Home Link Specifications Report

Scenarios + Defining I/O

1. **Scenario: Registration**
    a. *Sub-scenario:* If an unregistered user opens Home Link, they should see a button that leads to two signup options: one for tenants and one for the landlord.

    b. *Sub-scenario:* Unregistered users clicked the registration/sign up button, They will be redirected to the option button that asks them if they are a landlord or tenant. Based on their selected option they will be directed to a registration page.

    c. *Sub-scenario:* A user selecting tenant in the registration form,They will be directed to Tenant registration form, and they will be asked to fill the following information:
        i. Full name.
        ii. Email.
        iii. Phone number.
        iv. Data of birth.
        v. SSN.
        vi. Previous Home address.
        vii. Rental address.
        viii. Name of the landlord.
        ix. Any roommate ?
        x. Any pets ?
        xi. Monthly rent.
        xii. Utilities that are not included in the monthly rent.
        xiii. Upload a proof of residency { lease contract, or electricity bill} .

    d. *Sub-scenario*: A user selecting landlord in the registration form, They should be directed Landlord registration form, and they should be asked to fill the following information:
        i. Full name
        ii. Email address
        iii. Phone number
        iv. Date of birth
        v. SSN
        vi. Home address
        vii. List of owned properties addresses.

    e. *Sub-scenario:* Unregistered users submitting registration form, They should receive confirmation email with a link to verify their email.

f. *Sub-scenario:* Unregistered users receiving a confirmation email, They should be able to verify their email, when they click the verification link.

g. *Sub-scenario:* If the unregistered users are tenants, their account will not be activated until the proof of residency is verified.

h. *Sub-scenario:*If the unregistered users are landlords, their account will be activated with the verification link, but Homelink should be limited to provide only a request to claim their properties. On a successful claim, they should see all the Homelink services.

i. *Sub-scenario:* If the users are verified, they should receive a temporary password through the email to login to the website.

2. Scenario: Logins

a. The login page for all users is structured:
   i. Email field
   ii. Password field

b. This information is supplied by the user and goes to Google Firebase for authentication.

c. Forgot Email? / Forgot Password will also be handled through Firebase.

3. Landlord Scenarios (post login)

a. *Sub-scenario:* Claiming a home
   i. A landlord needs to claim a property. To validate, he must provide:
      1. Property deed
      2. Proof of insurance (specifically renter coverage)
      3. Address of property
      4. Photos (optional for verification)

   ii. [In the case this application was live, these details would be sent for review by a real person.] The property address is automatically checked by an address validator API (failing this step should flag the application for reviewer). Reviewer checks the legitimacy of the deed/insurance to cover renters, then confirms or rejects the application. On confirmation, the property data is registered in the database under the landlord's name. Any photos uploaded should be detected for the next step:

   iii. After the property is confirmed to the landlord, he is provided an option to upload property details including:
      1. Photos

2. Text description

iv. These will serve as references to the property before rental (for later assessments of damages) and provide support for the secondary goal of finding property to rent/finding corenters. This information is stored in the database under the corresponding property.

v. At this point, the landlord has successfully registered a new property. The database following this action is as follows:
>> (landlord) STEVE CARELL
>> (property) THE OFFICE
>> (deed) PHOTO
>> (insurance) PHOTO
>> (address) 0123 HOLLYWOOD AVE...
>> (photos) PHOTO, PHOTO1
>> (text description) Meet your new office…
>> (renter information) null
>> (third-party recommendations) null
>> (security deposit) null
>> ($ balance) null
>> (calendar) null
>> (alerts) null
>> (requests) null
>> (notice) null
>> (record)

vi. The landlord is directed to a home page, as specified by “Landlord-Portal” in mockups.

vii. When a new property is created, a new record is created. The record holds all information about the property and logs changes to ALL these fields.

b. *Sub-scenario:* Signing new renters (setup)
i. In the event a landlord signs new renters to a property, he will be provided a page to populate the db fields above in blue. In secondary implementation, contract, warnings/fines, and other fields will be added. The page will show:
1. A place to link to the renter’s account.
2. A text box to provide third-party recommendations such as utility companies
3. A place to charge security deposit
4. A place to charge any amount/schedule recurring charges

ii. A start flag is created in the property log to signify the start of a new rental period

c. *Sub-scenario:* Ending a rental contract
   i. When a tenant sends the “GIVE NOTICE” alert, it populates the NOTICE field of the property, triaged with high importance in ALERTS, and is displayed on the landlord’s home page.

