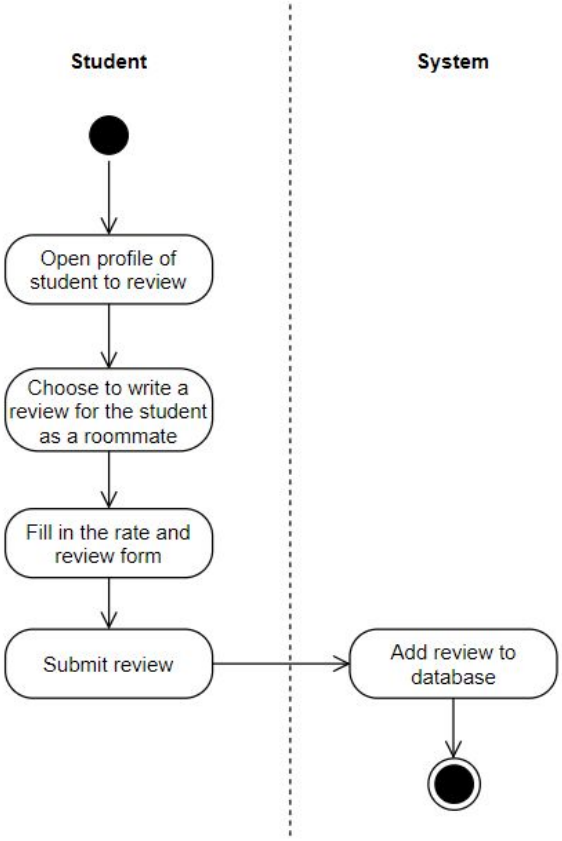



**User Story #8**
**Description:** As a student I would like to be able to rate and review my apartment.
**Rationale:** Giving feedback on apartment
**Source:** Discussion about BUC 8
**Fit Criterion:** Review is successfully added to listing page.
**Dependencies:** Review details.
**Supporting Materials:** None

**Business Use Case Name:** Student completes roommate review
**BUC #:** 9
**Trigger:** Student chooses to review roommate
**Pre-condition:** Student has to have lived with roommate before
**Active Stakeholders:** Student (trigger), roommate

1. Student chooses roommate to rate and review
2. Student rates and writes review for roommate
3. Review is saved to database and posted to roommate profile

**Outcome:** Student rating and review of roommate is posted on roommates profile and added to the systems database.

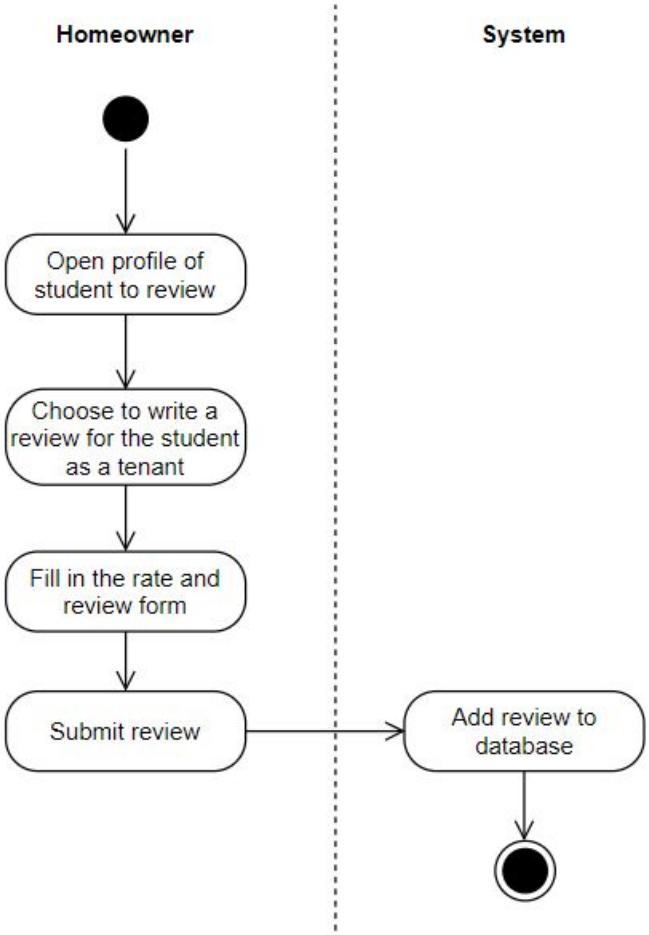



**User Story #9**
**Description:** As a student I would like to be able to rate and review my roommates.
**Rationale:** Giving feedback
**Source:** Discussion about BUC 9
**Fit Criterion:** Review is successfully added to roommate profile.
**Dependencies:** Review description.
**Supporting Materials:** None

**Business Use Case Name:** Homeowner writes student review
**BUC #:** 10
**Trigger:** Homeowners chooses to review student
**Pre-condition:** Individual being reviewed must be a student.
**Active Stakeholders:** Homeowner(trigger), student

1. Homeowner selects student to review
2. Homeowner writes and submits review
3. Review is added to the database

**Outcome:** The Homeowners review is posted to the students profile.

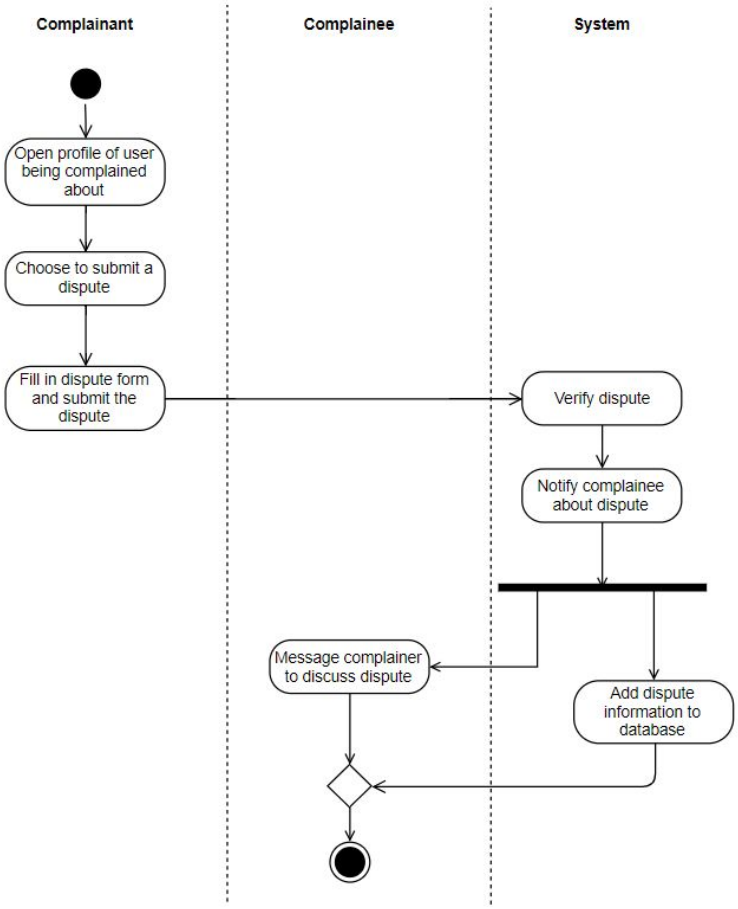



**User Story #10**
**Description:** As a homeowner I would like to be able to write reviews about students that are renting from me.
**Rationale:** Giving feedback on students that are renting
**Source:** Discussion about BUC 10
**Fit Criterion:** Review is successfully added to student profile.
**Dependencies:** Student is renting from landlord, review description.
**Supporting Materials:** None

**Business Use Case Name:** Complaint or Dispute submitted
**BUC #:** 11
**Trigger:** Student or homeowner is bothered by something resulting in a complaint being filed
**Pre-condition:** To file a complaint or dispute, you must be either a student or homeowner of an apartment.
**Active Stakeholders:** Student or homeowner(trigger),

1. Student or homeowner opens page of individual being complained about
2. User chooses to make a dispute against the chosen user
3. Dispute is written and submitted
4. Dispute is verified by someone working for the company
5. Both parties involved in the dispute contact each other through the messaging system

