Contents

Andrew, Connor, Matt, Jonathan

1. System description
2. Use Cases [include non-functional requirements], Case Diagram, Conceptual Diagram
3. Updated functional requirements after feedback from presentation for your reference at the end of the document

System Description

Our system allows users to design a house and location from scratch. The process is the system will display a 3 step process. First the user designs the exterior of the home using a CAD like interface. When the user is done, they will be taken to a new system which will load the Exterior or house blueprint in 3D and allow the user to customize the interior with furnishings. Finally for step 3, the user will be taken to a new page where they can filter locations to find the perfect one for their house. All this information will be communicated and displayed in a ‘stakeholder’ portal, a web page where banks, suppliers, and customers can go to get relevant information. This was split up this was as it makes for a logical workflow for the user and allows us 4 to design systems that will be connected in some aspects maintain a degree of independence so we are not waiting for each other to do work and can work in parallel.

4 systems each with 4 functional requirements and respective use cases are as follows

1. Andrew -> exterior house design
2. Connor -> interior house design
3. Matt -> location picker
4. Jonathan -> stakeholder portal

Exterior House Design

**Use Case:** adding a wall to the floorplan

**Primary Actor:** user (designing/ordering the house)

**Stakeholders and interests:**

-> user needs to be able to construct a floor plan to design the structure of the house

-> software designer wants to create UI easy to use

**Preconditions:**

-> User is on the floor plan design page (2d graphing paper page) of the process

**Success Guarantee (post condition):**

-> wall is added to the visual picture in 2-dimensional format

**Main success scenario (interaction of actors, system validation, system state change):**

1. User is on the floor plan design page (2d graphing paper page) of the process

Repeat 2-6 until user indicates done add walls to floor plan

2. User indicates a starting point by clicking anywhere on the design page at first, or at an adjacent wall.

3. While holding down the click, user may drag the cursor to indicate the length and direction of the wall/line

4. A text editor pops up by the cursor indicating length in feet, user may indicate how long this wall should be by typing in the text editor.

5. The software cross checks the compatibility of 2-d wall design to make sure it is valid

6. A “successfully added wall” notification appears and the floor plan is updated.

7. When user exits the system, the system saves changes and returns the user to the home page

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails):**

4a) The user types in an invalid length (character, too big, too small)

1. Error message is displayed

2. The wall is reset to 1 ft

3. System resets to step 4, the text editor shows 1ft with an option for user to change it again

5a) The user draws an invalid floor plan (not all lines connected)

1. Error message is displayed

2. The invalid wall causing this error is highlighted in red

3. The user is given an option to delete it (using delete feature) or edit it (using edit feature)

**Special Requirements (non functional):**

-> addition of a wall should be accepted or rejected within 1 second of user creating it

-> error message should pop up instantly with rejection

**Technology and data variation list:**

-> edit bar will be populated with options to add, delete, edit walls

-> there will be a 2-dimensional modeling visual as the main page

-> it will have cross checking capabilities algorithms to check all walls are connected and possible

**Frequency of occurrence ( how often you expect this use case to occur):**

-> This is a core use case for each user in developing a floor plan when ordering/designing this house. This use case may occur 100’s of times.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case:** raising walls to create a floor (ex. First, second floor)

**Primary Actor:** user (designing/ordering the house)

**Stakeholders and interests:**

-> user wants to dictate how tall the walls/ceilings should be within the limitations of architecture

**Preconditions:**

-> User has a valid floor plan and has accepted the final floor design

**Success Guarantee (post condition):**

-> walls are raised to a specified height and the model transitions to a 3-dimensional visual

**Main success scenario (interaction of actors, system validation, system state change):**

1. User indicates floor plan is done, a notification pops up prompting the user to raise the walls to create a floor

2. User indicates the height of the walls in a text editor

3. The 2D floor plan transitions into a 3D model

4. User given options to edit the external walls to add features or modify what material to be used

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails):**

2a) User indicates an incompatible height of the walls (character, too short, or too tall)

1. An Error message pops up
2. System reverts back to the 2D design
3. System prompts the user to enter the height of the walls again

**Special Requirements (non functional):**

-> the transition from 2d to 3d model executes within 1 second

->

**Technology and data variation list:**

-> Currently user has to user point and click to manipulate walls, but as AR/VR advances will be able to use gestures

-> data will be saved as json and .CAD file where .CAD is a custom file type for graphics

**Frequency of occurrence ( how often you expect this use case to occur):**

-> a couple times for each customer (person ordering the house) as they decide which floor plan they would want to design 3 dimensional features to it.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case:** delete a wall from floor plan

**Primary Actor:** user (designing/ordering the house)

**Stakeholders and interests:**

-> user wants the flexibility to change design decisions and remove walls

**Preconditions:**

-> User is on the floor plan design page (2d graphing paper page) of the process

**Success Guarantee (post condition):**

-> The desired wall is removed from the visual and the user has the freedom to add, delete, edit more walls on page.

