Buildr

CS30700: Design Document

Team 21:
Jialu Gu
Shafer Hess
Emily Ou
Derek Shu
Mangkorn Yuan

Index

- **Purpose** 2
  - Functional Requirements
  - Non-functional Requirements
- **Design Outline** 8
  - High Level Overview
  - System Components
  - Basic Sequence Overview
- **Design Issues** 11
  - Functional Issues
  - Non-functional Issues
- **Design Details** 15
  - Classes Diagram
  - Classes and their Interactions
  - State Diagrams
  - UI Mockups
  - Sequence Diagrams

Purpose

Students are always looking for new ways to interact with their campus and each other, but a lackluster selection of unique social campus applications severely limit the options for students. The purpose of Buildr is to provide students on a university campus with a new way to interact with their campus and each other.

The primary function of Buildr is an Augmented Reality (AR) modeling and sharing feature. Users will be able to design 3D models that can be placed in the world through the use of AR. A user will be able to share their creations with other users of the app.

The secondary function of Buildr is an look into the history of a user's campus using AR. User's will be able to learn about the history of some of the most prominent buildings on their campus. Pointing the camera at predesignated signs on campus will reveal models and statistics related to that building in augmented reality.

Through the combination of these two features, we will provide university students with a new way to build relationships with their campus and the people who interact with it on a daily basis.

Functional Requirements

User Accounts

As a user,

- "I would like to be able to register for my Buildr account so that I can participate"
- "I would like to be able to login to my Buildr account so that I can interact with the app and other users."
- "I would like to be able to reset my password if I forget it so that I can use the app without making a new account."
- "I would like to be able to change my associated email in case I want to create a new email account"
- "I would like to be able to change my password in case I need to change passwords for security reasons"

- “I would like to be able to have 2FA protection in case my password was compromised” (if time allows)

Account Settings

As a user,

- “I would like to be able to customize my avatar so that I can give my account some character”
- “I would like to be able to customize the notifications from the app so that I can choose only specific types of events to notify me about”
- “I would like to be able to change my display name so that it gives my account more character/personality”

User Interface/User Experience

As a user,

- “I would like to be able to have an easy-to-use UI on the Buildr app so that I can use the app without much confusion.”
- “I would like to be able to change the language of the app so that I can view descriptions of buildings in a language I understand better” (if time allows)
- “I would like to have access to a tutorial so that I can learn my way around the app when I first download it”
- “I would like to be able to view the change log for updates to the app so I can see what features have been added or removed for each build”
- “I would like to be able to have full control and transparency over my privacy settings and data collection so I know exactly what data is being collected and what data I’m providing”
- “I would like to be able to choose if the app can update over mobile data, wifi, automatically, or manually so I have control over when or if the app updates”

Single User Interactivity

As a user,

- “I would like to be able to view the history of buildings on campus.”
- “I would like to build creations like Lego blocks using 3D models provided by the app.”

- “I would like to be able to rotate and scale the 3D models provided by the app.”
- “I would like to be able to undo and redo when building my creation.”
- “I would like to be able to be able to see my creations in an AR environment”
- “I would like to save and load my creations on the local disk”
- “I would like to be able to search for specific locations inside of the app in case I really want to go see a specific point of interest”
- “I would like to be able to access my AR creations offline so that I can modify my work without an internet connection”
- “I would like to have preset guide routes I can follow in case I don’t really know good locations to visit”
- “I would like to have directional information on screen so I know where to go when navigating to a point of interest”

Multi User Interactivity

As a user,

- “I would like my modifications and modifications from other users to show up in the real-time public AR world” (if time allows)
- “I would like to be able to communicate with other users on the application”
- “I would like to be able to upvote/like creations from other people, so that I can show that I support their work”
- “I would like to be able to friend other users on the platform so that I can follow their creations” (if time allows)
- “I would like to be able to remove other users from friends in the app so that I can delete friends that I no longer wish to be friends with”
- “I would like to be able to see other users currently nearby and viewing the AR world so I feel like I’m truly interacting with people inside another world”
- “I would like to be able to enable or disable being publically visible in the AR world in case I want to hide from others in the AR world”

Moderation and Administration

As an administrator,

- “I would like to be able to censor inappropriate words in the comments and replace them with stars”
- “I would like to be able to delete other user creations in the application (if inappropriate)”
- “I would like to be able to modify text descriptions of the buildings inside the app to either add, edit, or delete information”
- “I would like to be able to ban other users from the platform to get rid of troublemaking users”
- “I would like to be able to timeout other users from the platform to give a warning to troublemaking users”

As a user,

- “I would like to be able to report inappropriate content made by other users so that the content will be reviewed and possibly removed”