**Outcome:** A complaint has be filed and will be stored in the systems database.

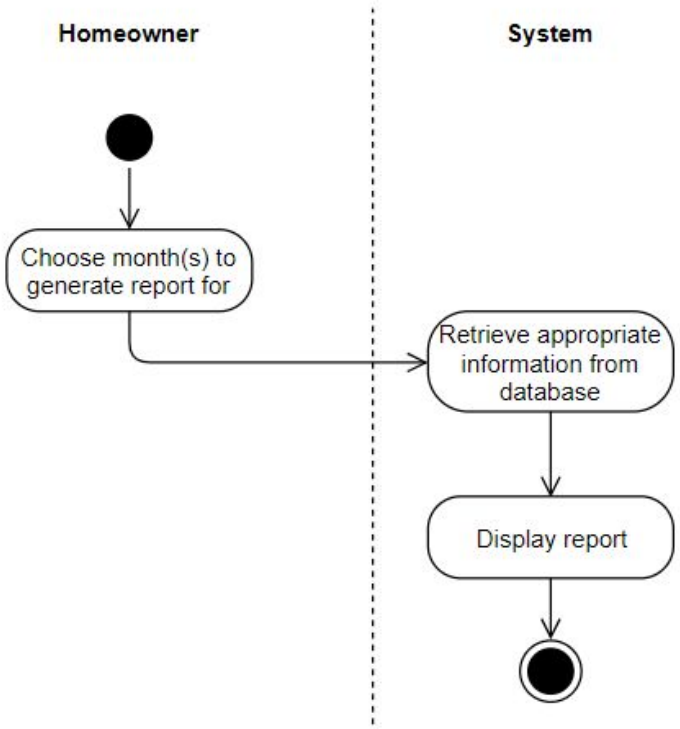



**User Story #11**
**Description:** As a student or homeowner I want to submit a dispute.
**Rationale:** I have a dispute to file upon a homeowner or student.
**Source:** Discussion about BUC 11
**Fit Criterion:** Dispute is successfully filed in our system.
**Dependencies:** Dispute information.
**Supporting Materials:** None

**Business Use Case Name:** Monthly Report
**BUC #:** 12
**Trigger:** Homeowner requests a report
**Pre-condition:** Homeowner chooses to receive monthly report before the month ends.
**Active Stakeholders:** Homeowner, Developers

1. Homeowner chooses the month(s) they want a report for
2. Retrieve information from database
3. Generate report using retrieved information
4. Display report

**Outcome:** A detailed report is sent displayed to the homeowner showing the details

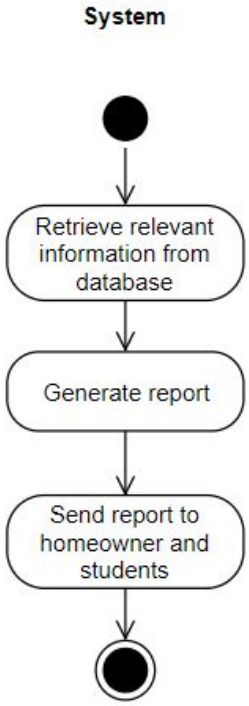



**User Story #12**
**Description:** As a student or homeowner I would like to receive a report on monthly payments on a specific date each month.
**Rationale:** I want to keep a history of payments.
**Source:** Discussion about BUC 12
**Fit Criterion:** Report is generated and sent successfully.
**Dependencies:** User must be a student renting a house or a landlord renting out their house.
**Supporting Materials:** None

**Business Use Case Name:** Annual Report
**BUC #:** 13
**Trigger:** Occurs at the end of the year
**Pre-condition:** Must be a Homeowner or student to receive
**Active Stakeholders:** Developers, Homeowner, Student

1. Information is fetched from the database at the end of the term
2. End of term report is automatically generated
3. Report sent to Homeowner and Student

**Outcome:** Report is sent to Homeowner and Students at the end of the term.

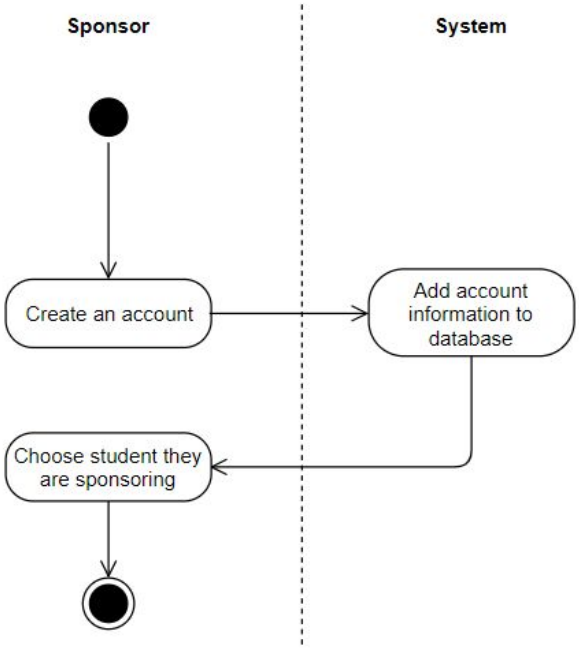



**User Story #13**
**Description:** As a homeowner I would like to receive a annual report on all payments made by students.
**Source:** Discussion about BUC 13
**Fit Criterion:** Report is generated and sent successfully.
**Dependencies:** User must be a landlord renting out their house to students.
**Supporting Materials:** None

**Business Use Case Name:** Adding Sponsor to student account
**BUC #:** 14
**Trigger:** Student has sponsor to rent for them
**Pre-condition:** Students must be living in an apartment
**Active Stakeholders:** Student (trigger), Sponsor

1. A new account is created for the sponsor
2. Sponsor information is collected
3. Sponsor indicates which student they are sponsoring.

**Outcome:** Sponsor is now added to the students account and sponsor pays rent on students behalf.

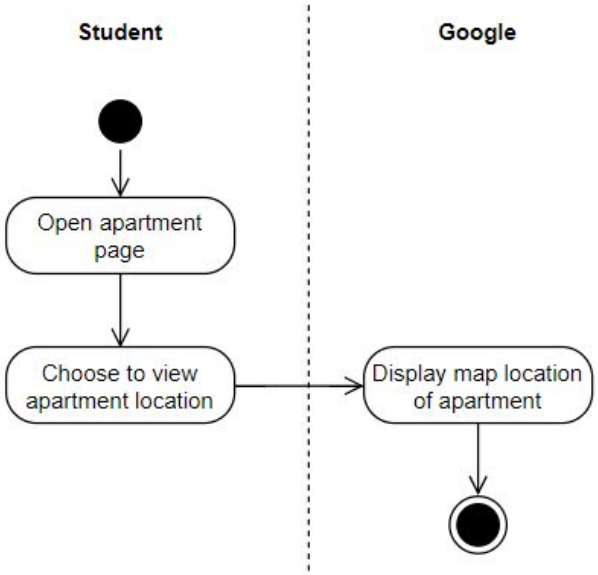



**User Story #14**
**Description:** As a student I would like to add a sponsor to my account to help me pay my rent.
**Rationale:** I cannot afford rent on my own.
**Source:** Discussion about BUC 14
**Fit Criterion:** Sponsor is linked to student account as a payer.
**Dependencies:** None
**Supporting Materials:** None

**Business Use Case Name:** Student views rental location on Google Maps
**BUC #:** 15
**Trigger:** Student activates google maps location feature
**Pre-condition:** Student must have an account on the system
**Active Stakeholders:** Student (trigger),

1. Student opens the page of a rental listing
2. Student selects option to view the location in Google Maps
3. Google Maps is opened and location is displayed

**Outcome:** Location of rental listing is open in Google Maps for student to look at.

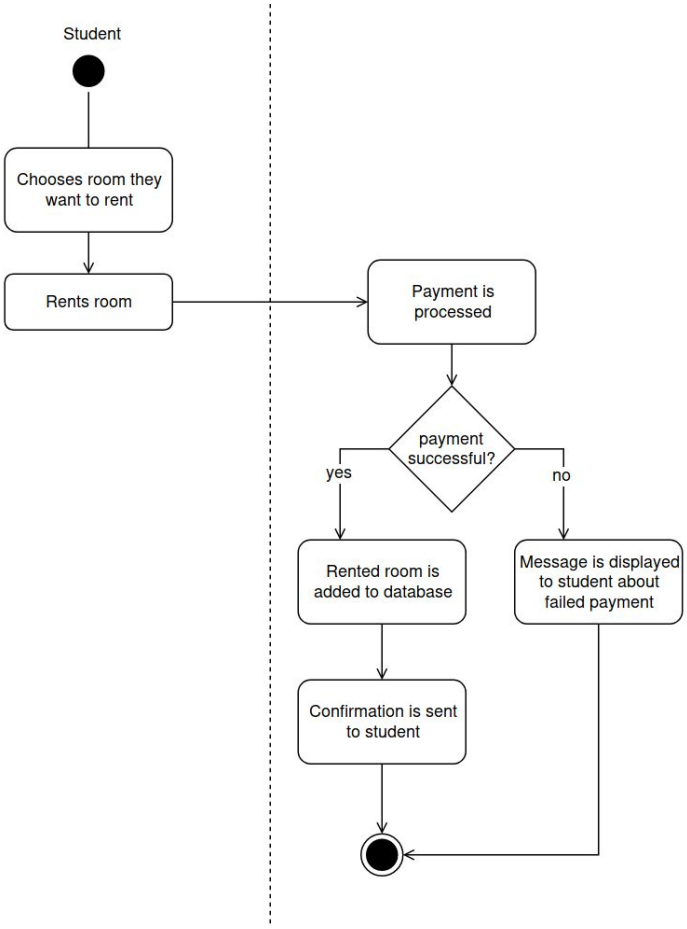



**User Story #15**
**Description:** As a student I would like to be able to see the location of the apartment or house that I looking to rent.
**Rationale:** I want to make sure that the house is in a location that I like.
**Source:** Discussion about BUC 15
**Fit Criterion:** Map is displayed to user with the correct location of said apartment or house.
**Dependencies:** None
**Supporting Materials:** Apartment/house listing