**Main success scenario (interaction of actors, system validation, system state change):**

1. User selects a wall to edit

2. System gives user options to delete or modify the selected wall

3. User selects to delete it

4. The 2D model removes the wall from the visual

5. User is free to add, edit, remove walls or indicate that they are finished with the 2D floor plan.

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails):**

3a) User selects to delete a wall that makes a valid floor plan to an invalid/unfinished floor plan

1. A message appears asking the user if they are sure they want to delete this wall
2. User has the option to select yes/no

2a. If selected yes, system removes wall and continues

2b. If selected no, system does not remove wall and reverts back to the previous floor plan

1. User is then free to add, edit, remove walls once again.

**Special Requirements (non functional):**

-> deletion of a wall and returning the user to editing mode should take less than 1 second

-> transition animation should be smooth

**Technology and data variation list:**

-> Must check compatibility of structure with a database / api and an algorithm to define floor plan as valid

**Frequency of occurrence ( how often you expect this use case to occur):**

-> This is a core use case for each user in developing a floor plan when ordering/designing this house. This use case may occur 100’s of times.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case:** selecting exterior material

**Primary Actor:** user (designing and ordering the house)

**Stakeholders and interests:**

-> user wants flexibility in choosing which material the exterior walls will be made of

-> suppliers and builders need to know what kind of material to use and how much external labor will go into building the exterior layer of home

**Preconditions:**

-> Floor plan is already mapped out and raised to the 3d model of a floor

**Success Guarantee (post condition):**

-> User can visually see the type of exterior on their designed model

-> cost of exterior material is added to the total cost

**Main success scenario (interaction of actors, system validation, system state change):**

1. User selects the wall to be modified.

2. User is given options to insert features such as windows, trims, designs, or modify its exterior material

3. User selects option to modify its exterior material

4. System gives users various options such as brick, fiber cement, cement, and various types of wood

5. User selects an option

6. The 3D model of the house is wrapped in depicted design material

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails):**

4a) User has already selected an option previously

1. The 3D model is already wrapped in depicted design material selected
2. The system allows user to select all but the previously selected option
3. The model changes depicted design material and continues

5a) User selects an exterior material not compatible with size and/or location of house

1. The system produces an error messages saying exterior material is incompatible with the type of house design
2. The system prompts user to select a different option
3. The 3D model of the house remains depicted in no design material

**Special Requirements (non functional):**

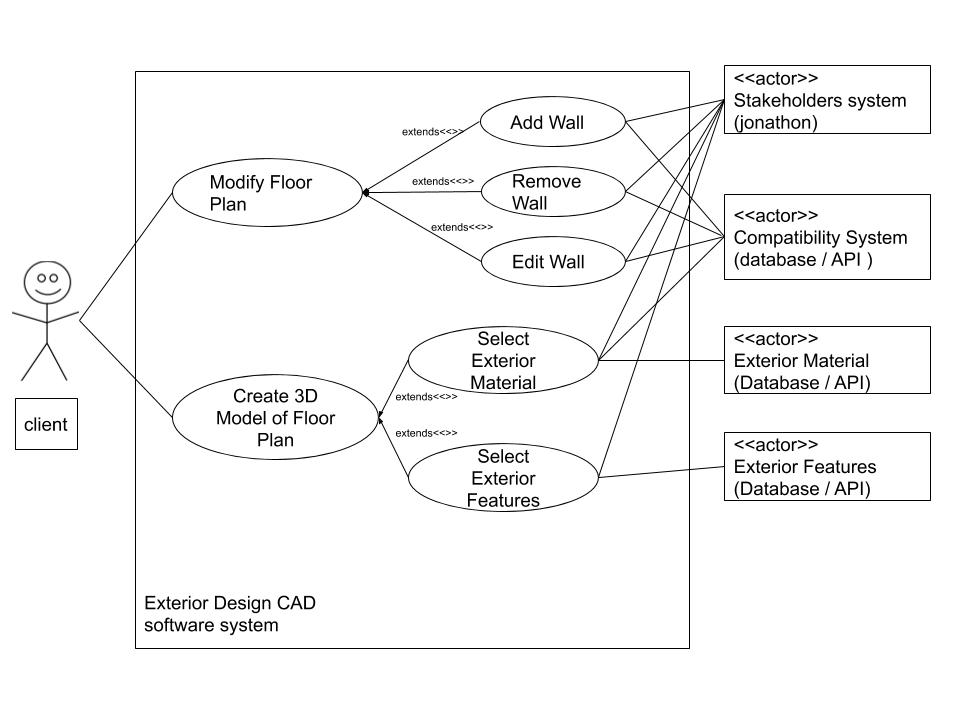
-> Once user selects a material, the 3D model shows the depiction of it in less than 1 second

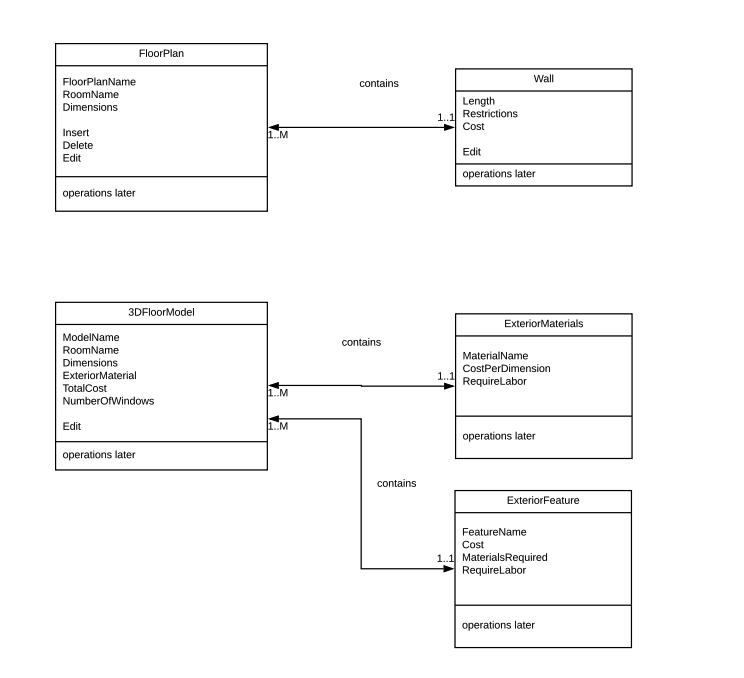
**Technology and data variation list:**

-> The model will have to cross check with a database api for compatibility with the house design and the exterior material