Non-functional Requirements

Performance Requirements

As a user,

- “I would like the application to be battery friendly on my phone so I don’t have to worry about draining too much battery while using the app”
- “I would like to be able to login quickly so I’m not waiting excessively long on the login screen”
- “I would like to be able to load the AR world quickly”

Security Requirements/Other

As a user,

- “I would like my password to be stored in a secure manner so that should the data be compromised, it isn’t just plain password text handed over to the hackers”

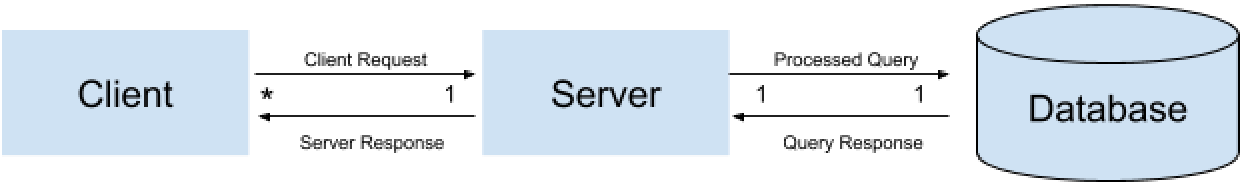
As a developer,

- “As a developer, I would like to use a database hosted in the cloud services so that usability is intuitive and simple”
- “As a developer, I would like to have a server architecture that allows us to easily scale the amount of users active in our app so that we can include as many users as possible.”
- “As a developer, I would like to have the ability to easily include users from other universities.”

Design Outline

High Level Overview

Buildr will be a smartphone application available on both iOS and Android app stores. Users can create and share 3D modeled creations as well as receive information about their campus. To accomplish this, Buildr will implement a client-server model where users communicate with the server to download, upload, and share their creations. The client-server model will be implemented with a Node.js server and a SQL database in the cloud.



System Components

1. Client

a. The Client is the main point of interfacing with the user.
b. Client sends requests to server based on the type of interaction.
c. The response from the server updates the view of the client appropriately. (Login Success/Failure, Upload Success/Failure, etc.).

2. Server

a. Server receives and processes user requests from the client.
b. When data needs to be uploaded or downloaded, the server will send the appropriate requests to the database.
c. Depending on the request received and the response from the database, the server will send the appropriate signals back to the client.

