Team 6:

Cris Kedmenec
Chris Graf
Daniel DeChiara
Nicholas Vitovitch
Caleb Everett
Teddy Sudol

Large Project Elaboration - Ruby on Rails

Concept:

Whether you're a University student or an apartment-dwelling city slicker, roommates tend to cause problems. Little things like dividing up chores, splitting bills, and syncing schedules is next to impossible when everyone is operating at different speeds. The simple act of communication breaks down and can cause problems within a dwelling.

Additionally, searching for roommates and living quarters is also a difficult task. If you don't already know a person, a Google search can point you to a facebook page and a Linked-In profile, but nothing indicative of sleep schedule, work ethic, or cleanliness.
A tool that addresses all of these issues would be helpful.

Applications:

- Facilitating communication between roommates.
- Fairly allocating household chores.
- Tracking bills and creating payment reminders.
-

Legal Concerns:

The Roomie application will not face any issues regarding the privacy of a client. The application will not require any individual to upload private information, and the user will have complete control over what they upload. The user will also have the power to regulate who can find, view or request to be associated with them.

The functionality that Roomie will provide does not require any sensitive information to be used or uploaded. The bill pay function allows people of a common dwelling to upload a pdf image of the bill. It is completely up to the users whether or not they use this functionality and if they want to remove sensitive information from the bill prior to uploading it. Uploaded bills will be saved in a single S3 instance and will be set to expire once everyone has been marked off as paid. Furthermore, only members of the dwelling will have permissions to access this bucket with read only permissions. Each dwelling's information will be private to solely its members. This includes shopping lists, chores lists, calendars and bills.

Like other online social media applications, each user will have to accept a privacy agreement before signing up to use this application. This agreement will cover all the bases of information privacy, and the user must understand that they are responsible for all information they upload. Any user who may chose to upload, download or make any information public will be held solely responsible for that information and their choice to do so.

Content posted within a dwelling is private to subscribers of said dwelling, and will not be accessible by other members of the Roomie community. What is posted within a dwelling is up to the member's of it, and it will not be accessed for any intent.

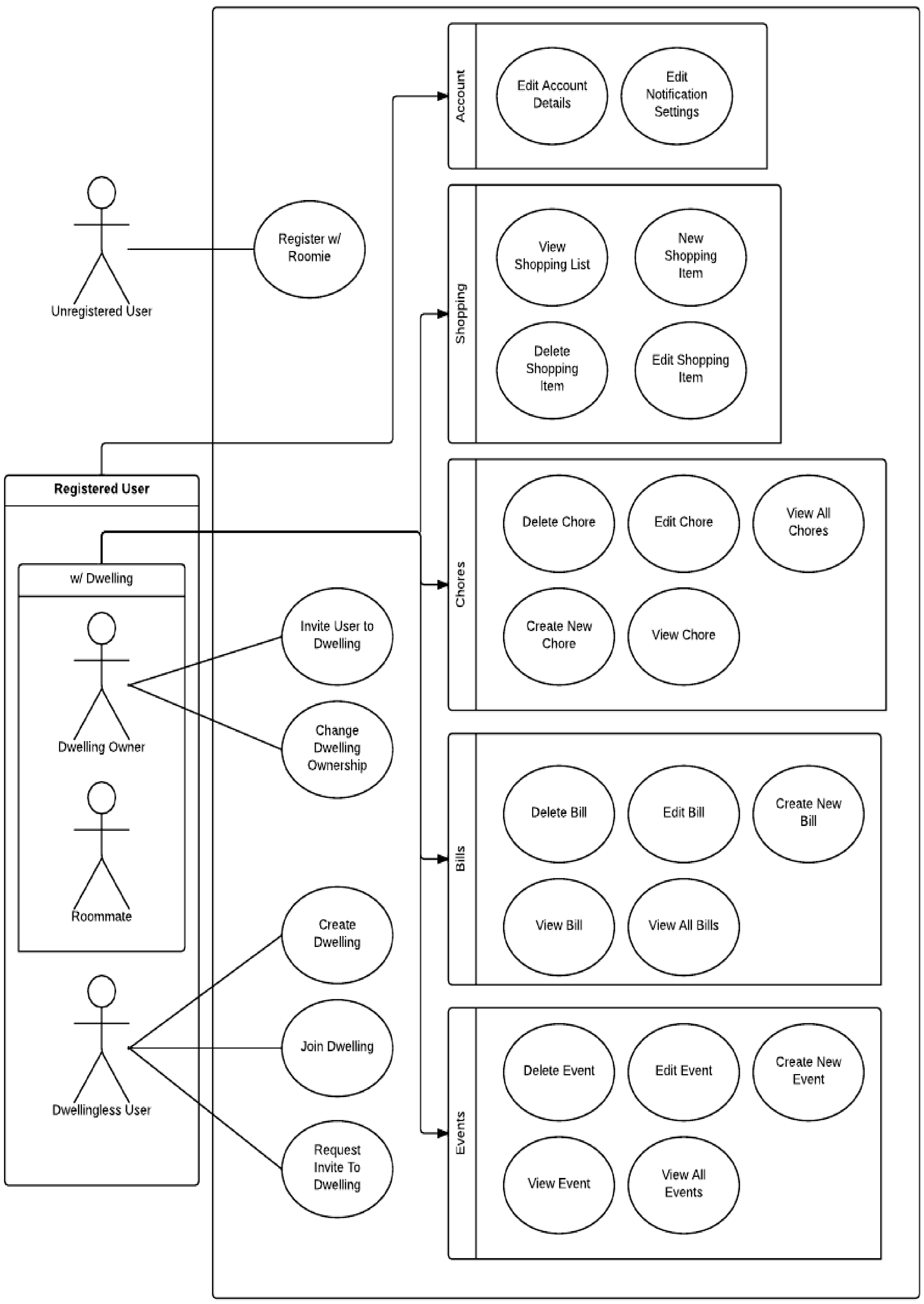
The location of a dwelling and the status of whether it is 'open or closed' will be public, to make it searchable by individuals interested in finding a place to live.