**Frequency of occurrence ( how often you expect this use case to occur):**

-> This will occur a couple times depending on customer wanting to see how various exterior designs look





Interior House Design

**Use Case:** adding furnishing items to a room

**Primary Actor:** the user (person who is ordering a house)

**Stakeholders and interests:**

-> user wants to easily add furnishing items to a room

-> supporting: external furnishing provider provides user with variety of choices

-> offstage: jonathan stakeholder webpage system wants updates on what user selects

-> offstage: software dev wants to make sure user has a frictionless and pleasant experience

**Preconditions:**

-> a specific room of the house blueprint ( house with exterior all finished ) is selected and displayed on the screen

-> furnishing options relevant to the option are shown in a sidebar on the right

**Success Guarantee (post condition)**

-> item is placed in room, added success notification pops up, total house cost is updated. The specific room on the screen now displays the item, materials needed for the house is updated

**Main success scenario (interaction of actors, system validation, system state change)**

1. User is on the interior design page of the process

Repeat 2 - 5 until user indicates done adding items

2. User drags an item from the furnishing sidebar into the room

3. A ‘added successful’ notification appears and the room updates to show the item in the room where the user placed it … these two happen simultaneously

4. The total cost of the house is updated

5. The items (furniture) list needed is updated with the added item

6. When user exits, system saves changes and returns the user to the home page

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

3a) the item was unable to be added to the users selected location in the room

1) system displays error message (item overlap, not enough room, item not available, etc)

2) system resets to the state at step 1 of main success scenario

6a) system fails to save the state

1)error message is displayed

2) user remains at the current page with all room modifications

2a) user tries to save again aka repeat step 6

2b) user exists application

**Special Requirements (non functional)**

-> item should be accepted or rejected within 1 seconds of the user dropping it into the room

-> if system fails to save the state and user exits, on website / application reload should try to recover the most recent state

**Technology and data variation list**

2a) the furnishing sidebar will be populated with different items depending on what room / house style is currently selected

2b) images of furnishings will be JPEG or PNG

**Frequency of occurrence ( how often you expect this use case to occur)**

->This is a core use case for each user who orders a house this may occur 100’s of times

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case:** selecting an interior style (modern, contemporary, traditional, industrial, minimalist, etc.)

**Primary Actor:** the user (person who is ordering a house)

**Stakeholders and interests:**

-> user wants to filter the furnishing list by style

-> supporting: external furnishing provider provides user with variety of choices

-> offstage: software dev wants to make sure user has a frictionless and pleasant experience

**Preconditions:**

-> furnishing options are shown in a sidebar with a single select box at the top of this sidebar

**Success Guarantee (post condition)**

-> the sidebar is filtered by the style and now only displays furnishings of selected style

**Main success scenario (interaction of actors, system validation, system state change)**

1. User is on the interior design page of the process

Repeat 2 - 4 as long as user is switching style selections

2. User selects a style from the single select box in the sidebar

3. A loading icon appears while the list is updated accordingly

4. The sidebar refreshes with a next list of furnishings all with the style selected

5. When the user exits the system, everything closes, returns user to homepage

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

4a) the sidebar fails to load furnishings with selected style

1) error message is displayed

2) loading icon doesn’t disappear

3) blank sidebar is shown

4) system resets to step 2 and message displayed

**Special Requirements (non functional)**

-> sidebar should response with response or failure within 3 seconds

-> styles with no furnishing options shouldn’t crash the system

**Technology and data variation list**

2a) styles available list depend on furnishing supplier

2b) styles available list depend on room selected

4a) sidebar displays different furnishings depending on style selected

**Frequency of occurrence ( how often you expect this use case to occur)**

-> a couple times for each customer (person ordering the house) as once they decide on a style they usually stick to it

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case:** user wants to change selected room

**Primary Actor:** the user (person who is ordering a house)

**Stakeholders and interests:**

-> user wants to select another room to decorate the interior of

-> supporting: andrew’s exterior design module is providing the house blueprint

-> offstage: software dev wants to make sure user has a frictionless and pleasant experience

**Preconditions:**

-> house blueprint of a specific floor is displayed

**Success Guarantee (post condition)**

-> the clicked on room is now selected and displayed on screen

**Main success scenario (interaction of actors, system validation, system state change)**

1. User is on the interior design page of the process and house blueprint of a floor is displayed

Repeat 2 - 4 until desired room is selected

2. User clicks on a room displayed on the screen

3. Transition animation zooms in to display selected room in the center of the screen from a 2D to 3D view

4. Room is displayed in the center of the screen with adjacent rooms displayed on edges of selected room

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

1a) A specific room is already selected

1)user selects same room and nothing happens

2) user selects different room and display updates to show new room

2a) user clicks on already selected room and nothing happens

3a) transition animation fails (times out, use to much compute, etc) and system resets to state at step 1