   ii. When a contract period is completed and the contract will not be renewed OR a contract is terminated due to subletting, the following steps are completed:
      1. All scheduled alerts/charges are halted (see next section for payment handling)
      2. A page?/form? is provided to the landlord to close the renter/landlord relationship. He will wait to submit this until all payments have been collected. This page will provide an option to refund the security deposit.

   iii. An end flag is created in the property log to signify the end of a rental period.

d. *Sub-scenario:* Managing current renters
   i. After login, a landlord is directed to a homepage. From this page he can view alerts, requests, payment information sent to him by the tenants (this is covered in “Tenant Scenarios”). From this page the landlord can navigate to specific property pages, then input data including:
      1. Alerts (scheduled/custom)
      2. Appointments
      3. $ Charges

   ii. Alert and appointment information is sent to the database, then directed to the tenant portals. Money transfers are handled with an API (see “interface to existing systems” section).

   iii. A landlord on a property page should also be able to view (and print) a running log of the rental history by clicking a button.

   iv. All changes are tracked in the property record.

4. Tenant Scenarios:

a. *Payment*:
   i. *Scenario:* Verified tenants are logged in successfully, they should find a tab in their homepage that directs them to the payment page. Once the

tab is clicked, they should be able to see a table that has the following tabs:

1. Make a payment
2. Recent activity
3. Payment accounts

ii. *Sub-scenario 1.1*: Make a payment Tab

1. Tenants are on the payment page, and they clicked on the make payment tab, they should see a page that ask them the following inputs:
    a. The upcoming payment due date.
    b. The amount.
    c. A pay now button that directs them to the payment gateway page.
2. *Sub-scenario:* 1.1.1 : Make a payment Tab, clicking on pay now button
    a. Tenants are on the make payment tab, and they clicked on pay now button, they should be directed to the payment gateway that display a payment form that has the following inputs:
        i. Select payment account
        ii. Payment amount
        iii. Submit button

iii. *Sub-scenario 1.2* Recent activity Tab

1. Tenants are on the payment page, and they clicked on the recent activity tab, they should see a table that outputs all the following information
    a. Payment Date
    b. Amount
    c. Payment Account

iv. *Sub-scenario:* 1.3 : Payment Accounts Tab

1. Tenants are on the payment page, and when they click on the payment accounts tab, they should see the following inputs:
    a. all the accounts they use to make their payments
    b. a button to add more payment accounts.

b. *Request*:

i. *Scenario 1:* Verified tenants are logged in successfully, they should find a tab in their homepage that directs them to the request page. Once the request tab is clicked, they should be able to see a table with two tabs:

1. submit a request
2. view request history.

ii. *Sub-scenario 1.1:* Submit a Request Tab

1. Tenants are on the request page, and they see a table with two tabs and they clicked submit a request tab , they should be able to see a request form that has the following inputs:
    a. Select priority: { Emergency, High, Medium, Low }
    b. Category: { Appliances, Electrical, Flooring, HVAC, Locks,keys, doors, Plumbing, Snow removal, Trash, Walls, ceilings, Windows, Other}.
    c. Full Description
    d. Upload an attachment
    e. Permission to enter property: { yes, no}

iii. *Sub-scenario 1.2*: Request activity tab
1. Tenants are on the request page, and they see a table with two tabs and they clicked request activity tab , they should be able to see a table that tracks all tenants requests with following output:
    a. request id
    b. request date.
    c. Category.
    d. Description.
    e. Status.
    f. completion date.
    g. Maintenance note.
    h. attachment.

c. *Schedule*:
i. *Scenario 1:* Calendar View
1. Verified tenants are logged in successfully, in the homepage,they should be able to see all appointments made by their landlord and all their maintenance requests in their calendar.

ii. *Scenario 2:* Alerts view
1. Verified tenants are logged in successfully, in the homepage,they should be able to see all alerts made by their landlord. These alerts outputs the following information:
    a. Notice alerts
    b. Fine alerts
    c. Maintenance alerts
    d. Inspection alerts
    e. Package pickup alerts

List of Features To Implement

MVP Requirements (plan to accomplish. Required*):

- Tenant portal
    - Pay the rent (API)
    - Any kind of requests
        - Regular (door hinge broken, pick up package, etc.), Emergency (water leaking), Change (painting on wall). Each request will have a priority.
        - Log of previous requests and maintenance
    - Contract history (payments/rent changes)
    - Calendar for viewing appointments with landlord (inspections) and outside contractors.
    - Alerts (rent, weather, upcoming maintenance)

- Landlord portal
    - Charge Rent
    - Calendar for viewing and scheduling visits, showings, maintenance, etc.
    - Give move-out notice.
    - Can provide utilities recommendations.
    - Should be able to view every service needed for each property. Some have to pay utilities themselves, so provide a way to keep track of those payments.
    - Todo list for payments, collection, inspections, maintenance.
    - Claim a home -- similar to Zillow, you can input an address and unit, and if it is unclaimed you can claim it as a home you own and rent it out. We can't create a detailed authentication system for this since this requires licenses and other requirements, but we can model how it would work (eg. making the claim, filing a counterclaim if someone else has already claimed it and you believe it was an error).