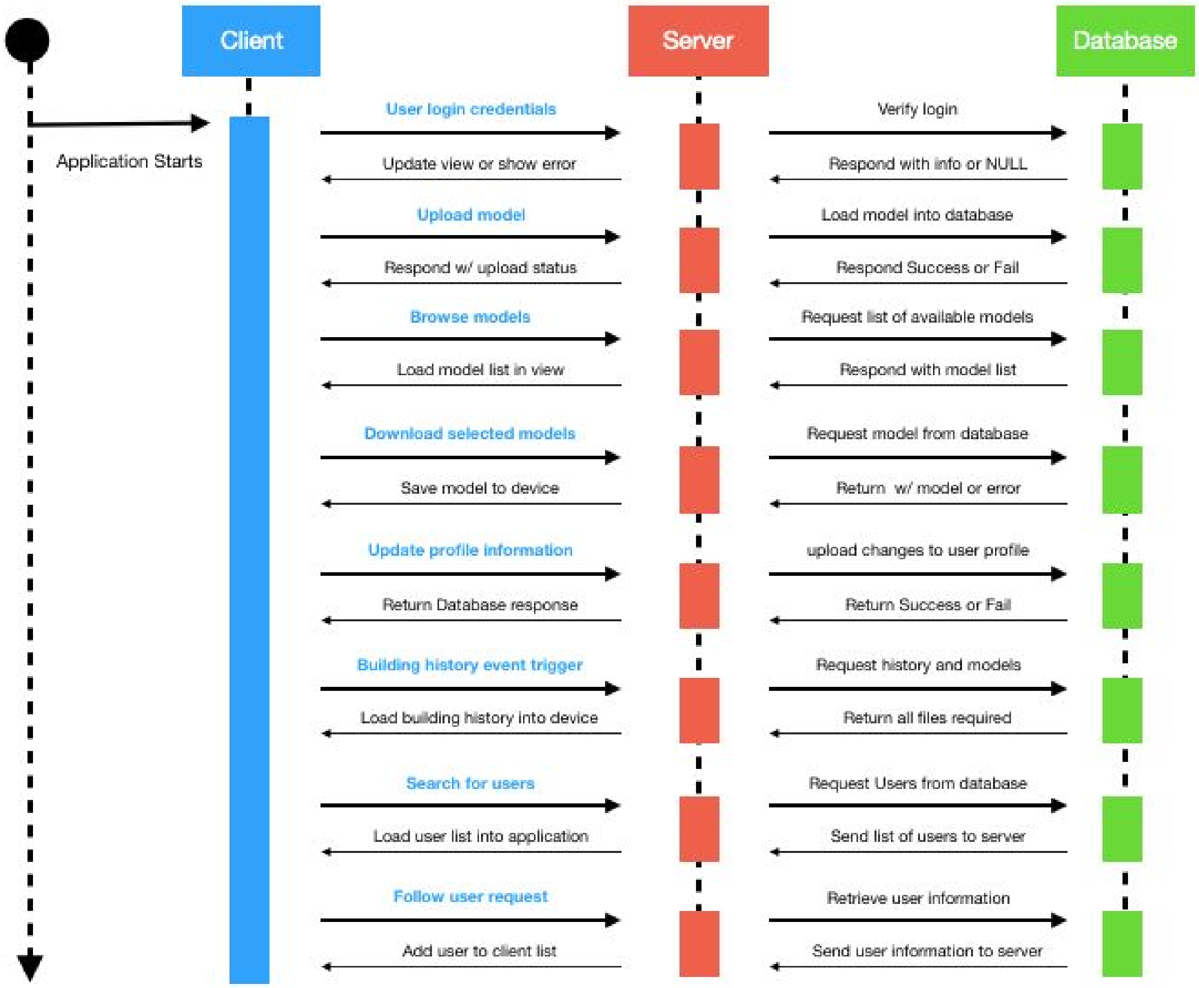
3. Database

a. The Database will receive and process signals from the server.
b. Database will store user information, content that must be downloaded for the client, and all models that are accessible by multiple users.
c. When data is retrieved from the database, it is sent back to the server which then sends the final data back to the client and user.

Basic Sequence of Events

Sequence Overview

The following diagram illustrates a simple sequence overview that a user will encounter when they launch the Buildr app. Users begin by logging into the app. Login is handled by sending the entered credentials to the server which then validates the credentials through our database. Once a user has logged into their account, many other sequences are opened up. Upon login users can upload and download models, search for other users, update profile information and follow other users. All of the AR building is handled within the client application and the only interaction that it has with the server and database is the addition of models to our database.


Client
Server
Database
Application Starts
User login credentials
Update view or show error
Verify login
Respond with info or NULL
Upload model
Respond w/ upload status
Load model into database
Respond Success or Fail
Browse models
Load model list in view
Request list of available models
Respond with model list
Download selected models
Save model to device
Request model from database
Return w/ model or error
Update profile information
Return Database response
upload changes to user profile
Return Success or Fail
Building history event trigger
Load building history into device
Request history and models
Return all files required
Search for users
Load user list into application
Request Users from database
Send list of users to server
Follow user request
Add user to client list
Retrieve user information
Send user information to server

Design Issues

Functional Issues

1. **Should we require users to provide a university email?**
    - **Option 1:** University email is required to login to Buildr
    - **Option 2:** Any email will be accepted by Buildr
    - **Option 3:** Users do not need any email to login into Buildr

    Final Choice: OPTION 2

    Justification: By limiting a user account to just a university email, we severely limit the audience that we can reach through our app. Opening the app to all email accounts will allow visitors to be a part of our experience whether they attend university or not, greatly increasing the size of our potential audience.

2. **Should users have to allow other users to follow them?**
    - **Option 1:** Anyone can follow the creations of other users
    - **Option 2:** Only allow people a user accepts to follow their creations
    - **Option 3:** There will be no follow system / Users will post anonymously

    Final Choice: OPTION 1

    Justification: We want Buildr to be a collaborative app where users interact with people they may not normally interact with. If you are interested in another user's creations then we believe that restricting your ability to follow them removes part of the intractability we want in our app.

3. **How the users build their 3D models on the client?**
    - **Option 1:** Use models with dynamic mesh for users to modify
    - **Option 2:** Use pre-made static models for users to build them like building Legos
    - **Option 3:** Allow users to import their 3D models from others softwares such as Maya.

    Final Choice: OPTION 2

    Justification: Using models with dynamic mesh harms the performance of the software as they usually have too much trigs to render. Allowing users to import 3D models from other software is also not a good option because Unity may not be able to recognize their format. Using pre-made models for users to assemble is the best choice because it is easy for users to operate and saves the performance via gpu instancing.

4. **How will we deal with there being too many objects placed and/or rendered?**
   - **Option 1:** Limit users to a certain quota of objects they can place.
   - **Option 2:** Render random, but an appropriate amount of, objects for users to see for their specific instance.
   - **Option 3:** Render popular creations by default, but let the user manually render other creations if they want.