**Special Requirements (non functional)**

Transition animation should be smooth and quick, no lag

Switching rooms should not cause everything to re-render to minimize computing resources and data usage

**Technology and data variation list**

2a) adjacent rooms will vary by selected room and floor

2b) with AR/ VR in few years users will not click but use gestures / mobile instead of using a desktop / laptop to design their house

3a) these transitions will eventually take place in AR / VR and will not need the 2D to 3D transition element

**Frequency of occurrence ( how often you expect this use case to occur)**

-> a couple times for each customer (person ordering the house) as houses usually only contain a handful of rooms and switching back and forth between rooms during the design phase is uncommon

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case:** user wants to get updated furnishing suggestions

**Primary Actor:** the user (person who is ordering a house)

**Stakeholders and interests:**

-> user wants to ideas of want to put in their rooms

-> supporting: external furnishing provider provides user with variety of relevant choices

-> offstage: software dev wants to make sure user has a frictionless and pleasant experience

**Preconditions:**

-> a room is currently selected and side bar displayed

**Success Guarantee (post condition)**

-> suggestions list is updated with new options

**Main success scenario (interaction of actors, system validation, system state change)**

1. User changes the state of the room

2. Room / Visuals update to reflect this

3. Loading icon is displayed on screen until sidebar finishes getting all information

4. System displays sidebar with styles selector, furnishing options, and at the bottom new suggested furnishings

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

1a) user adds, removes, or updates an item from a room

1b) user changes the style selector in the sidebar

1c) user choose a different room

4a) if sidebar fails to update reverts to previous state in step 1

**Special Requirements (non functional)**

-> options recommended should be relevant to the room and user preferences

-> neural net should make recommendations in real time

-> recommendations should be based off style select, items current in room, attributes of items currently in room, type of room selected (bedroom, kitchen, etc)

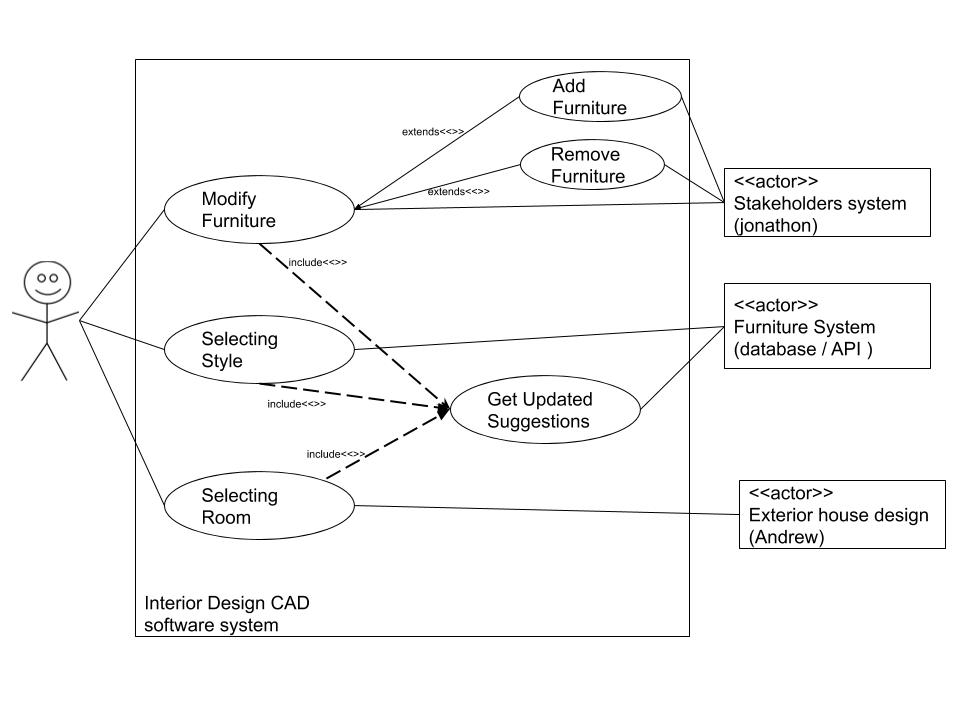
**Technology and data variation list**

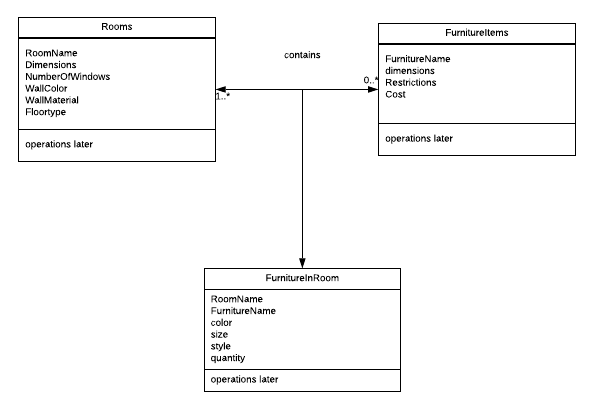
4a) furnishing options make come from external sources in JSON, BSON, XML