Secondary (would be nice to implement, but probably will not get to most because of the shorter semester. Optional*):

- Tenant portal
    - Sign the contract digitally and submit images of the initial state of the unit (send to all signers/co-signers)
    - Social functionality for tenants: users can find other users and submit an application for a unit together.
    - Page to view the contract.
    - Page to view property expectations and their consequences:
        - this should be updatable (from landlord-side)
        - able to handle seasonal expectations, such as turn on/shut off irrigation
        - alerts when expectations changed

- Vendor portal

    - Vendors can make accounts, and bid on contracts from landlords. They would have a log of previous work, and landlords could have a “friends”/trusted network for vendors they’ve worked with before.

- Landlord portal
    - Publish the contract.
    - Provide property expectations and consequences.

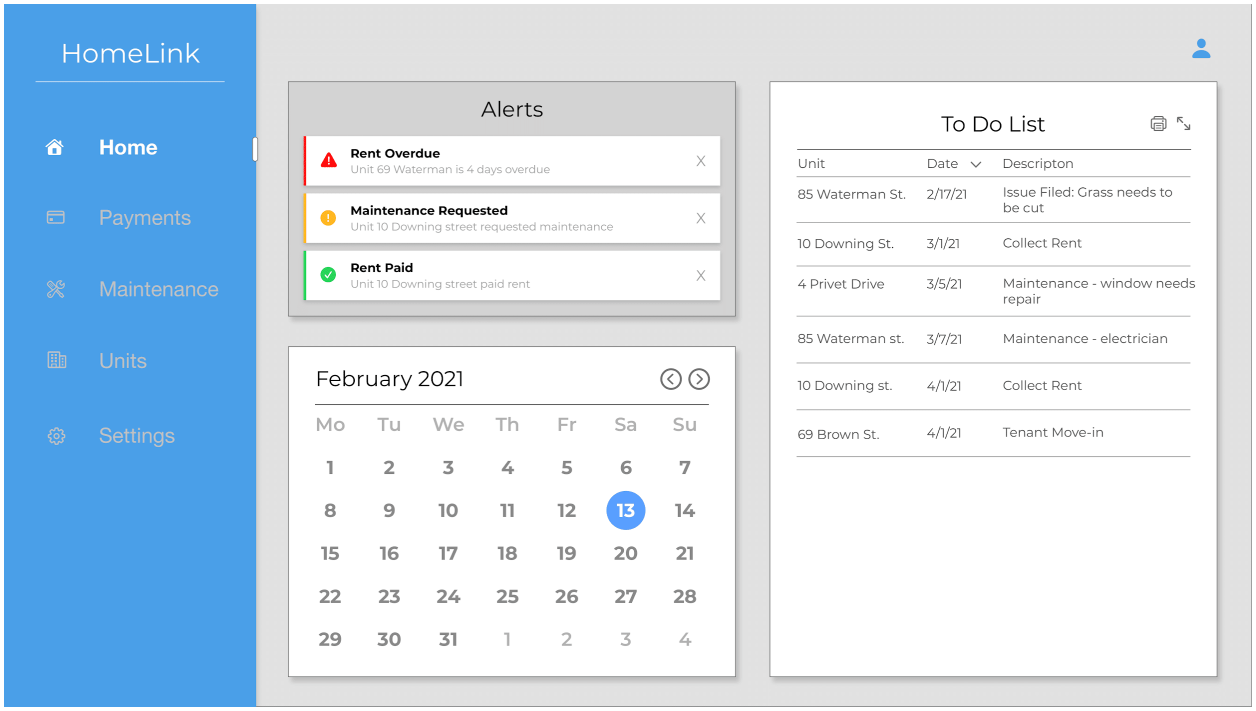
Tertiary (unlikely to implement, but if we could that would be great. Optional*):