   Final Choice: OPTION 3

   Justification: Having a public sandbox like this would eventually cause the entire grid to be filled with objects. This would severely hinder performance and also make it impossible to decipher what anyone actually built. That's why we will render popular (most liked) creations by default and then allow the user to toggle on/off other creations they would like to view. Rendering random objects would be too inconsistent for the community as a whole, and limiting users to a certain quota would ruin the fun behind the application.

Non-functional Issues

1. **Which cloud service should we use to host our database?**
   - **Option 1: Microsoft Azure**
   - **Option 2: Amazon Web Services**
   - **Option 3: Google Cloud Services**

   Final Choice: OPTION 2

   Justification: Amazon Web Services provides more detailed documentation about how to set up servers and databases through AWS. Furthermore, pricing for Amazon Web Services seems to be more appropriate for our purposes.

2. **Which database should we use?**
   - **Option 1: Microsoft SQL**
   - **Option 2: MySQL**
   - **Option 3: OracleDB**

   Final Choice: OPTION 2

   Justification: Because MySQL is an open-source database solution, it has a larger use community compared with other commercial databases. Therefore, there are much more documentation and tutorials about MySQL. Since we don't have much experience with databases as a group, MySQL is the better choice for us to learn and implement.

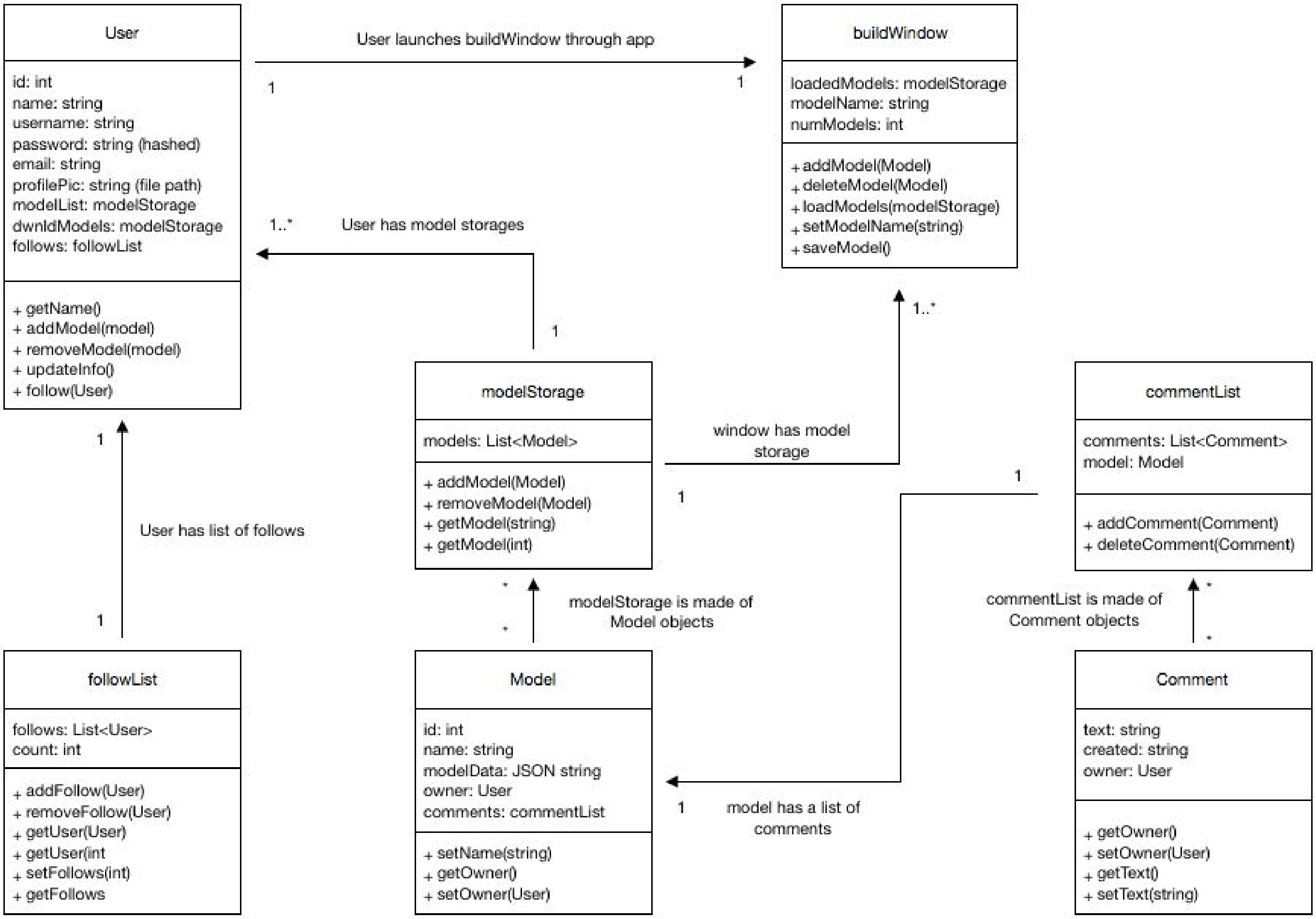
3. **Which development platform should we use for the client developing?**
   - **Option 1: Android Studio and ARCore SDK**
   - **Option 2: Vuforia**
   - **Option 3: Unity Game Engine and its AR asset**