6a) neural network will need to be retrained every so often

**Frequency of occurrence ( how often you expect this use case to occur)**

-> hundreds of times for each customer (person ordering the house) as each action (changing rooms, adding furnishings, selecting style) will cause suggestions to need updating





Location Picker

**Use Case:** Software suggesting location near the user

**Primary Actor:** the user (person who is ordering a house)

**Stakeholders and interests:**

-> User wants suggestions of available locations near him/her

**Preconditions:**

-> Location picker functionality is selected

**Success Guarantee (post condition)**

-> User looks through potential locations near him/her and selects a suggestion to further evaluate

**Main success scenario (interaction of actors, system validation, system state change)**

1. User selects the location picker tab from our UI

2. Location picker loads on screen

3. Beneath the search bar there are suggestions of available locations near the user

4. User is able to select a suggested location with one click

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

3a) Location uses current location of user to find available locations within a 50 mile radius

3b) It is possible that there are no possible locations around the user and therefore the UI under the search bar would display “No available locations within 50 miles”

**Special Requirements (non functional)**

-> Each location should have a short description beneath it describing size of property

-> Location description should also contain what city and county the property is in

**Technology and data variation list**

3a) Our technology will search available areas in a certain radius

3b) If the location of user is not being shared with software the suggestions will simply be “locations near x” where x is our most searched location

**Frequency of occurrence ( how often you expect this use case to occur)**

-> Every time a user uses the location picker suggestions will appear. If the user picks a location the first time using the location picker than this will only occur once, but if not, this use case will occur multiple times.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case:** User searches a city/county in the location picker

**Primary Actor:** the user (person who is ordering a house)

**Stakeholders and interests:**

-> The user wants to find a property in a location of his/her choice

**Preconditions:**

-> Location picker functionality is selected and a city/county is entered into search bar

**Success Guarantee (post condition)**

-> Locations that are near searched city/county are displayed for user to click through

**Main success scenario (interaction of actors, system validation, system state change)**

1. User selects the location picker from the UI

2. Location picker loads on screen

3. User enters in preferred city/county into search bar

4. User enters a radius that is used to define exact zone in which the user is searching for

5. The location picker filters database to show locations within radius of the searched location

6. Properties around searched location are displayed for user to analyze

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

3a) If location does not contain any potential properties an error will occur on the screen

4a) if radius is too small the the user will be prompted to enter a larger radius

* To try and prevent this we will recommend radius’ in increments of 5 with the smallest radius being 5 miles

6a) If successful, the user will be able to look at the location of each available property and from their be able to click the property and check out specific details of the property

**Special Requirements (non functional)**

-> Search bar offers fill-in suggestions as the user types in city/county

-> After search takes place, the software tells the user how many properties fit his/her search

-> Searching is quick and efficient

**Technology and data variation list**

5a) Sorting method will be implemented to sort through database and find properties within searched area

6a) Locations will be displayed in a box like view where a picture of property is shown with location and dimensions right beneath

**Frequency of occurrence ( how often you expect this use case to occur)**

-> This use case will occur every time that the user is looking up properties in a specific location. This will most likely occur multiple times before the user finds the property that they are looking for.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case:** User selects a property to further evaluate property details

**Primary Actor:** the user (person who is ordering a house)

**Stakeholders and interests:**

-> The user wants to look at the details of a location to see if it fits his/her desires

**Preconditions:**

-> A location is selected either from suggestions or the search query

**Success Guarantee (post condition)**

-> User is able to see details about property such as dimensions, acreage, internet connectivity, surrounding schools, etc.

**Main success scenario (interaction of actors, system validation, system state change)**

1. User selects location picker from UI

2. User searches for a property or views the suggestions

3. User selects a property of interest

4. The location picker loads the details of a property

5. User is able to read through all of the information that our database has stored about a potential property

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

4a) Some properties may be new and therefore will not have a fully detailed report

* May be missing Internet connectivity
* May not have a price yet
* May not have a school system within reasonable range

5b) If successful, the user can compare the data of properties and use the information to determine exactly what kind of property he/she wants

**Special Requirements (non functional)**

-> Clean layout of all the details regarding a property

-> Outline on side of UI with links that will take the user down to exactly the section of the details he/she wants to see

**Technology and data variation list**

4a) All the details will be stored and updated in our location database

5a) Will have a user inquiry section that will allow users to request extra data about property

**Frequency of occurrence ( how often you expect this use case to occur)**

-> Everytime a location is selected by the user, details will be loaded about the property. The odds of the user only viewing one property before making a final decision is highly unlikely, so this use case will occur more than once.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Use Case:** User bookmarks a property

**Primary Actor:** the user (person who is ordering a house)

**Stakeholders and interests:**

-> The user sees a property that fits his/her interests so he/she wants to store property

**Preconditions:**

-> User finds a property that they are interested in

**Success Guarantee (post condition)**

-> User successfully adds property into a list of preferred properties

**Main success scenario (interaction of actors, system validation, system state change)**

1. User selects location picker from UI

2. User searches for a property or views suggestions

3. Users selects property of interest

4. The location picker loads details of property

5. If the details match exactly what user is looking for, he/she can add property to list of preferred properties, so that he/she can easily access this property later on

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

5a) If the details are not what user is looking for, there is no reason for the user to use this functionality

5b) If successful, it allows the user to have all of the properties that he/she likes in one place

* Allows for quick easy comparisons
* Makes it fast for users to further evaluate properties days after they first saw the property