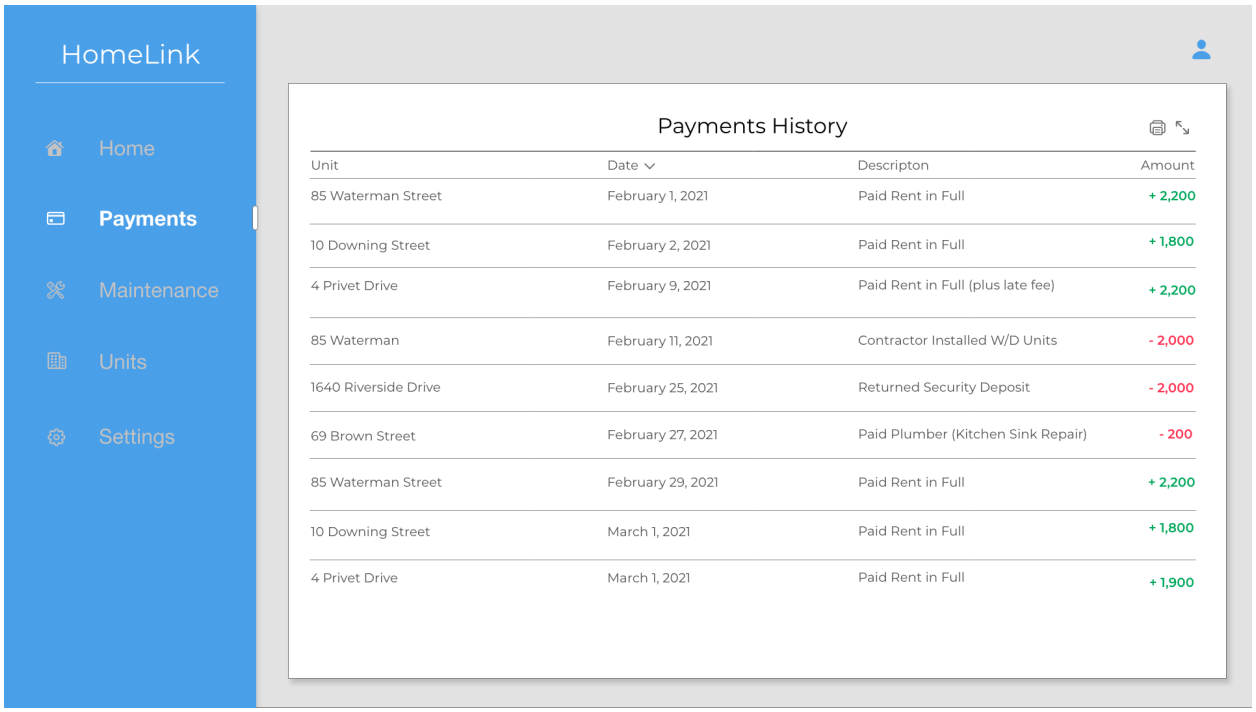
- Tenant portal
    - Subletter can pay the landlord directly, or pay the primary tenant through the website.
    - Virtual home inspection.
    - Notifications for upcoming visits or maintenance (sent to email or phone).
    - Integration with Zillow to view houses on a map.

- Landlord portal
    - After inspection landlord can upload report AND issue warnings or fines:
    - Multiple people handling the same property.
    - View-only account for those with a managing company.
    - Notifications for upcoming visits or maintenance (sent to email or phone).
    - Integration with Zillow to view houses on a map.