**Business Use Case:** Student rents room
**BUC #:** 16
**Trigger:** Student decides to rent room from homeowner
**Pre-condition:** Student must have an account on the system
**Active Stakeholders:** Student (trigger), Homeowner

1. Student searches for apartment to rent
2. Student decides which apartment they would like to rent
3. Decision is made between student and homeowner to let student rent room
4. Payment is made

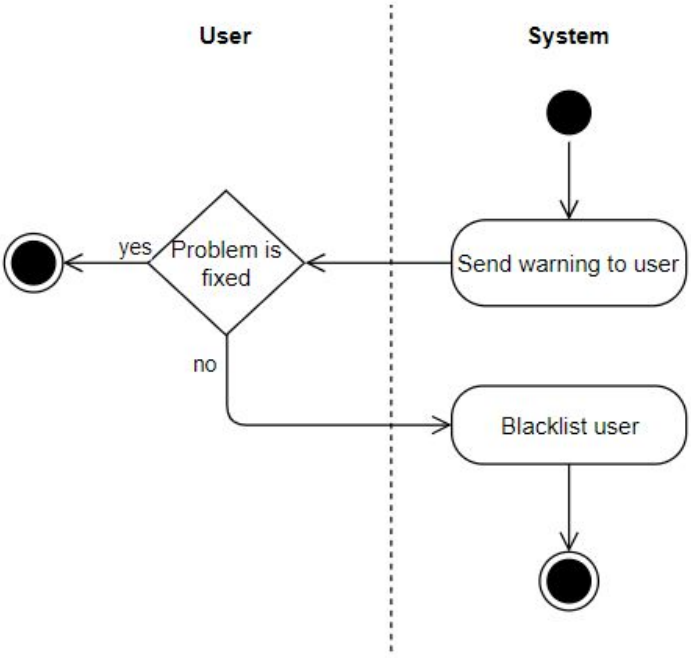
**Outcome:** Student is now renting room from Homeowner




**User Story #16**
**Description:** As a student I would like to be able to rent a room in an apartment.
**Rationale:** I like the apartment and would like to live there.
**Source:** Discussion about BUC 16
**Fit Criterion:** Room is rented and confirmation has been successfully sent to student.
**Dependencies:** Payment information
**Supporting Materials:** Apartment/house listing

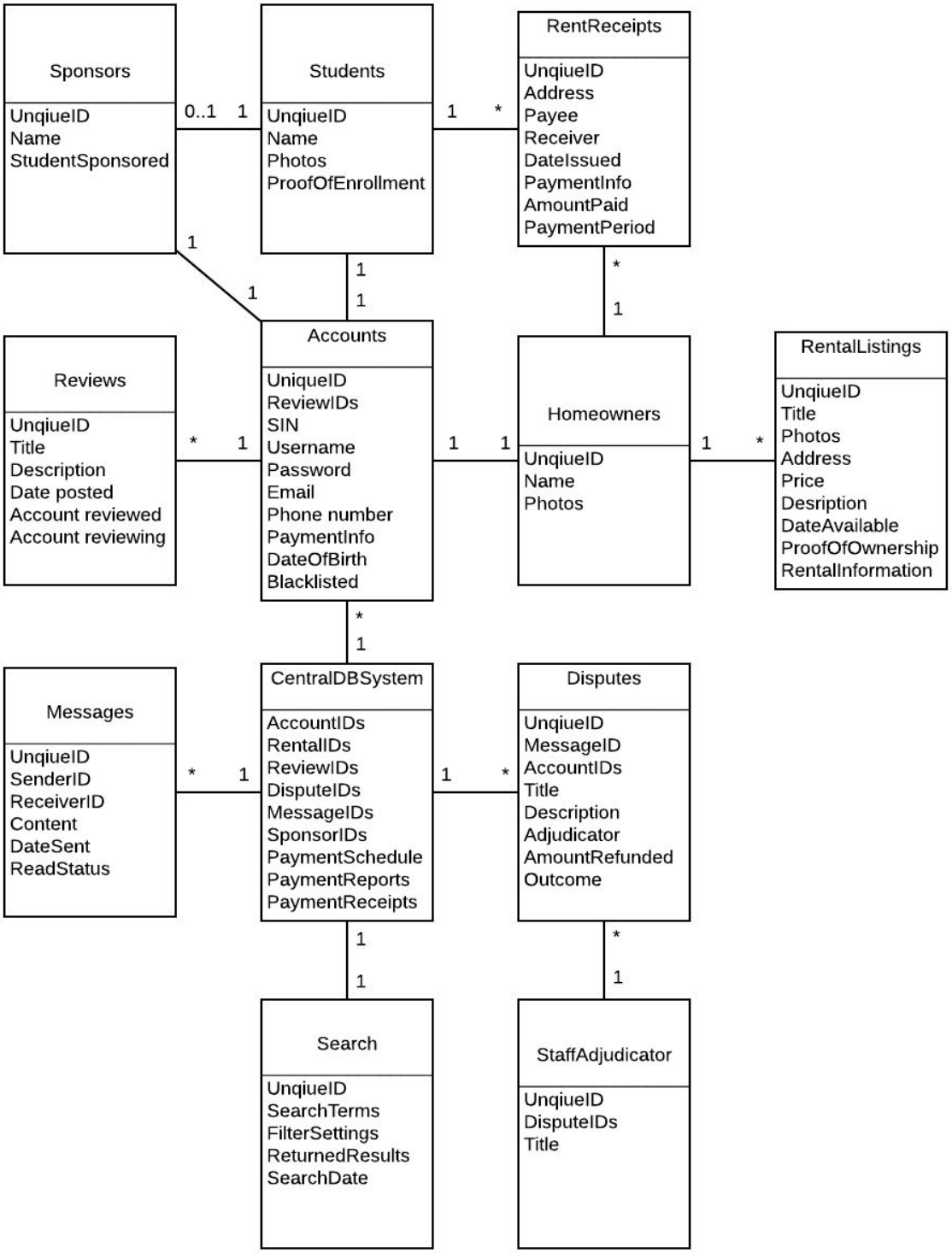
**Business Use Case:** Blacklist user
**BUC #:** 17
**Trigger:** Multiple disputes have been filed against the user
**Pre-condition:** User must have an account
**Active Stakeholders:** User being blacklisted

1. User receives a warning that they will be blacklisted
2. User is given a period of time to fix the problem
3. Blacklist user if problem has not been fixed
   A3.1 If problem has been fixed, notify user that they will not be blacklisted




**User Story #17**
**Description:** As a homeowner I have been blacklisted from creating listings or as a student I have been blacklisted from renting from homeowners.
**Rationale:** Multiple disputes have been filed against me and I have failed to respond accordingly.
**Source:** Discussion about BUC 17
**Fit Criterion:** User cannot use the system any longer.
**Dependencies:** Multiple disputes filed against user.
**Supporting Materials:** None