**Special Requirements (non functional)**

-> List generated is clean and easy to sort through

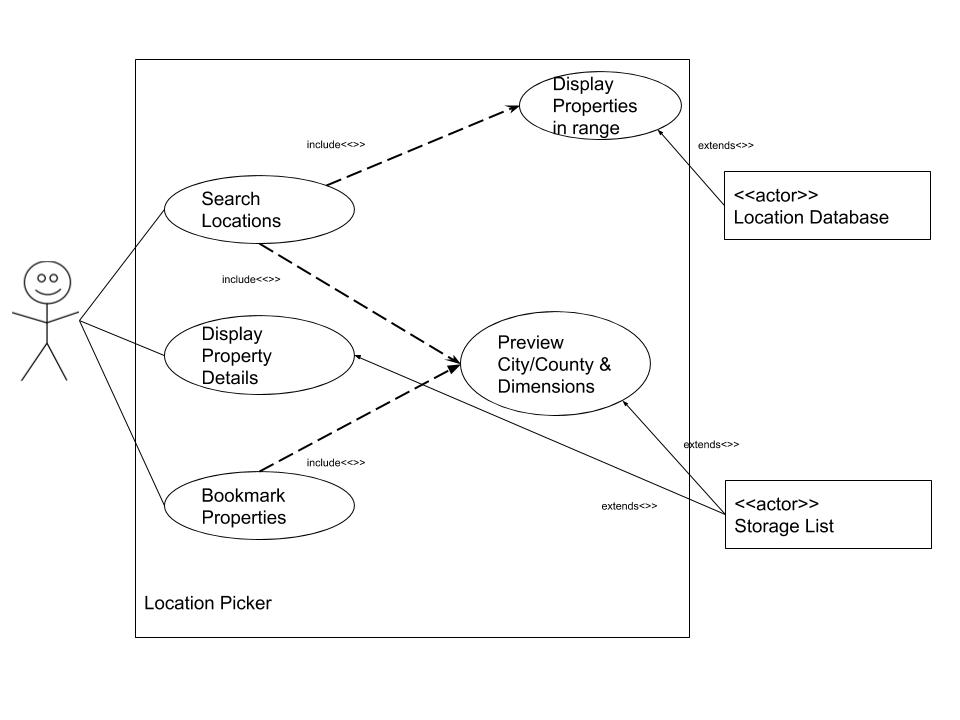
-> List has a search functionality that can be used if user has a lot of bookmarked properties

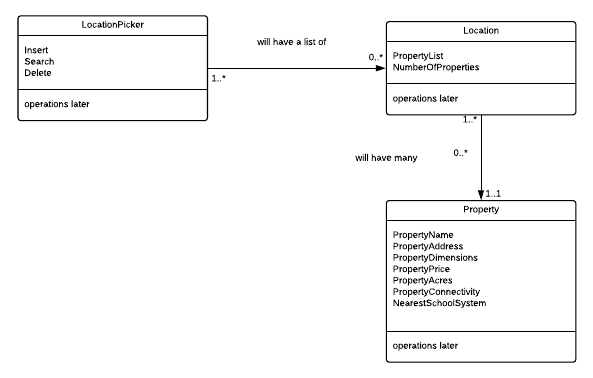
**Technology and data variation list**

5a) An array list of locations would be the best to implement this functionality because it is easy to insert and remove properties using this data structure

**Frequency of occurrence ( how often you expect this use case to occur)**

-> Everytime a user feels that a property is worth storing they should add it to the list. This use case should occur multiple times as it is likely that users will have a lot of potential properties in mind before they narrow it down to one final option.





Stakeholder Portal

**Use Case:** The client receives an update when a portion of their house is completed.

**Primary Actor:** The project manager

**Stakeholders and interests:**

->The client: wants periodic updates when some construction is completed

->Supporting actor: The primary construction contractor who has completed

**Preconditions:**

->A portion of construction has been completed on the client’s house.

**Success Guarantee (post condition)**

->The client receives the update and summary on the completed section of their home

**Main success scenario (interaction of actors, system validation, system state change)**

1. The program manager first logs into the website with their account information

2. PM selects the tab with the list of homes that they currently oversee

3. Then they select the house that currently had a portion completed

4. They choose the button selection on the same page to send a message to the client

5. A dialog box opens to send a message to the client

6. The manager types in a summary of the work completed on the component of the house into the text box in the dialog window

7. The manager attaches relevant images of the component to allow the client to see what was completed

8. Then they click the “send” button to send the message and image attachments to the clients account and get a confirmation message that it was sent.

9. Next time the client logs in to the website, they will get a notification in their messages with the summary and images of the completed component

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

**-**>9a. The message fails to send to the client and the user receives an error message

1. The user then proceeds to reload the page

2. The user then redo’s steps 6-8.

a. They do get a positive confirmation message

b. They get an error again and should repeat steps 1-8 again.

**Special Requirements (non functional)**

**->**The user should receive the message/update on their account within 3 seconds

**->**The system should be able to handle 1000 messages (send/receive) at any time.

**Technology and data variation list**

**->**1a. user passwords will have to be reset every three months

->8a. Messages will be sent using RSA/AES encryption formats to protect personal information being sent through messages

**Frequency of occurrence ( how often you expect this use case to occur)**

This should occur very often, as construction is completed on different components for different houses, each individual client should receive a update; there could be up to multiple occurrences of this case.