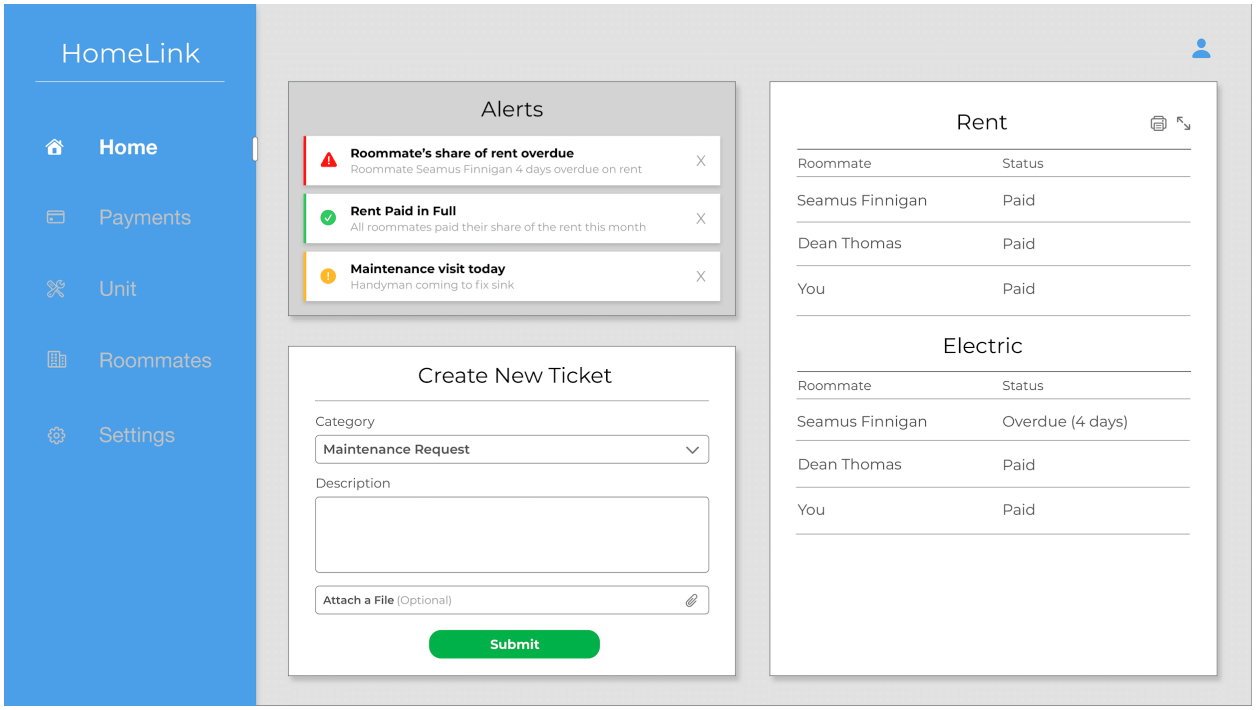
Defining the User Experience

Sketches of web pages (not final design)





These first two are the landlord-side.



This is the tenant-side.

Identify Interface to Existing Systems

1. PostgreSQL on AWS
2. Google Firebase for Authentication
3. FullCalendar Integration
4. PayPal/Mastercard/Visa API
5. Google Maps API (address validation)
6. Open Street Maps API (if we choose to pursue the social roommate-finder secondary goal)

Outline of web site and pages

- Tenant Screen
  This is the primary point of interaction for someone who is looking to rent a home, or is currently renting one. They will have:

  - A primary dashboard, which will include the calendar/list, the requests form, and an alert section.
  - A payment page, where they can pay the landlord.
  - A contact page, with important numbers.
  - A history page, as described in the prior section.

- Landlord Screen
  This is the primary point of interaction for the landlord.
  - A primary dashboard, which will include the calendar and todo list.
  - A page which lists the homes that one owns, as well as the relevant details about each.
  - A page for the landlord to claim a home and add it to their account.

Detail what the application will do

1. Pay/Charge Rent
    a. Integrate with Paypal/Mastercard/whatever financial provider API for the tenant to pay the landlord.
    b. Inputs: the rent amount (from the landlord), recurrency of charge and payments (with different options, like weekly or monthly),
    c. Outputs: Notification of payment for tenant, notification of receipt for landlord.
2. Make/Receive a Request
    a. Submit a new entry into the requests database, and notify the landlord of the new request. Landlord is prompted with a response option, through which they can inform the tenant that they've seen the issue and are dealing with it. If it's an emergency request, we could send a mobile notification to the landlord's number through a firebase API (not an MVP requirement because we haven't decided if we should instead prompt the user to call the landlord or call the police for an emergency).
    b. Inputs: the description and relevant data
    c. Outputs: Notification for landlord of receipt, notification for tenant of receipt.
3. View History
    a. Both sides should be able to view different rent and request histories. This is just a matter of reading from the database.
    b. Inputs: Filters for date, user, and type.
    c. Outputs: The relevant list.
4. View Calendar
    a. The landlord can view a calendar of upcoming events or appointments across all of their properties. This is a matter of pulling the list of relevant events from the database and populating a calendar through the FullCalendar integration.
    b. Inputs: The day, week, or month of the relevant user.
    c. Outputs: A calendar of events.
5. View Alerts
    a. Both groups can view relevant alerts about events. These can be upcoming rent payments, overdue notices, maintenance request status, etc.
    b. Inputs: None really, just based on the user that logged in.
    c. Outputs: The list of alerts.
6. View/Claim Properties
    a. The landlord can enter an address and claim the property as theirs, in order to link it to renters. They can enter the address into a validator, after which they'll be able to claim it and add it to their account. If it's already claimed, they can make a counter-claim. The resolution for this sort of system isn't within the scope of this class, but we'll mimic the basic elements. If we have the time, as a secondary goal we'd link this with Zillow to have the actual homes displayed on a map and claimable.
    b. Inputs: The address.

c. Outputs: The “home” at that location. In reality, when a user claims a home it’ll be added to the database as one of their homes, and then they can add renters to that home. We’ll have pretty tight control policy around the landlord owning multiple homes to make sure tenants are assigned the right home.