If a user is interested in deleting their account, then the Roomie application will remove all information and connections privy to it.

Any user of Rommie will have to consent to acknowledging that they are of the legal age of 18 to use this application. Roomie is a social media outlet centered around an individual's current living situation, and will not allow people under the legal age to own a private residency to make an account.

Currently, we are considering adding functionality to have a user rating system for roommates. This is just an idea, however would be the main consideration on the legal front. We would allow users to chose whether they want to allow themselves to be rated or rate others. We would make it known that we do not support, claim or will be held responsible for any statement made by a user of our application, as they are simply using a product that we provide and can make or accept data about themselves on their own accord.

Use Case Diagram


Unregistered User
Register w/ Roomie
Account
Edit Account Details
Edit Notification Settings
Shopping
View Shopping List
New Shopping Item
Delete Shopping Item
Edit Shopping Item
Registered User
w/ Dwelling
Dwelling Owner
Roommate
Dwellingless User
Invite User to Dwelling
Change Dwelling Ownership
Create Dwelling
Join Dwelling
Request Invite To Dwelling
Chores
Delete Chore
Edit Chore
View All Chores
Create New Chore
View Chore
Bills
Delete Bill
Edit Bill
Create New Bill
View Bill
View All Bills
Events
Delete Event
Edit Event
Create New Event
View Event
View All Events

Use Case: Create Account

Actors	● Unregistered User (Looking to create account) ● Registration Form ● RDB (Handles creation of account in DB) ● SES (Confirmation Email Service) ● S3 (User Account Image) ● Dashboard
Triggers	User follows link (from dashboard or email) indicating they wish to join ou service.
Pre-Conditions	*opt: user has link w/ credentials to join dwelling.*
Post-Conditions	Unregistered User will create a registered account.
Normal Flow	1. User will navigate to Registration Form 2. User will fill in all fields of form with valid credentials 3. User will submit form 4. If form is improperly filled out, display an error message and return to step 2. 5. Registration Form supplies information to RDB, S3 to create user account. 6. Registration Form supplies information to SES to create an email validation. 7. If Unregistered User cannot find email, they can request that ema be re-sent from Registration Form, return to step 6. 8. Unregistered User navigates to personal email client and follows confirmation link in email. 9. User account is confirmed, verification is logged in RDB 10. User is forwarded to Dashboard

Use Case: Edit Account Details

Actors	● Registered User ● RDB (Handles creation of account in DB) ● S3 (User Account Image)
Triggers	User clicks Account Settings link.
Pre-Conditions	User is logged in.
Post-Conditions	User Information will be updated.
Normal Flow	1. User modifies account information 2. User submits form 3. If data is valid the new account information is stored in the database, otherwise the form is rendered, with error messages explaining what the user needs to fix