   Final Choice: OPTION 3

   Justification: Unity not only provides AR solutions but also provides a complete API for game logic, UI control, and web communication. It also has the largest user community among the three options. Thus it has more detailed documents. Because its functionality and detailed documents, we consider Unity as the best choice.

Design Details

Classes Diagram



Classes and their Interactions

User

About

- A User object will exist for every person who signs up for the app.
- The User class will contain basic account information relating to the user.
- Users will be assigned an ID number for use in methods.
- Getters and Setters are available to update and retrieve information regarding a user.

Interactions

- The User class interacts with the following classes: buildWindow, modelStorage, followList
- The User launches the buildWindow class when they want to construct a new model in the app. The buildWindow class can access the users saved models in the modelStorage object in the User class
- Each User class will have multiple modelStorage class objects to store downloaded and created models on the user's local device.
- A User will have a followList that stores the Users that they want to follow.

buildWindow

About

- The buildWindow class is invoked when a user starts to create a new model in the app.
- It allows the developers to interface with the user's saved and created models.

Interactions

- The buildWindow class interacts primarily with the User class to load the appropriate models and information.
- buildWindow also interacts with the modelStorage class so that it can store models within the class for easy updating and removal.

Model

About

- The Model class is the main class that stores information about user and developer created models.
- Models contain information regarding the design, color, orientation, etc in a JSON file to be read by Unity.

Interactions

- The Model class interacts with the modelStorage and commentList classes.
- Models are stored in the modelStorage class to be used elsewhere in the application.
- Each Model class will have a commentList class that associates user comments with a specific model.

modelStorage

About

- The modelStorage class acts as a storage medium for Models to be used by other classes in the application.
- This class has methods to add or remove models based on user input.

Interactions

- The modelStorage class interacts with the following classes: Model, buildWindow, User.
- Each User class object will have modelStorage classes to store Models.
- The buildWindow class contains modelStorage class objects to interface with the User models.
- modelStorage classes are comprised of Model class objects.

Comment

About

- Comment is the object that allow users to leave messages on other user's models.
- Comments are stored in a text variable accessible by getters.
- The poster of the comment is stored in the object .

Interaction

- The Comment class interacts primarily with the commentList class and only exists within this class.

commentList

About

- The commentList class holds all the comments that are associated with a Model.
- There are getters and setters for viewing all comments that go with a model.

Interaction

- commentList class objects are stored within the Model class.

followList

About

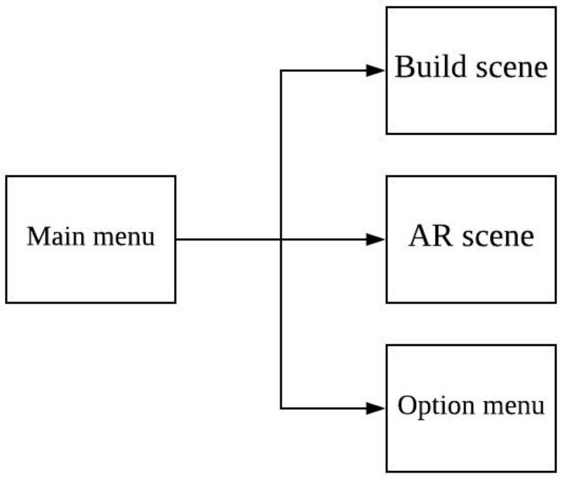
- The followList is the list of all the other users the user decides to follow.
- This class has methods to add and remove other user and the amount of follows a user has.