---------------------------------------------------------------------------------------------------------------------

**Use Case:** Checking supply of materials for a particular house

**Primary Actor:** build manager (from primary construction contractor)

**Stakeholders and interests:**

**->**Financial institution (offstage actor) needs to know what materials are necessary, in order to estimate costs of construction

->Exterior and Interior systems(supporting actor) provides information on what materials are needed

->Material supplier (supporting actor) is the one who keeps inventory of the materials required and available for that specific house

**Preconditions:**

**->**The house needs to have a client and a design already selected for materials to be visible and accessible.

**Success Guarantee (post condition)**

->The manager gains access the list of building materials for that specific house component

**Main success scenario (interaction of actors, system validation, system state change)**

1. The manager first logs into the website with their account information

2. Manager selects the tab with the list of homes that they currently oversee

3. They choose the house in particular from the list in which they wish to view inventory

4. They select the button for viewing blueprint design created by the client

5. Then the manager will select the specific component of the house they wish to view materials for

6. Then they click the “properties” tab on that component, select view supply inventory for that component to see the list of building materials for the component.

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

->4a. The manager chooses the house they wish to see building materials for, but the webpage does not load completely

1. The manager goes to click the dialog button for the list, but nothing comes

2. They refresh the page and let it completely load

3. The page reloads completely with full functionality

4. The manager clicks the button for the list and successfully opens it

->6a. When the user clicks on the button for the list, and the list is not there

1. The user refreshes the page in order to see if it was loading error

2. The page refreshes correctly, but does not display the list

3. The user needs to send a message to the supplier in order to have them post the list

**Special Requirements (non functional)**

->When accessed, the entire list no matter how extensive should load within 5 seconds

->When updated, the list should be globally updated within 10 seconds

**Technology and data variation list**

**->**API’s will be used to access the lists from the supplier; a API call will display the list that the supplier has posted.

**Frequency of occurrence ( how often you expect this use case to occur)**

This should be used frequently. If a construction foreman or project manager wants to check up on the supplies they should be able to access it 24/7 instantaneously.

---------------------------------------------------------------------------------------------------------------------

**Use Case:** Making a request for additional supplies

**Primary Actor:**

->The construction foreman from the construction team

**Stakeholders and interests:**

->Material supplier (supporting actor) provides the raw materials to construct the house

->Exterior and Interior systems(supporting actor) provides information on what materials are needed

->Financial institution (offstage actor) if more materials are needed/cut then total costs will change

**Preconditions:**

->The house that the request is for must currently be in the process of being built or designed

**Success Guarantee (post condition)**

**->**The construction manager sends off an updated request for more materials to build the house

**Main success scenario (interaction of actors, system validation, system state change)**

1. The construction foreman logs into the website

2. The foreman selects the house that they are in charge of building.

3. The foreman then accesses the entire materials list for all components of the house by clicking on it

4. The foreman clicks the “Edit” button next to the materials list to access and alter the list of building supplies

5. A dialog box opens with the current list

6. The foreman enters and updates the supply values on the list in the dialog box

7. When finished, they click the update request at the bottom of the dialog box which then closes upon completion

8. When the supplier logs back into their account, they will have a message waiting in their inbox with the updated request.

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

**->**7a. The user goes to click the update request on the dialog box, but the operation fails.

1. The user refreshes the web page to see if there is a problem

2. The user then resubmits the update form again successfully

**Special Requirements (non functional)**

**->**Request should be sent and verified within 5 seconds

->If request fails to be sent, the user should be notified within 10 seconds

**Technology and data variation list**

**->**7a. API will be used to communicate, i.e. API call that will send the request to the supplier’s account.

->5a. Browser dialog boxes will be used to view or enter information

**Frequency of occurrence ( how often you expect this use case to occur)**

This should occur infrequently, as after the client designs the blueprint for the house, the materials should be decided. However, if the client changes their mind about the particular design for something, then the supply manifest for that component(s) would have to altered.

---------------------------------------------------------------------------------------------------------------------

**Use Case:** accessing a client’s financial records

**Primary Actor:** The financial institution

**Stakeholders and interests:**

->Construction contractor (offstage actor) needs to know whether the client can pay for construction

->Material/equipment supplier (offstage actor) should know whether the client has the ability to finance materials and equipment used to create the housing project

**Preconditions:**

-> The client is looking to purchase/create a house with the 3D printing system and has posted documents in regards to their financial history

**Success Guarantee (post condition)**

**->** The financial institution successfully accesses the client’s posted fiscal history

**Main success scenario (interaction of actors, system validation, system state change)**

1. The financier logins into their account on the website with their password and username

2. The financier chooses a tab with a list of clients, and selects the client in particular

3. After choosing the client, their profile will appear with all their information

4. The financier clicks on ‘view documents’ for that client and chooses ‘financial records’

5. A dialog box will open and ask the financier to put in their official credentials of their financial institution to confirm their identity and purpose.