Use Case: Edit Notifications

Actors	● Registered User ● Notification Settings Form ● RDB (Handles creation of account in DB) ● SES (Handles email notification) ● SMS (Handles text notificaitons)
Triggers	User clicks Settings > Notifications link.
Pre-Conditions	User is logged in.
Post-Conditions	User notification settings will be updated.
Normal Flow	1. User indicates notification settings in Notification Settings Form 2. User submits form. 3. Settings are logged in RBD 4. SES is updated to reflect desired settings 5. SMS is updated to reflect desired settings

Use Case: Create Dwelling

Actors	● Registered User with no dwelling ● RDB (Handles creation of account in DB) ● S3 (User Account Image)
Triggers	User clicks Create Dwelling link
Pre-Conditions	User is logged in, and has no Dwelling
Post-Conditions	User will create a dwelling, which they will be the owner of.
Normal Flow	1. User fills in dwelling information. 2. User submits form. 3. If data is valid dwelling is created, and user is set as the owner of that dwelling otherwise the form is rendered with error messages, user goes back to step one 4. User is redirected to the dwelling dashboard

Use Case: Join Dwelling

Actors	Registered User with no dwelling RDB (Handles creation of account in DB) S3 (User Account Image)
Triggers	User clicks Join Dwelling link

Pre-Conditions	User is logged in, and has no Dwelling
Post-Conditions	User will be added to a dwelling, and directed to the dashboard for that dwelling
Normal Flow	1. User clicks link to join a dwelling 2. User joins the dwelling

Use Case: Invite Users to Dwelling

Actors	Dwelling owner RDB (Handles creation of account in DB) S3 (User Account Image)
Triggers	Dwelling owner invites someone to dwelling
Pre-Conditions	User is logged in, and the owner of a dwelling
Post-Conditions	A new user will be added to the dwelling
Normal Flow	1. Dwelling owner enters a list of email addresses to be invited to join the dwelling 2. The email will prompt the recipient to create an account, or sign ir if they have one. 3. that account will be added to the dwelling

Use Case: Change Dwelling Ownership

Actors	Dwelling owner Dwelling Member RDB (Handles creation of account in DB)
Triggers	Dwelling owner asks to transfer ownership of the dwelling to another member
Pre-Conditions	User is Dwelling owner, another dwelling member exists
Post-Conditions	User permissions are updated, new user is dwelling owner
Normal Flow	1. Dwelling owner selects a new user to be the new dwelling owner 2. Other user accepts the position 3. Permissions are updated in the database
Alternate Flow	1. Dwelling owner selects a new user to be the new dwelling owner. 2. Other user denies the position. 3. Dwelling owner is alerted, and the database is not updated.

Use Case: View Shopping List

Actors	• Registered User • Dwelling • RDB (Contains all Shopping Lists)
Triggers	User clicks Dashboard > List link.
Pre-Conditions	User is logged in and belongs to a Dwelling.
Post-Conditions	User is viewing the shopping list for their dwelling.
Normal Flow	1. RDB is queried for the shopping list of Dwelling. 2. Results are displayed in browser as a list of items.

Use Case: New Shopping Item

Actors	• Registered User • Dwelling • New Shopping List Form • RDB (Contains all Shopping Lists)
Triggers	User clicks Dashboard > List > New link.
Pre-Conditions	User is logged in and belongs to a Dwelling.
Post-Conditions	User has created a new shopping list.
Normal Flow	1. User fills out information form for the list. 2. RDB stores the list data.

Use Case: Edit Shopping Item

Actors	• Registered User • Dwelling • RDB (Contains all Shopping Lists)
Triggers	User clicks Edit link in List description.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the List exists.
Post-Conditions	The list has been updated.
Normal Flow	1. User enters new information in list information form 2. RDB updates the list entry in the database

Use Case: Delete Shopping Item

Actors	• Registered User • Dwelling

	● RDB (Contains all Lists)
Triggers	User clicks Delete link in Bill description.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the List exists.
Post-Conditions	The list has been deleted.
Normal Flow	1. User confirms the list deletion 2. RDB removes the list entry

Use Case: View All Bills

Actors	● Registered User ● Dwelling ● RDB (Contains all Bills)
Triggers	User clicks Dashboard > Bills link.
Pre-Conditions	User is logged in and belongs to a Dwelling.
Post-Conditions	User is viewing bills for their dwelling.
Normal Flow	3. RDB is queried for bills of Dwelling. 4. Results are displayed in browser as a list of bills with due dates and amounts.

Use Case: View Bill

Actors	● Registered User ● Dwelling ● RDB (Contains all Bills)
Triggers	User clicks Dashboard > Bills > Bill link.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the Bill exists.
Post-Conditions	User is viewing the bill description.
Normal Flow	1. RDB is queried for the specifics of selected bill. 2. The bill is formatted by Rails 3. Results are rendered to browser.