Interaction

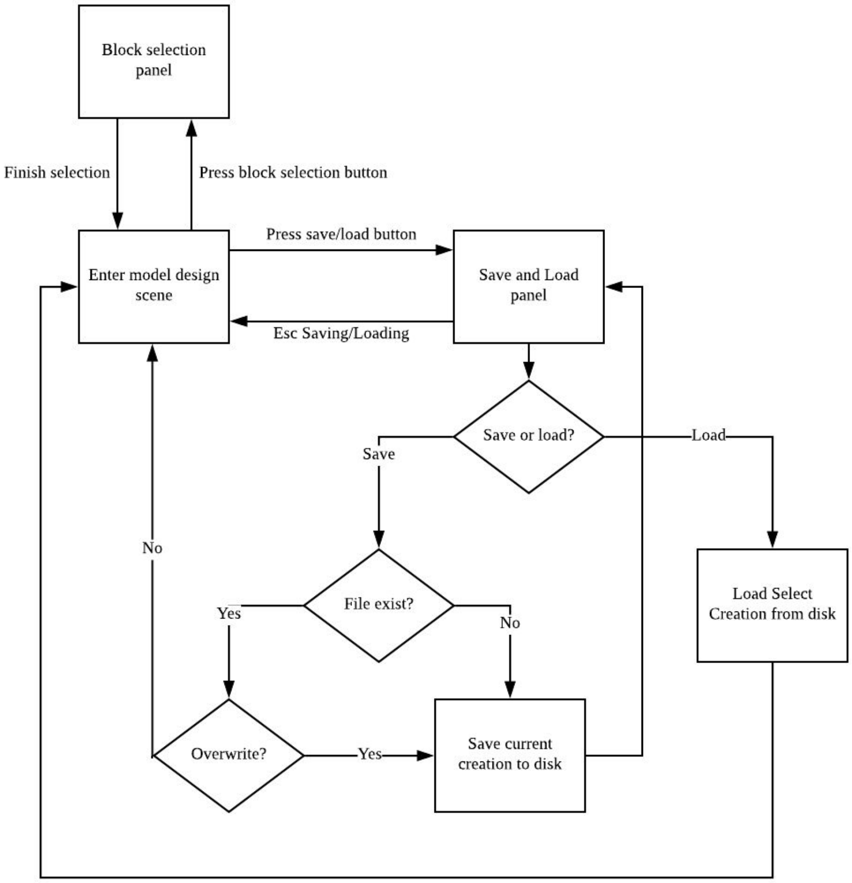
- The followList interacts with the User as it creates an array of all the users the user would like to follow.