# Business Data Model



## Data Dictionary

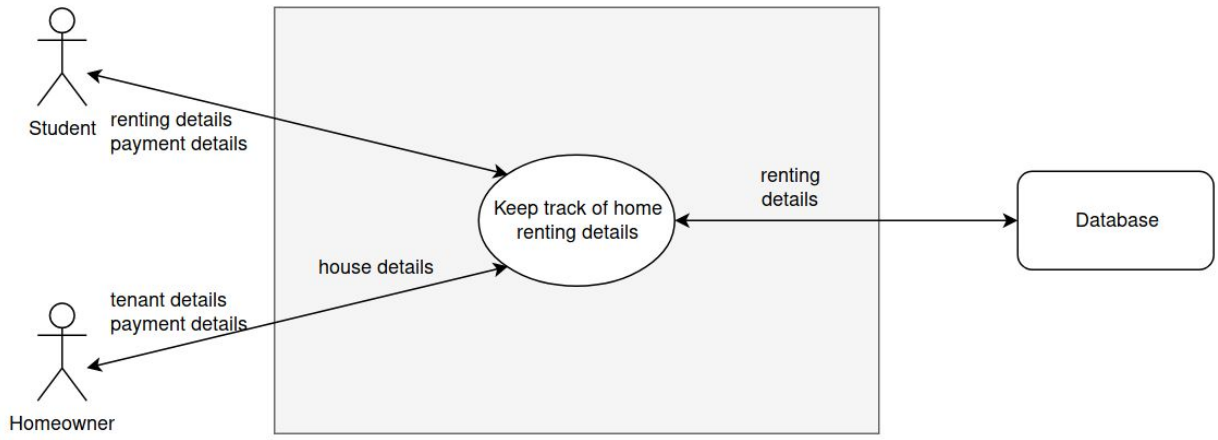
| Table | Attributes |
|---|---|
| Sponsors<br>- Stores information about users who are sponsors for students | UniqueID : INT<br>- ID<br>Name : STRING<br>- Name of user<br>Student Sponsor : INT<br>- ID of student being sponsored |
| Students<br>- Stores information about users who are students/renters | UniqueID : INT<br>- ID<br>Name : STRING<br>- Name of user<br>Photos : BLOB<br>- Photos to be displayed on profile<br>ProofOfEnrollment : FILE<br>- Ensures that students are in-fact students and elligble to use service |
| RentReceipts<br>- Stores information regarding receipts to be sent to landlords and tenants at pre-designated dates | UniqueID : INT<br>- ID<br>Address : STRING<br>- Address of apartment<br>Payee : INT<br>- Who paid amount<br>Receiver : INT<br>- Who received amount<br>DateIssued : DATETIME<br>- Date the receipt was sent<br>PaymentInfo : STRING<br>- Any extra comments regarding the payment<br>AmountPaid : FLOAT<br>- Amount paid<br>PaymentPeriod : DATETIME<br>- Which period the receipt is for |
| Reviews<br>- Stores reviews for user profiles to be displayed on their profile | UniqueID : INT<br>- ID<br>Title : STRING<br>- Review Header<br>Description : STRING<br>- Body of review<br>DatePosted : DATETIME<br>- Date that review was posted<br>AccountReviewed : INT<br>- The AccountID being reviewed<br>AccountReviewing : INT<br>- The AccountID reviewing |
| Accounts<br>- Used to store account information to access login, and adjust some settings of the app | UniqueID : INT<br>- ID<br>ReviewID : INT<br>- Reviews associated<br>SIN : INT<br>- Social Insurance Number<br>Username : VARCHAR[25]<br>- Unique username |

| | |
|---|---|
| | Password : VARCHAR[25]<br>- Encrypted password<br>Email : STRING<br>- Main point of communication<br>PhoneNumber : INT<br>- Secondary point of communication<br>PaymentInfo : JSON<br>- Stores information for sending/receiving money<br>DateOfBirth : DATETIME<br>- Date of birth of user<br>BlackListed : INT<br>- AccountIDs that are blocked |
| Homeowner<br>- Stores unique information about landlords that will be renting out their property | UniqueID : INT<br>- ID<br>Name : STRING<br>- Name of the user<br>Photos : BLOB<br>- Photos to display on profiel |
| RentalListings<br>- Stores information about the posts themselves, accessed to pull information for webpages | UniqueID : INT<br>- ID<br>Title : STRING<br>- Title of listing<br>Photos : BLOB<br>- Photos of property<br>Address : STRING<br>- Address of property<br>Price : FLOAT<br>- Monthly rate for rental<br>Description : STRING<br>- Poster's explanation of property<br>DateAvailable : DATETIME<br>- The date that the property is available to move into<br>ProofOfOwnership : FILE<br>- Proof that the poster owns the property<br>RentalInformation : STRING<br>- Specifies information about building, ex. townhouse, apartment complex |
| Messages<br>- Tracks messages and who the corresponding users are | UniqueID : INT<br>- ID<br>SenderID : INT<br>- AccountID of sending party<br>ReceiverID : INT<br>- AccountID of receiving party<br>Content : STRING<br>- The text being sent<br>DateSent : DATETIME<br>- Date message was sent<br>ReadStatus : STRING<br>- Status of whether the receiver has viewed message |
| CentralDBSystem<br>- Keeps track of all IDs and schedules the distribution of rental receipts | AccountID : INT<br>- ID<br>RentalID : INT<br>- RentalID |

| | |
|---|---|
| | ReviewID : INT<br>- ReviewID<br>DisputeID : INT<br>- DisputeID<br>MessageID : INT<br>- Message ID<br>SponsorID : INT<br>- SponsorID<br>PaymentSchedule : DATETIME<br>- Date preset for automatic payments<br>PaymentReport : JSON<br>- Monthly logging of rent payments<br>PaymentReceipt : FILE<br>- Rent receipts generated to be sent to users |
| Disputes<br>- Keeps reference to the dispute reason, messages associated, and staff associated with a given dispute | UniqueID : INT<br>- ID<br>MessageID : INT<br>- ID of messages associated with case<br>AccountID : INT<br>- Accounts associated with case<br>Title : STRING<br>- Title of case<br>Description : STRING<br>- Description of case<br>Adjudicator : INT<br>- Staff who is resolving case<br>AmountRefunded : FLOAT<br>- Requested amount being refunded<br>Outcome : FLOAT<br>- Resulting amount being refunded |
| Search<br>- Stores previous search results to gauge interest and optimize search results | UniqueID : INT<br>- ID<br>SearchTerms : STRING<br>- Terms used in search<br>FilerSettings : JSON<br>- Filter settings<br>ReturnedResults : BLOB<br>- Post IDs of returned results<br>SearchDate : DATETIME<br>- DateTime of search |
| StaffAdjudicator<br>- Keeps track of which dispute cases staff are assigned to | UniqueID : INT<br>- ID<br>DisputeID : INT<br>- Disputes associated with staff<br>Title : STRING<br>- Title of staff |