Use Case: New Bill

Actors	● Registered User ● Dwelling ● New Bill Form

	● RDB (Contains all Bills) ● S3 (Stores extra bill content)
Triggers	User clicks Dashboard > Bills > New link.
Pre-Conditions	User is logged in and belongs to a Dwelling.
Post-Conditions	User has created a new bill.
Normal Flow	3. User fills out information form for the bill 4. RDB stores the bill data 5. S3 stores user-uploaded content for bill

Use Case: Edit Bill

Actors	● Registered User ● Dwelling ● RDB (Contains all Bills) ● S3 (Stores extra bill content)
Triggers	User clicks Edit link in Bill description.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the Bill exists.
Post-Conditions	The bill has been updated.
Normal Flow	3. User enters new information in bill information form 4. RDB updates the bill entry in the database 5. New content stored in S3

Use Case: Delete Bill

Actors	● Registered User ● Dwelling ● RDB (Contains all Bills) ● S3 (Stores extra bill content)
Triggers	User clicks Delete link in Bill description.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the Bill exists.
Post-Conditions	The bill has been deleted.
Normal Flow	3. User confirms the bill deletion 4. RDB removes the bill entry 5. Content in S3 is deleted

Use Case: View All Chores

Actors	● Registered User ● Dwelling ● RDB (Contains all Chores)
Triggers	User clicks Dashboard > Chores link.
Pre-Conditions	User is logged in and belongs to a Dwelling.
Post-Conditions	User is viewing chores for their dwelling.
Normal Flow	1. RDB is queried for chores local to Dwelling. 2. Results are displayed in browser as a list of upcoming chores anc chore rotations.

Use Case: View Chores

Actors	● Registered User ● Dwelling ● RDB (Contains all Chores) ● S3 (For any rich content associated with this chore)
Triggers	User clicks Dashboard > Chores > Chore link.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the Chore exists.
Post-Conditions	User is viewing bill description.
Normal Flow	4. RDB is queried for bill specifics of selected chore. 5. S3 serves up rich content associated with chore. 6. Results are rendered to browser.

Use Case: New Chore

Actors	● Registered User ● Dwelling ● New Bill Form ● RDB (Contains all Chores) ● S3 (For any rich content associated with this chore)
Triggers	User clicks Dashboard > Chores > New link.
Pre-Conditions	User is logged in and belongs to a Dwelling.
Post-Conditions	User has created a new chore.
Normal Flow	6. User fills out information form for the 7. RDB stores the chore data 8. S3 stores user-uploaded content for the chore

Use Case: Edit Chore

Actors	● Registered User ● Dwelling ● RDB (Contains all Chores) ● S3 (For any rich content associated with this Chore) ● EC2 (generates rich content)
Triggers	User clicks Edit link in Chore description.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the Chore exists.
Post-Conditions	The chore has been updated.
Normal Flow	6. User enters new information in chore information form 7. RDB updates the chore entry in the database 8. New content stored in S3

Use Case: Delete Chore

Actors	● Registered User ● Dwelling ● RDB (Contains all Chores) ● S3 (For any rich content associated with this chore)
Triggers	User clicks Delete link in Chore description.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the Chore exists.
Post-Conditions	The chore has been deleted.
Normal Flow	6. User confirms the chores deletion 7. RDB removes the chore entry 8. Rich content in S3 is deleted

Use Case: View All Events

Actors	● Registered User ● Dwelling ● RDB (Contains all Events)
Triggers	User clicks Dashboard > Events link.
Pre-Conditions	User is logged in and belongs to a Dwelling.
Post-Conditions	User is viewing events for their dwelling.
Normal Flow	5. RDB is queried for events local to Dwelling. 6. Results are displayed in browser as a list of upcoming events and as a calendar.

Use Case: View Event

Actors	● Registered User ● Dwelling ● RDB (Contains all Events) ● S3 (For any rich content associated with this event)
Triggers	User clicks Dashboard > Events > Event link.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the Event exists.
Post-Conditions	User is viewing event description.
Normal Flow	7. RDB is queried for event specifics of selected event. 8. S3 serves up rich content associated with event. 9. Results are rendered to browser.