State Diagrams

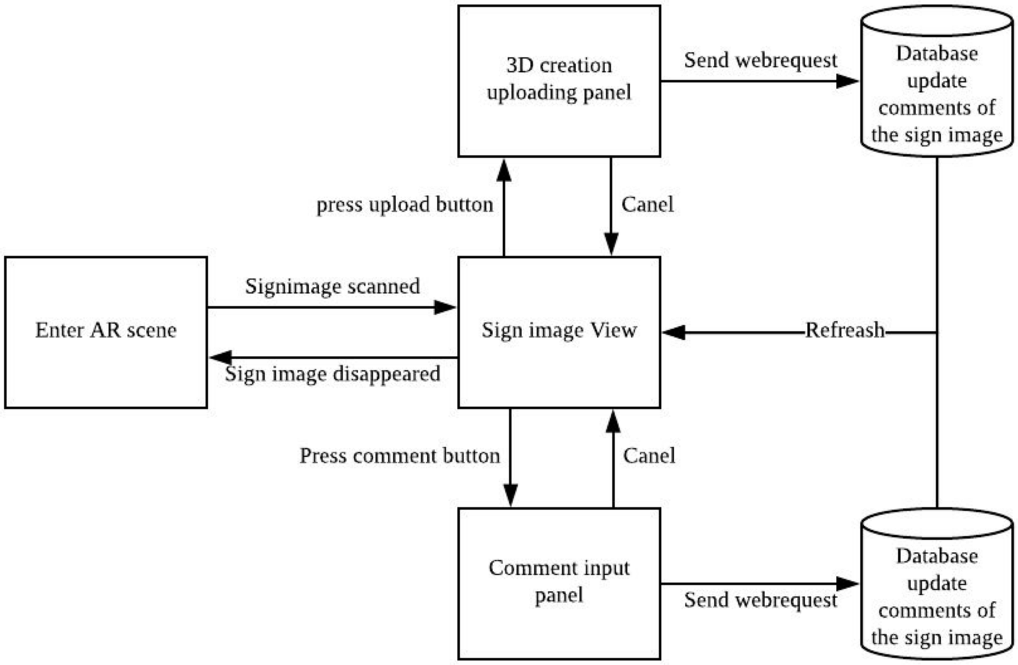
Main Scene State Diagram



Build Scene State Diagram

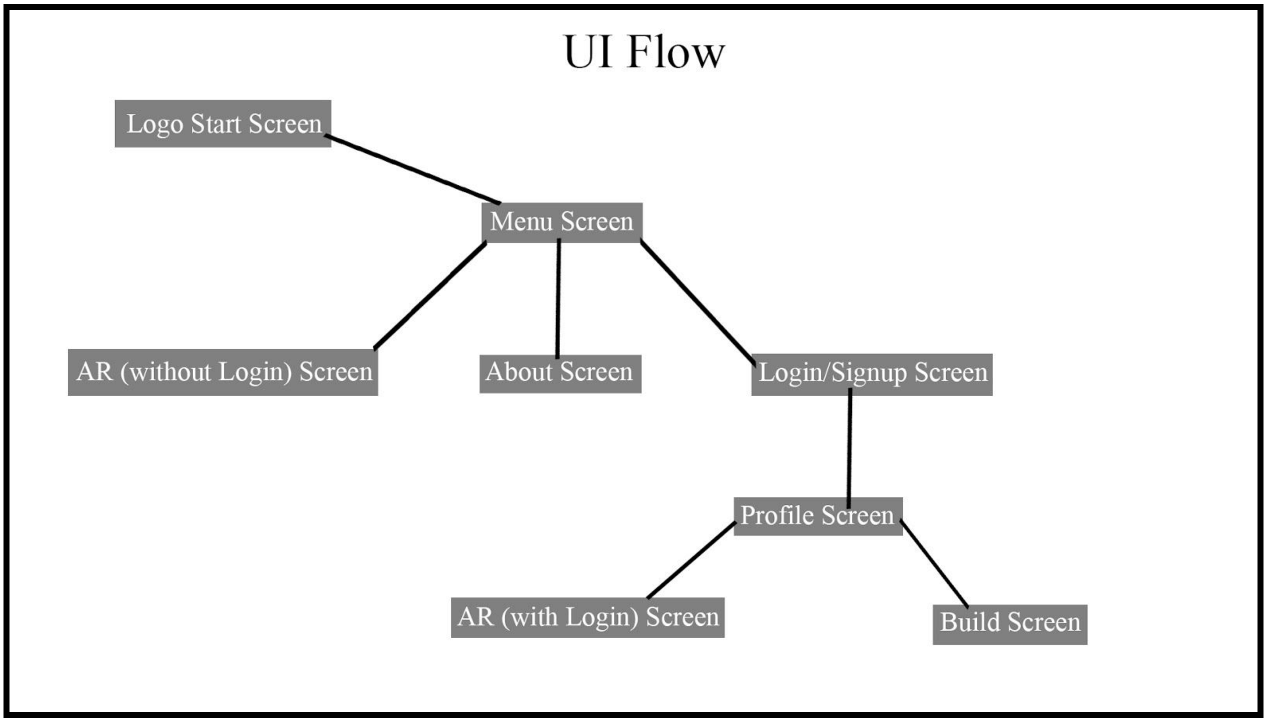