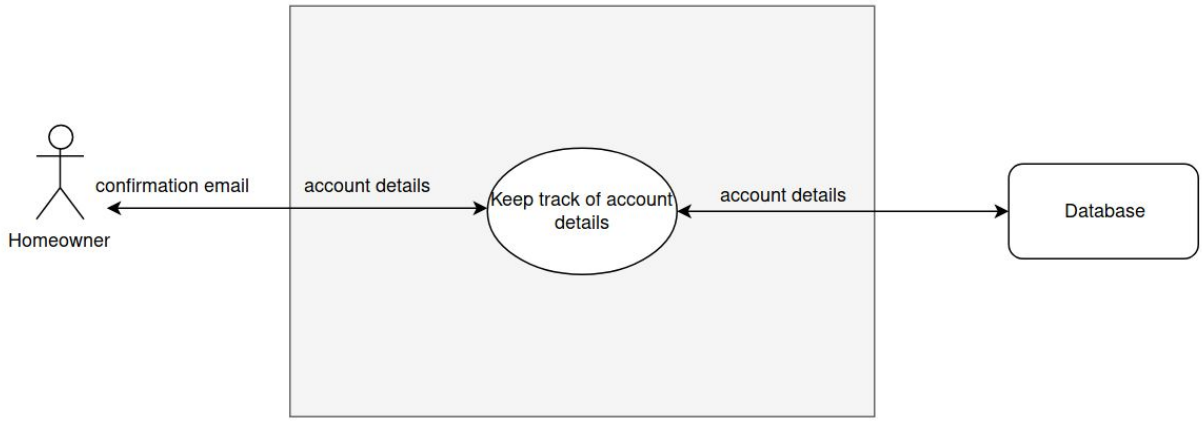
# Scope of the product

Product Use Case Diagram

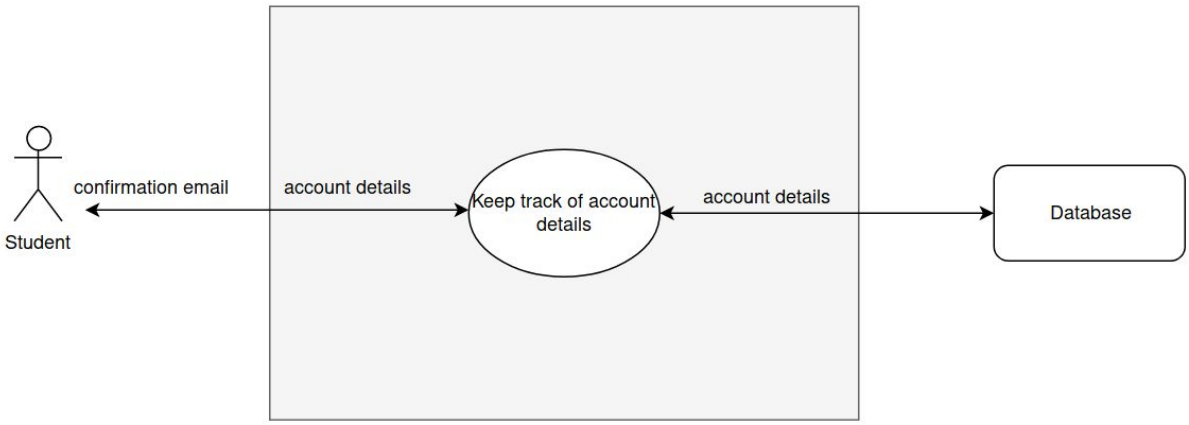



Individual Product Use Cases

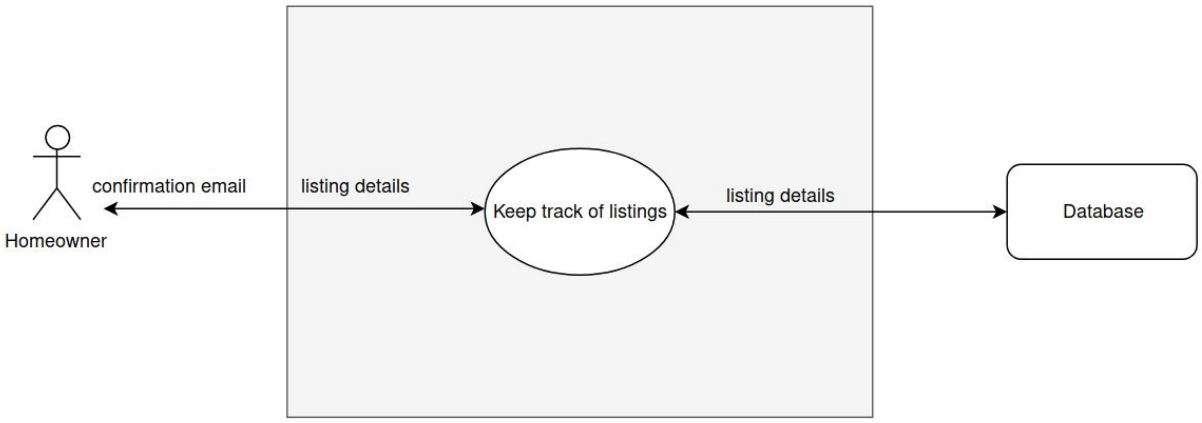
1. Product use case name: Create a Homeowner account
   Trigger: Homeowner creates homeowner account
   Preconditions: The user must be a homeowner
   Interested stakeholders: Homeowner, Student



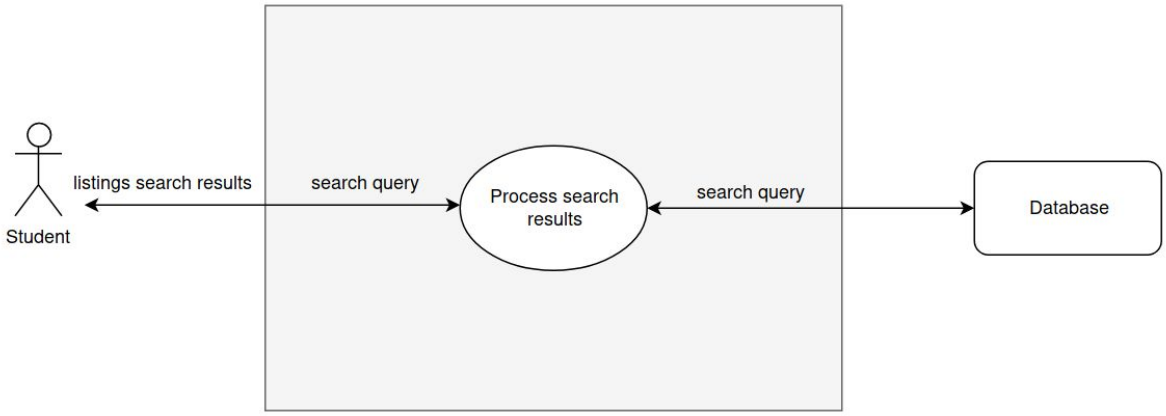
2. Product use case name: Create a student account
   Trigger: Student creates student account
   Preconditions: The user must be a student
   Interested stakeholders: Student, Homeowner, Sponsor



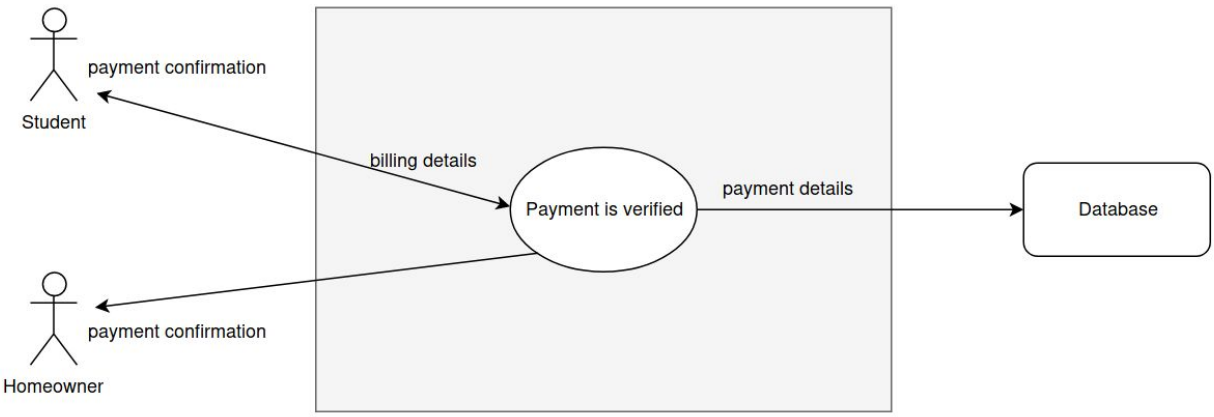

3. Product use case name: Create a new rental listing
   Trigger: Homeowner opens page to create a rental listing
   Preconditions: Must have a homeowner account
   Interested stakeholders: Homeowner, Student



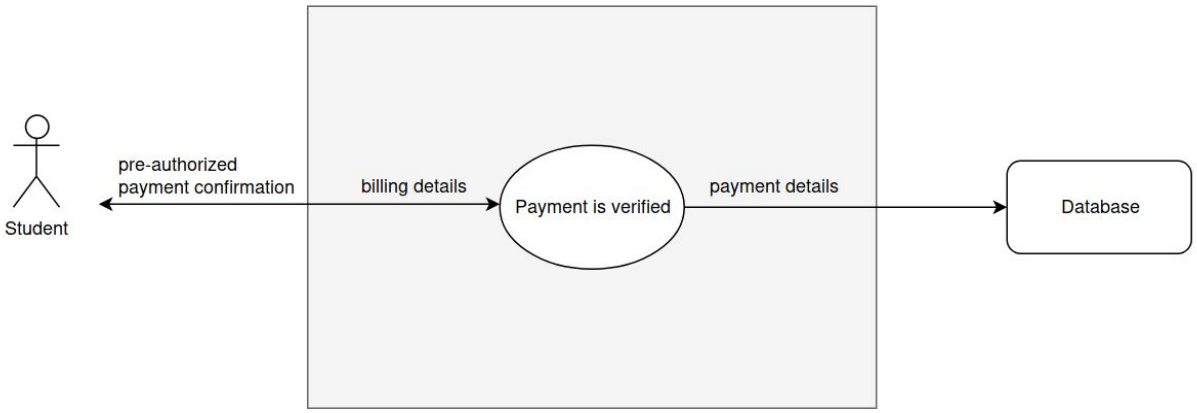
4. Product use case name: Display search results
   Trigger: User creates search query and searches
   Preconditions: None
   Interested stakeholders: Student, Homeowner