Use Case: New Event

Actors	● Registered User ● Dwelling ● New Event Form ● RDB (Contains all Events) ● S3 (For any rich content associated with this event)
Triggers	User clicks Dashboard > Events > New link.
Pre-Conditions	User is logged in and belongs to a Dwelling.
Post-Conditions	User has created a new event.
Normal Flow	9. User fills out information form for the event 10. RDB stores the event data 11. S3 stores user-uploaded content for event

Use Case: Edit Event

Actors	● Registered User ● Dwelling ● RDB (Contains all Events) ● S3 (For any rich content associated with this event) ● EC2 (generates rich content)
Triggers	User clicks Edit link in Event description.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the Event exists.
Post-Conditions	The event has been updated.

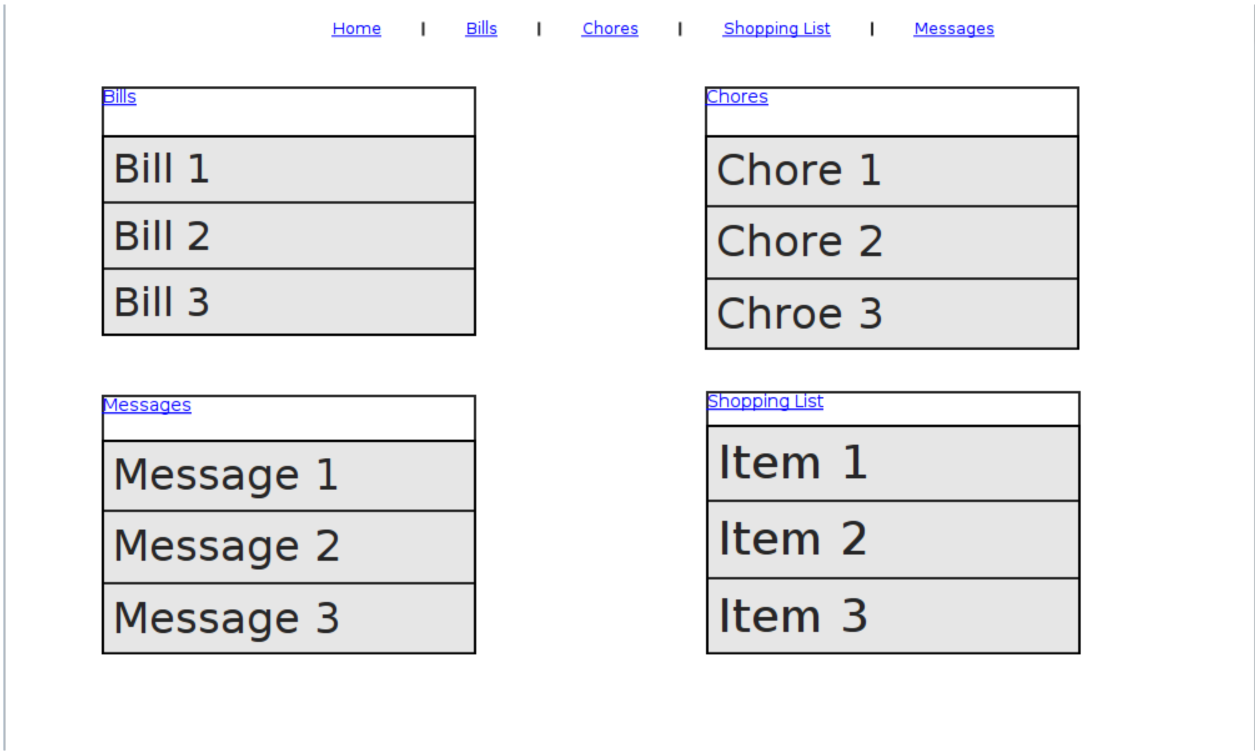
Normal Flow	9. User enters new information in event information form 10. RDB updates the event entry in the database 11. New content stored in S3

Use Case: Delete Event

Actors	● Registered User ● Dwelling ● RDB (Contains all Events) ● S3 (For any rich content associated with this event)
Triggers	User clicks Delete link in Event description.
Pre-Conditions	User is logged in, belongs to a Dwelling, and the Event exists.
Post-Conditions	The event has been deleted.
Normal Flow	9. User confirms the event deletion 10. RDB removes the event entry 11. Rich content in S3 is deleted

Mock-Up:

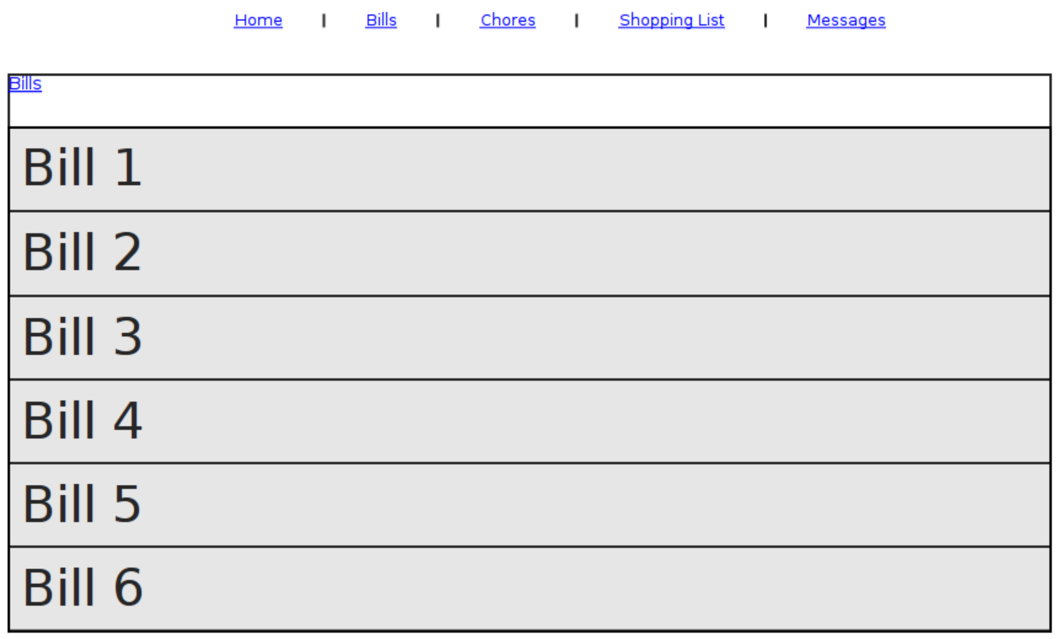
User Interface: A series of mock---up images/pages that show the user interface for your application (web, mobile, or native), with descriptive, detailed paragraphs that explain their purpose



The Dwelling home page functions similar to how a bulletin broad, or refrigerator might.

Notifications of upcoming events, important bills, messages from roommates, and shopping lists are posted here. From the dashboard users can add new items, and check the status of upcoming items.

Users can view details on each item by clicking on them in the lists, or they can view all items of a topic by clicking on the section header.