6. The financier successfully logs in and views the client’s documents

**Extensions (conditional and branching statements … both success and failure … what happens if a step in main success scenario fails)**

**->**5a. The financier fails to authenticate their identity by accident

1. The client’s account and personal information is locked down

2. The client receives a notification on their account and email about unauthorized attempts to access their account.

3. The client can then re-login to their account to remove the lockdown, and let the financier know to retry

4. The financier restarts from step 1 until success access

**Special Requirements (non functional)**

->Verification of identity and official credentials of the financier should occur within 10 seconds

**->** Privacy of information should be monitored and audited when personal information is accessed

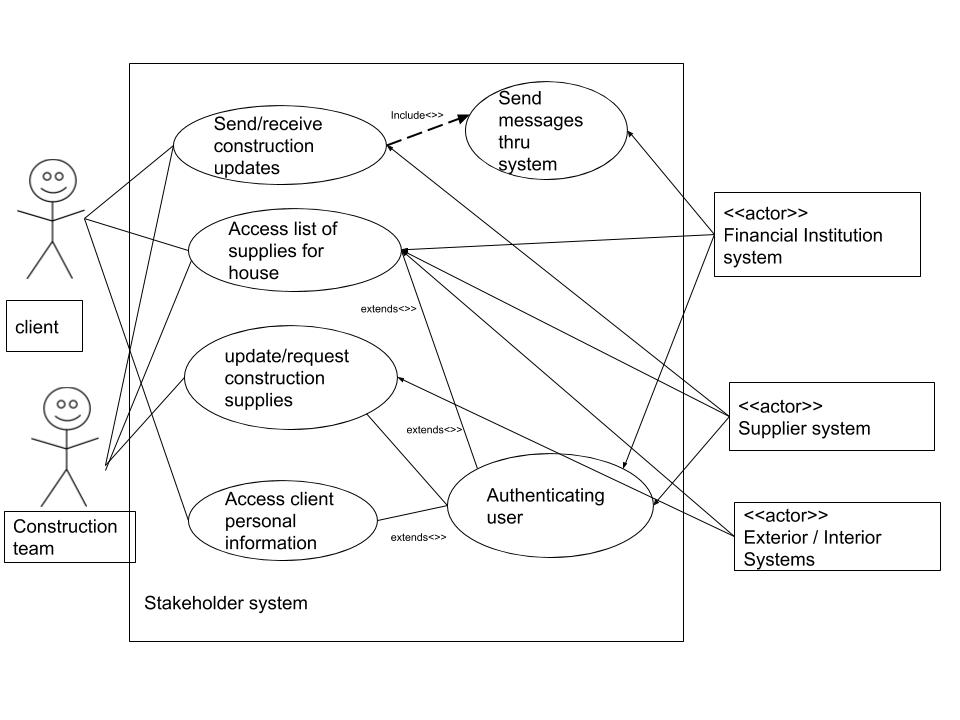
**Technology and data variation list**

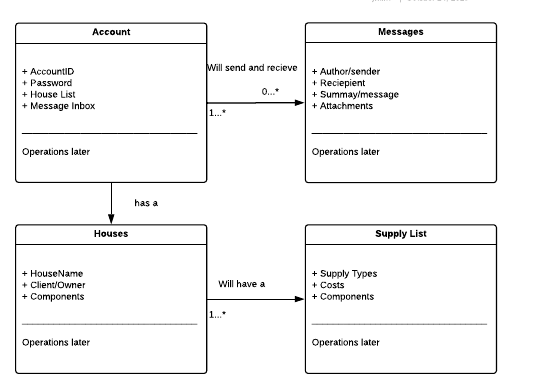
**->** 5a. Thefinancial institution’s database server will process the authentication request and return the result via an API call from the user’s website.

-> 6a. Account lockout tool will be used if the user fails to authenticate their identity 3 times incorrectly.

**Frequency of occurrence ( how often you expect this use case to occur)**

This many occur frequently, as every time a brand new client wants to create a new house, the financial advisor from the institution would need to check the fiscal background of that client. Even previous clients would still need to provide





Updated Functional Requirements

3d printing a house

1&2 broke CAD software into house into two parts

epic 1) Andrew -> house design (outside [ walls, windows, doors, roof] )

-> allow users to drag walls up, resize, add roof, cut holes for windows etc.

-> connect rooms by dragging boxes together

-> should pull in material choices (from API) and display them to user. Ex. user would select brick for outside wall and then drag up like in CAD to build it

-> should check compatibility of select options (ex. Slate roof requires steel beams, wood won’t support the weight)

-> offer suggestions based on location (ex. sunny area would suggest

solar panels, colder area might suggest thicker walls with insulation)

epic 2) Connor -> house design (inside)

->should pull in design options (cabinets, floors, windows etc) from

external API and display relevant ones to user. kitchen Cabinets for bedroom makes no sense

-> offer default style [ style defined by choice of furniture, cabinets, etc] choices (modern, french, old school, goth)

-> drag and drop furniture , select color, material of couch

-> offer suggests based on room and previous choices

epic 3) Matt -> location picker -> offers available locations for construction to users

-> display available locations from DB in nice front end UI

-> continuously update available locations from external API

-> implement a search algorithm to filter locationscons

-> have information about each location (power lines, sewer available,

road access, internet cable access etc.)

epic 4) Jonathan -> part of website, let's say a webpage for financing, material suppliers,

customers to interact with … each user will want different info, this will use an API, but you need to make 4 functional requirements

-> 1. Client should be able to receive updates about the status of their house

-> 2. Primary construction contractor should be able to access and see list of available materials for every house being built.

-> 3. Construction team should be able to make requests to material/equipment suppliers

-> 4. Financial institutions should be able to access potential clients’ fiscal history.