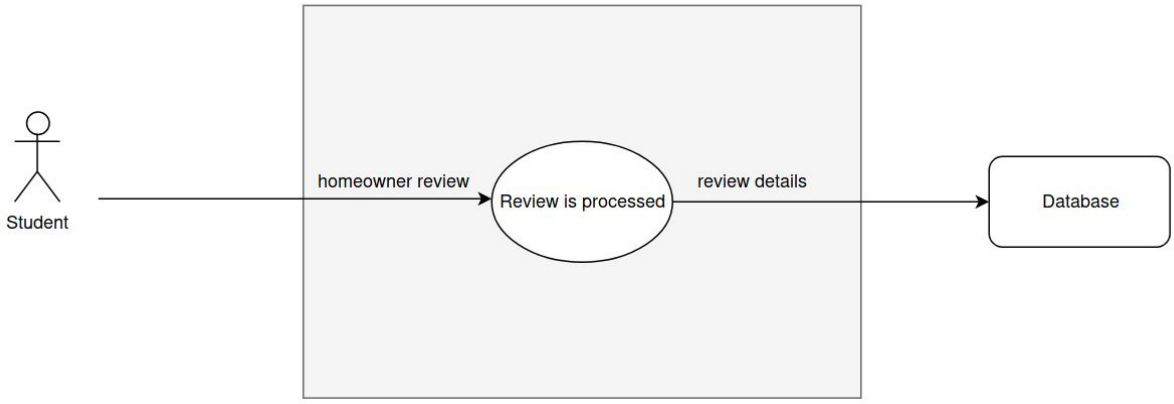

5. Product use case name: Rent Payment
   Trigger: Reach the beginning of a new month
   Preconditions: Must be a tenant of a house
   Interested stakeholders: Student, Homeowner, Sponsor



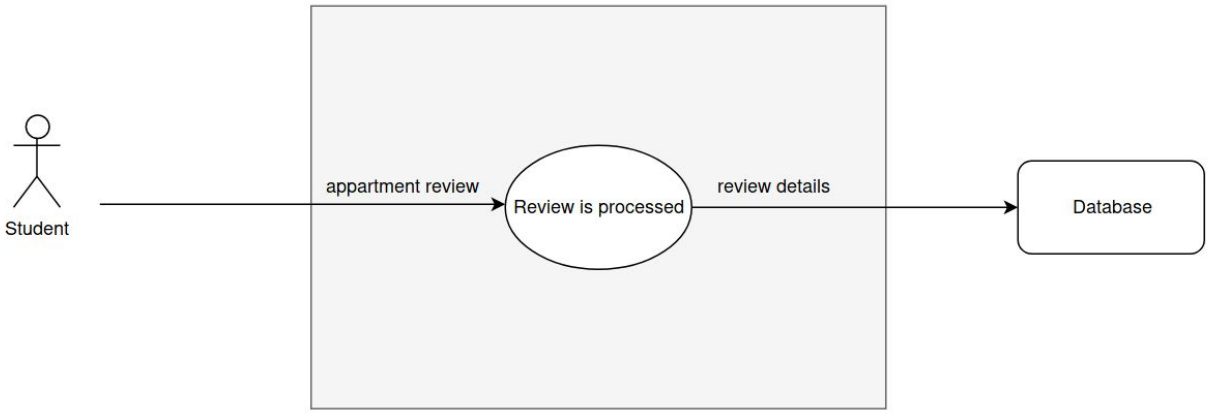
6. Product use case name: Pre-authorized payment setup
   Trigger: Student opens page to configure pre-authorized payments
   Preconditions: Must have a student account
   Interested stakeholders: Student, Homeowner, Sponsor



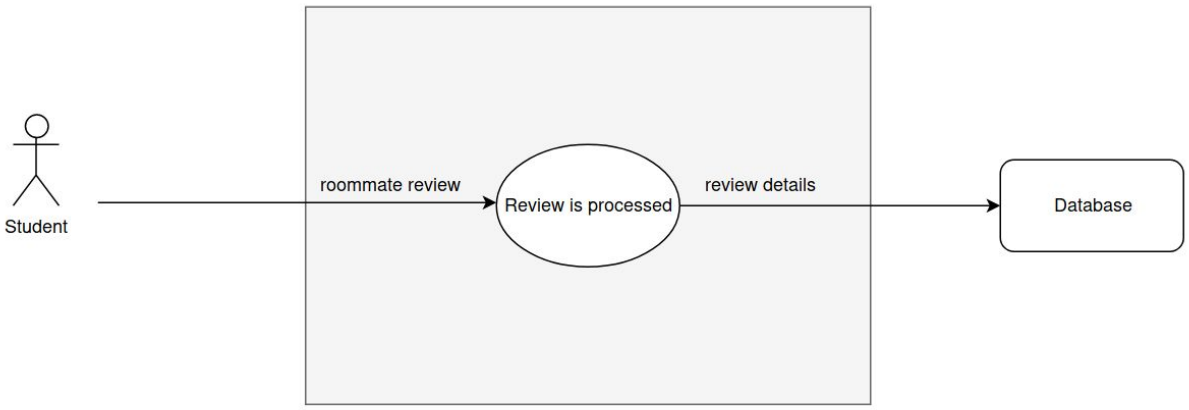

7. Product use case name: Student completes homeowner review
   Trigger: Student submits a homeowner review
   Preconditions: Student must have rented from the homeowner, and must have a student account
   Interested stakeholders: Student, Homeowner



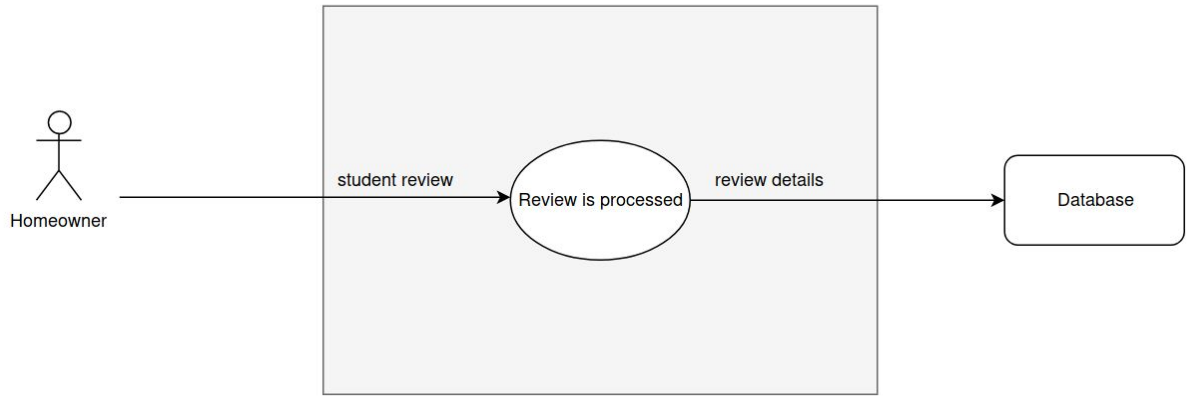
8. Product use case name: Student completes apartment review
   Trigger: Student submits an apartment review
   Preconditions: Student account, has rented at apartment
   Interested stakeholders: Student, Homeowner



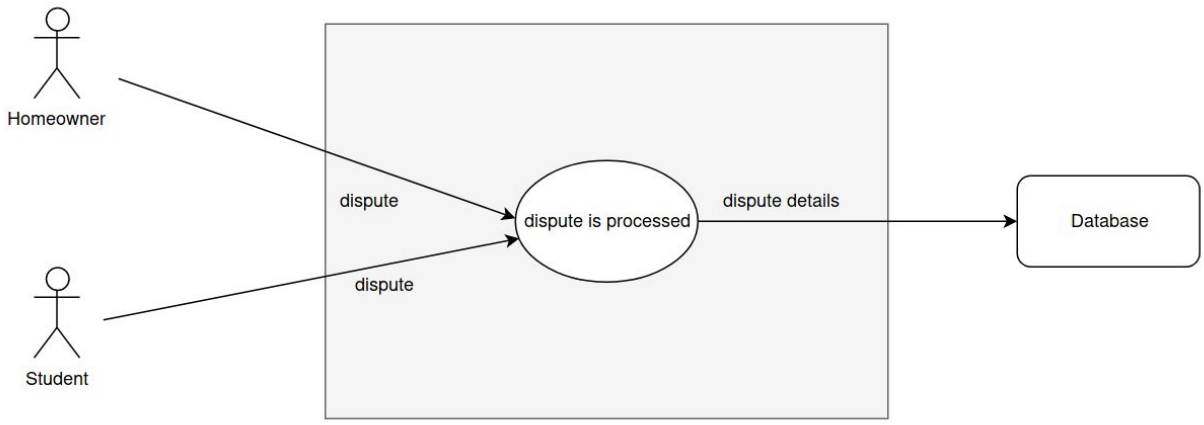

9. Product use case name: Student completes roommate review
   Trigger: Student decides to submit roommate review
   Preconditions: Must be a student renting a house with roommates·
   Interested stakeholders: Student