Home | Bills | Chores | Shopping List | Messages

Bills

Bill 1

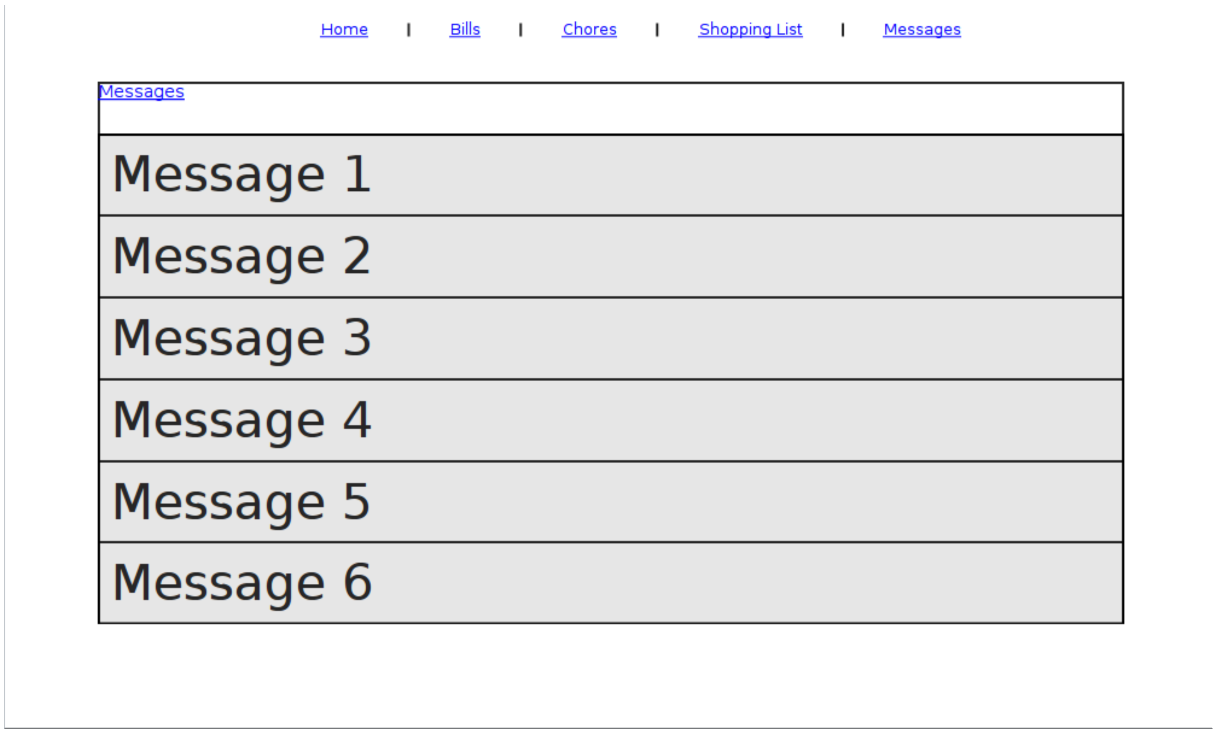
Bill 2

Bill 3

Bill 4

Bill 5

Bill 6


Home | Bills | Chores | Shopping List | Messages

Messages

Message 1

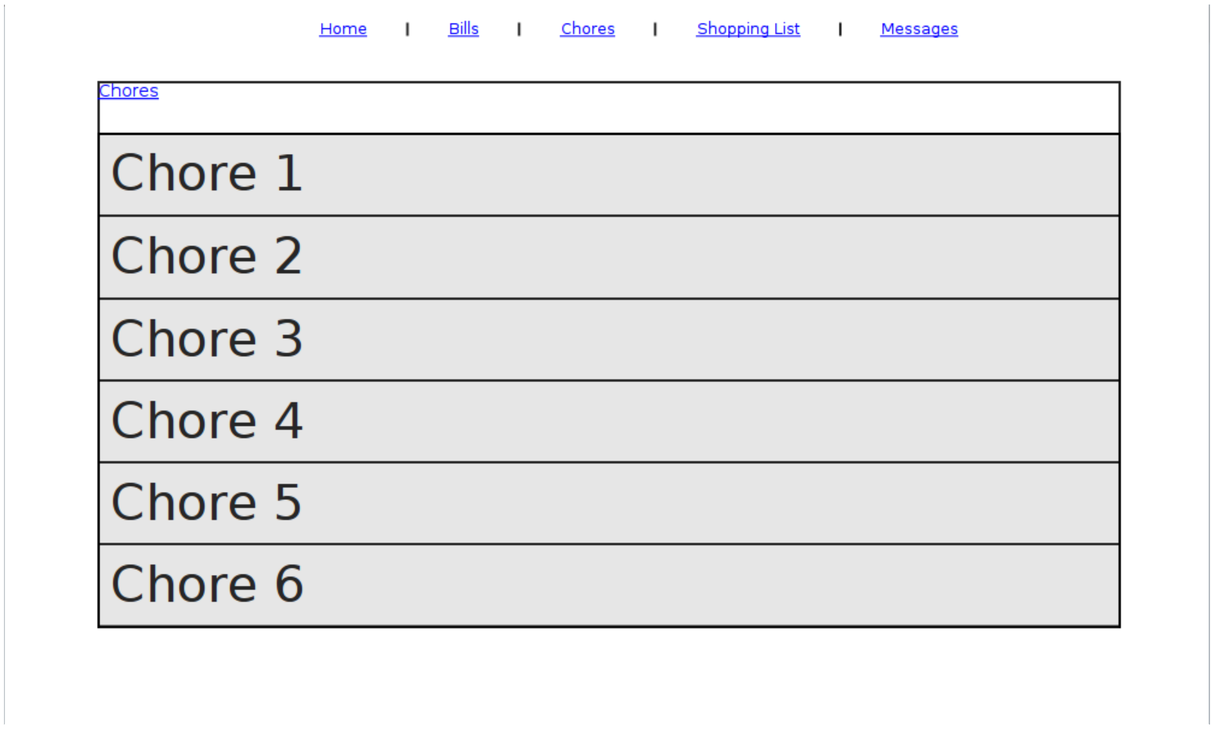
Message 2

Message 3

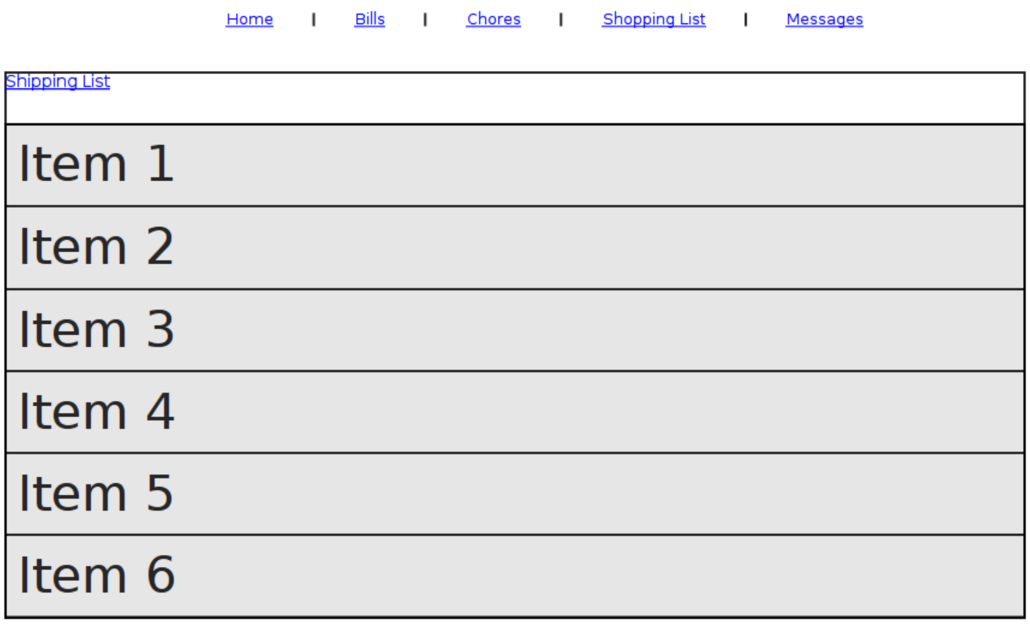
Message 4

Message 5

Message 6

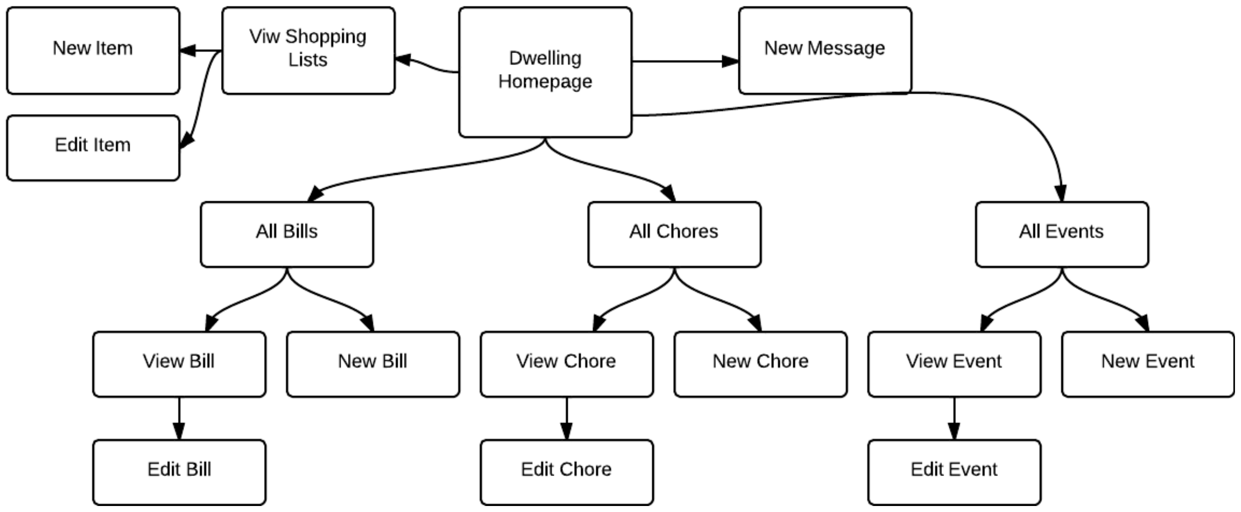


Chore 1
Chore 2
Chore 3
Chore 4
Chore 5
Chore 6



Item 1
Item 2
Item 3
Item 4
Item 5
Item 6

Site Structure: A proposed site structure (if web application) chart (see http://webstyleguide.com/wsg3/3---information---architecture/3---site---structure.html) Use an UML---aware tool, like Lucidchart, to draw the diagrams in an applicable UML notation.
Further refine and publish the milestones detailing the subtasks and resource allocation for each stage. Create tickets to assign tasks to each team member.



The site will be setup so users will be able to use the dashboard for most activities, and they will only have to visit other pages when doing administrative work. Users will be able to view information about upcoming events, bills and chores, see what is on the shopping list, and see messages from their roommates on the dashboard.