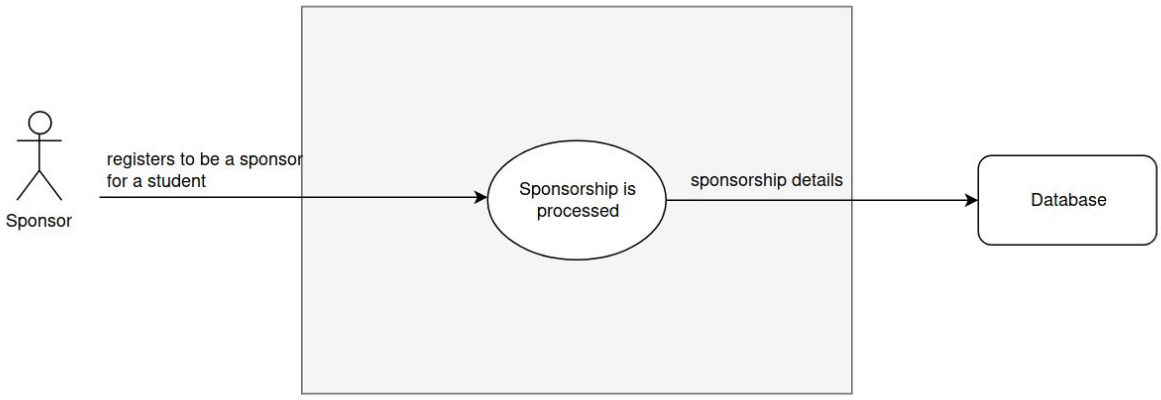
10. Product use case name: Homeowner writes student review
    Trigger: Homeowner submits student review
    Preconditions: Homeowner account, student must be renting from the homeowner
    Interested stakeholders: Homeowner, Student



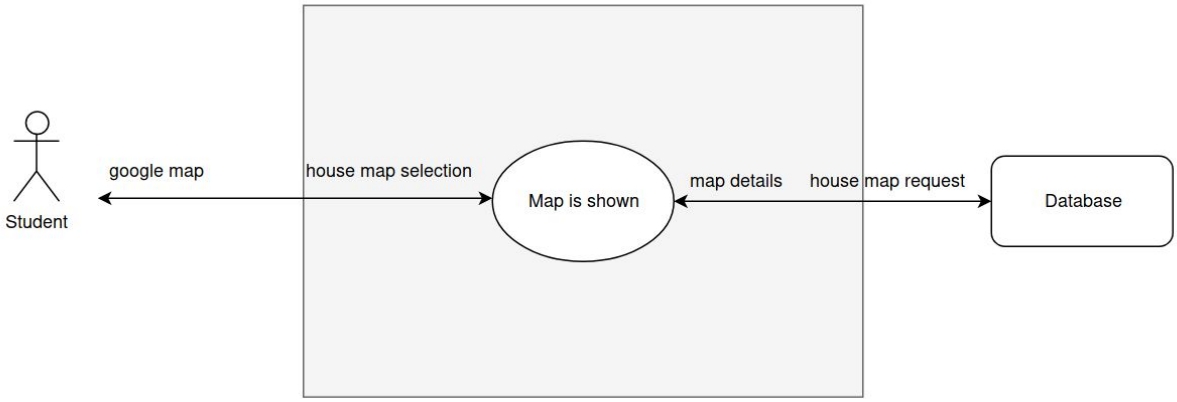

11. Product use case name: Complaint or Dispute submitted
    Trigger: A complaint is submitted
    Preconditions: Must have an account
    Interested stakeholders: Homeowner, Student, Sponsor



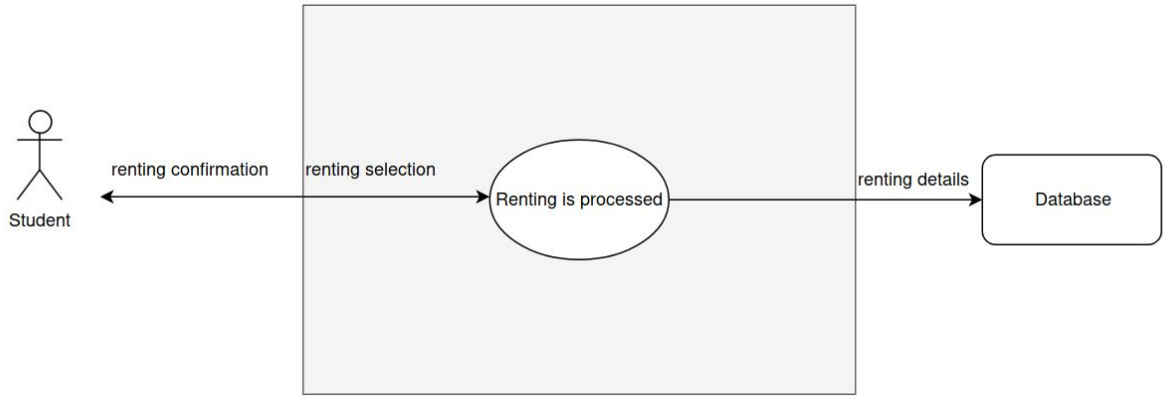
12. Product use case name: Adding Sponsor to student account
    Trigger: Sponsor chooses to add themselves to a specific student account
    Preconditions: Sponsor account, student to add themselves to
    Interested stakeholders: Sponsor, Student




13. Product use case name: Student views rental location on Google Maps
    Trigger: Student clicks to open map
    Preconditions: Rental listing
    Interested stakeholders: Homeowner, Student, Sponsor



14. Product use case name: Student rents room
    Trigger: Student chooses to rent desired room
    Preconditions: Must be a registered student
    Interested stakeholders: Student, Homeowner, Sponsor



# Non-functional Requirements

Look and feel requirements

- Customizable colour schemes
  Description: As a user I want to customize my experience with the app
  Fit Criterion: Users should be able to go to a settings menu and adjust colour
- Accessible to all students
  Description: As a user with disabilities, I want to still be able to use the app
  Fit Criterion: App should cater to users with special needs

Usability and humanity requirements

- Easy to use
  Description: As a user I want to easily be able to do activities quickly
  Fit Criterion: Have an interface that is intuitive and has a max of 4 clicks to get anywhere
    1. Large buttons and menus that are easy to understand
    2. Compatible with text to speech readers
    3. Option for dyslexic friendly font
- Easy to learn
  Description: As a user I want to be able to use the system quickly after discovering it
  Fit Criterion: Intuitive functionality similar to other common marketplace websites
    1. New users should be able to understand how to use the software within 5 mins
    2. Intuitive interface that follows modern design trends for website
- Available in both English and French
  Description: As a user who speaks french, I want to be able to use the app
  Fit Criterion: Text should be accurately translated to french

Performance requirements

- Provide search results fast
  Description: As a user I want to be able to view search results quickly.
  Fit Criterion: Optimized database schema and queries
    1. Data from the Search table should be analyzed to optimize searches and prioritize results that appear multiple times

Operational and environmental requirements

- Should be fault tolerant, should work under heavy loads
  Description: As a user I want to use a system that is reliable and won't lag
  Fit Criterion: The system can handle 5000 users simultaneously
    1. Should be able to handle at least 5000 users simultaneously
    2. Should use a standard SQL database engine
- Should run on all browsers

Description: As a user I want to use the system on any device accessible to me
Fit Criterion: System is accessible on any device

1. All major browsers should be able to use the service without issues

- Should run on both desktop and mobile app
  Description: As a user I want to be able to use this system wherever I am
  Fit Criterion: Web and app will be designed similarly and be available on all platforms

Maintainability and support requirements

- Backups
  Description: As a company we want to ensure our data is stored with lots of redundancy
  Fit Criterion: Follow a backup cycle that ensures data is stored in multiple locations
  1. Nightly backups of listings, and account changes to a raid style database
  2. Bi-weekly backups of entire system to an on-site logging database
  3. Monthly backups of entire system to an offsite logging database
- Maintenance time
  Description: As a company we want to ensure that our product does not experience any downtime during peak traffic hours
  Fit Criterion: Follow a monthly version update system, and allow for daily patches to prevent downtime
  1. Maintenance on site can happen on a daily basis but only on non-peak hours
  2. Monthly maintenance for version updates

Security requirements

- Should ensure the security and integrity of data
  Description: As a company we want to give our users the comfort that their data is safe
  Fit Criterion: Data should undergo encryption while being stored in our database system
  1. Data should be kept private from the users aside from their own information
  2. All information kept on databases should be encrypted to prevent on site attacks
  3. Data on remote servers should also be encrypted as well
- Should ensure the privacy of users' personal data
  Description: As a user I want to keep my personal information secure
  Fit Criterion: Provide users with the option to choose what they display on their profiles
  1. Users should be able to choose what personal information they share
- Should ensure that only authorized personnel employed by the client can access the users data in case of dispute
  Description: As a user I want personal information exchanged in a dispute to be secure
  Fit Criterion: Give users the option to link other accounts to theirs as trusted accounts

Cultural requirements

- Compliant with company branding standards
  Description: As a company we want the app to seamlessly blend with our other products
  Fit Criterion: Ensure interface is similar to other applications in both fit and feel
- Should not be offensive to any religious or ethnic group

Description: As a I don’t want a safe space to view apartment listings
Fit Criterion: App avoids any mention of religion, or ethnic groups

Legal requirements

- Should allow auditors to audit the accounting data
  Description: As a tax auditor I need access to certain information to file taxes
  Fit Criterion: Authorize specific companies to have access to limited data
- Should comply to all Canadian privacy laws
  Description: As a business we are required to keep an amount of user data protected
  Fit Criterion: Follow development standards and ensure we meet Canadian regulations

# Project Issues

Open issues

1. Developing an internal map app is large and expensive

Off-the-shelf solutions

1. Integrate google maps in place of creating our own

Risks

| **Risk Type** | **Possible Risks** | **Probability** | **Effect** |
|---|---|---|---|
| Technology | 1) Inability to coordinate and process batches of transactions at preauthorized time of the month (thousands of users all processing payment at once) | 1) Medium, without stress tests against our system we are unsure of whether or not this may happen | 1) Catastrophic, will result in potentially thousands of dollars being lost to the void |
| | 2) Potential of payment processing systems going deprecated or faulting could cause a key part of the app to fail | 2) Low, we are considering massive corporations failing so this is unlikely | 2) High, the loss of a payment method would result in loss of profits |
| | 3) Flaws in dispute system could result in false claims being passed and money being transferred that should not have been transferred otherwise | 3) Medium, without proper user testing it is unsure how someone may mess with system to result in errors | 3) Low, false claims could be easily reverted manually and fixing the issue will require hiding the problem from the public and releasing bug fixes |
| | 4) Difficulty getting homeowners (adults) to become familiar and engaging with the service. | 4) Medium, adults are not as tech savvy as the younger generation | 4) High, without homeowners using the app, there is no apartments to be rented |

| | | | |
|---|---|---|---|
| People | 1) A very wide range of developers is required which may mean that development will start slowly, or we will be unable to develop app | 1) Medium, because of how many developers are required it will take a significant amount of time to find appropriate staff | 1) Medium, will slow the development process, but likely won't result in no product being produced |
| | 2) Requirements for extra staff for dispute system as well as marketing and sales during operational phase may result in a need to increase budget | 2) Low, the budget for this project is relatively high | 2) The inability to hire this staff will result in difficulty in user uptake and bring in clients |
| | 3) No criminal background check required for tenants. Will be judged on their profile reviews from other homeowners. | 3) Medium, although it is not required, homeowner could still ask tenants for such information | 3) Medium, because it is not required, it could be a safety concern |
| | 4) Inaccurate reviews from homeowners and tenants. | 4) Low, users of the system are encouraged to be completely honest when writing reviews | 4) High, the entire rating system could be flawed if such an event occured. |
| Organizational | 1) Shift in project management staff could result in project requirements changing | 1) Low, unlikely that a new project management office would be brought in to a new product | 1) Medium, would require restructuring of project and delay product as we adjust development |
| | 2) Peterborough governing body does not allow us to produce a system based in the Peterborough Kawartha area | 2) Low, the presence of services like Kijiji shows that this is unlikely | 2) Low, just means that we pivot and move our product to another region |
| | 3) Initial investors may drop out of product meaning | 3) Low, the product is clear | 3) High, budget will lower resulting in |

| | | | |
|---|---|---|---|
| | budget for project would drop drastically | and has a clear market | potential layoffs of developers and shortening of development period |
| Tools | 1) If our product is too similar to other competitors already on the market we risk a lawsuit<br><br>2) Requirement on Peterborough bus system to remain consistent and optimal for our system may vary | 1) Low, most filter search systems are very similar<br><br>2) Low, bus routes have been consistent for years | 1) Catastrophic, a lawsuit would result in no more funding and thus no more development funds<br><br>2) Low, this would affect user experience however not affect our usage rates very much, if product is #1 used product on market |
| Requirements | 1) The dispute system will require personnel to judge the validity of disputes and make decisions about whether to process dispute or not<br><br>2) Payment processes that we require do not authorize out system to use them to process payments | 1) Definite, this system needs to be in place in order for disputes to work<br><br>2) Low, services like this are often approved | 1) Low, however we require developing a system to deal with this<br><br>2) High, means that a key functionality of our product is not offered |
| Estimation | 1) Budget may be too low for a 1.5 year project requiring at least 6 developers plus operational employees<br><br>2) Time estimate may result in a crunch to get the product complete | 1) Low, 1.5years * 52 weeks a year * 40 hours a week * $20 developer wage is about $62,400. This gives us room for 45 employees at this rate<br><br>2) 1.5 years is a relatively short amount of time to develop a system of this calibre | 1) Medium, having to cut back on staff will result in slower development speed and a lower quality of final delivered product.<br><br>2) Medium, will result in delivery time being pushed back |

# Costs

**Input Business Use Cases = 63 Function points**

- BUC 1: Create a homeowner account
    - Classes Referenced = 3
    - Attributes = 13
    - 6 Function points
- BUC 2: Create a student account
    - Classes Referenced = 3
    - Attributes = 15
    - 6 Function points
- BUC 3: Create a new rental listing
    - Classes Referenced = 3
    - Attributes = 11
    - 6 Function points
- BUC 5: Rent Payment
    - Classes referenced = 4
    - Attributes = 9
    - Function points = 6
- BUC 8: Student completes apartment review
    - Classes referenced = 5
    - Attributes = 9
    - Function points = 6
- BUC 9: Student complete roommate review
    - Classes referenced = 4
    - Attributes = 10
    - Function points = 6
- BUC 10: Homeowner writes student review
    - Classes referenced = 5
    - Attributes = 9
    - Function points = 6
- BUC 11: Complaint or Dispute submitted
    - Classes referenced = 6
    - Attributes = 18
    - Function points = 6
- BUC 14: Adding sponsor to student account
    - Classes referenced = 5
    - Attributes = 7
    - Function points = 6
- BUC 16: Student rents a room from a homeowner
    - Classes referenced = 7
    - Attributes = 25
    - Function points = 6

- BUC 17: Blacklist user
    - Classes referenced = 2
    - Attributes = 3
    - Function points = 3

**Output Business Use Cases = 26 Function points**

- BUC 4: Display search results
    - Classes referenced = 3
    - Attributes = 7
    - Function points = 5
- BUC 12: Monthly Report
    - Classes referenced = 4
    - Attributes = 12
    - Function points = 7
- BUC 13: Annual Report
    - Classes referenced = 4
    - Attributes = 12
    - Function points = 7
- BUC 15: Student views rental location on Google maps
    - Classes referenced = 6
    - Attributes = 6
    - Function points = 7

**Time-Triggered Business Use Cases = 10 Function points**

- BUC 6: Pre-authorized payment setup
    - Classes referenced = 4
    - Attributes = 4
    - Function points = 4
- BUC 7: Automated Rent Payment
    - Classes referenced = 5
    - Attributes = 12
    - Function points = 6

**Internally Stored Data = 66 Function points**

- Accounts
    - 3 Record Elements
    - 21 Attributes
    - 10 Function points
- Rent Receipts
    - 1 Record Element
    - 8 Attributes
    - 7 Function points

- Rental Listings
    - 1 Record Element
    - 9 Attributes
    - 7 Function points
- Reviews
    - 1 Record Element
    - 10 Attributes
    - 7 Function points
- Central Database System
    - 1 Record Element
    - 9 Attributes
    - 7 Function points
- Messages
    - 1 Record Element
    - 6 Attributes
    - 7 Function points
- Search
    - 1 Record Element
    - 5 Attributes
    - 7 Function points
- Disputes
    - 1 Record Element
    - 8 Attributes
    - 7 Function points
- Staff Adjudicator
    - 1 Record Element
    - 3 Attributes
    - 7 Function points

**Externally Stored Data = 0 Function points**

**Staff month estimate = 12.367 Staff months**

Effort = (function points / 150) * function points $^{0.4}$

| | | |
|---|---|---|
| Input = | (63 / 150) * 63 $^{0.4}$ = | 2.203 staff months |
| Output = | (26 / 150) * 26 $^{0.4}$ = | 0.638 staff months |
| Time = | (10 / 150) * 10 $^{0.4}$ = | 1.675 staff months |
| Internal = | (66 / 150) * 66 $^{0.4}$ = | 2.351 staff months |
| External = | (0 / 150) * 0 $^{0.4}$ = | 0 staff months |

Total = 2.303 + 0.638 + 1.675 + 2.351 = 12.367 staff months