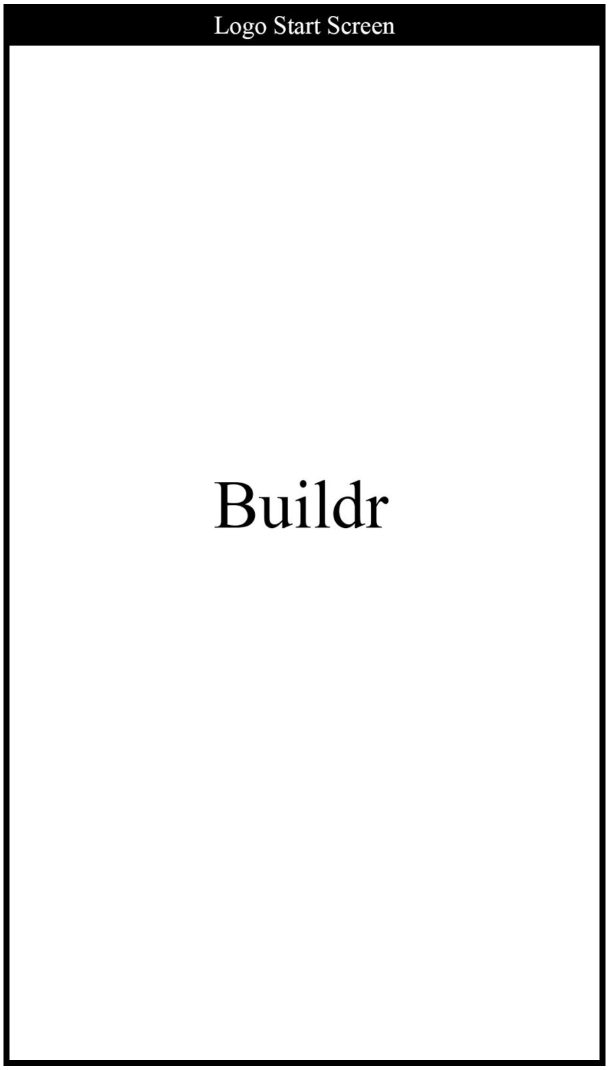
AR Scene State Diagram

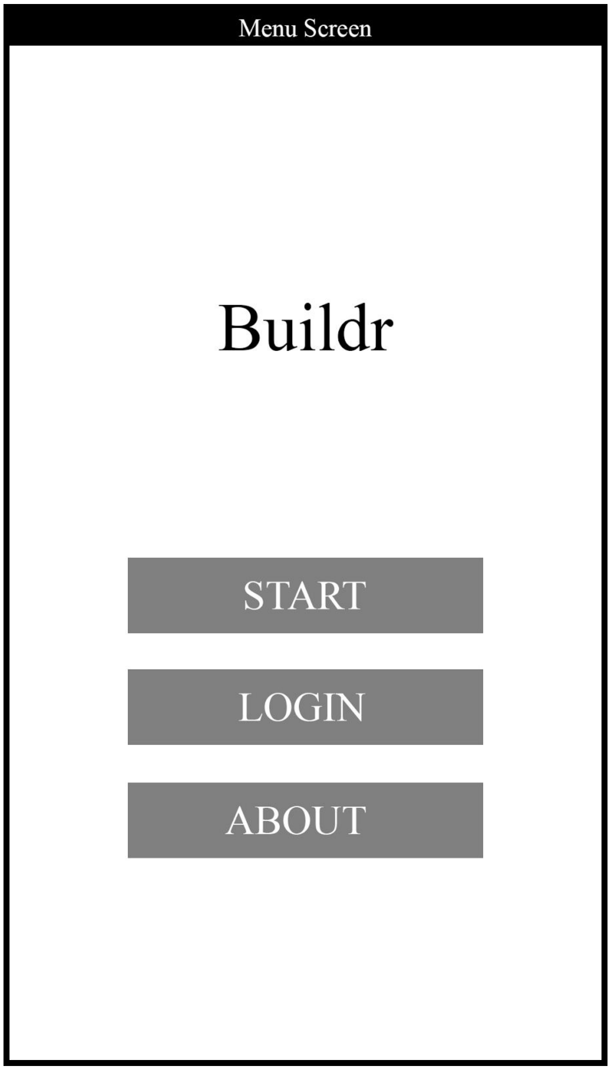


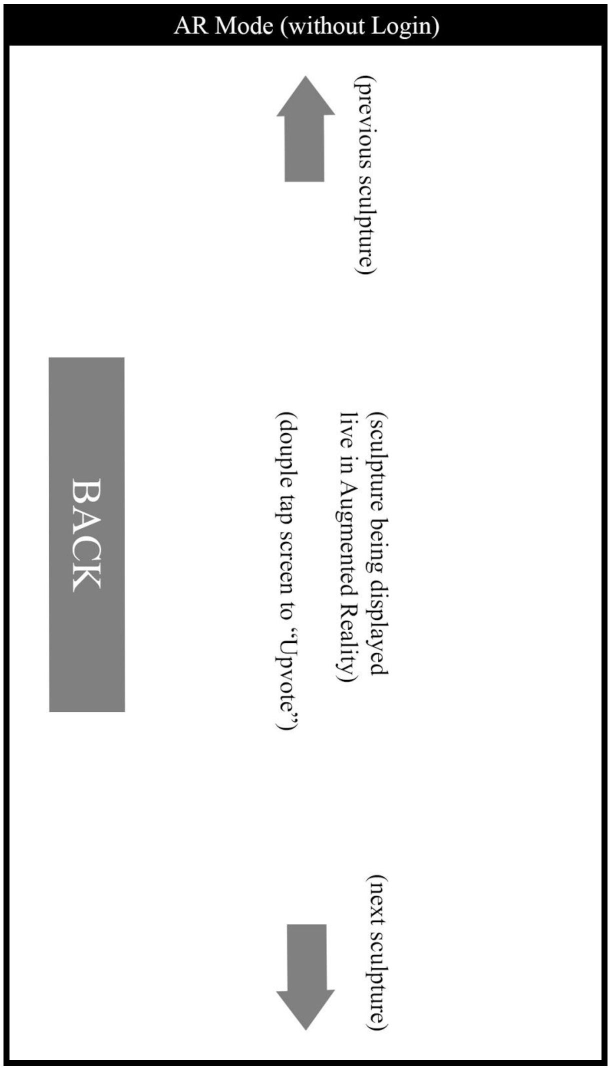
UI Mockup

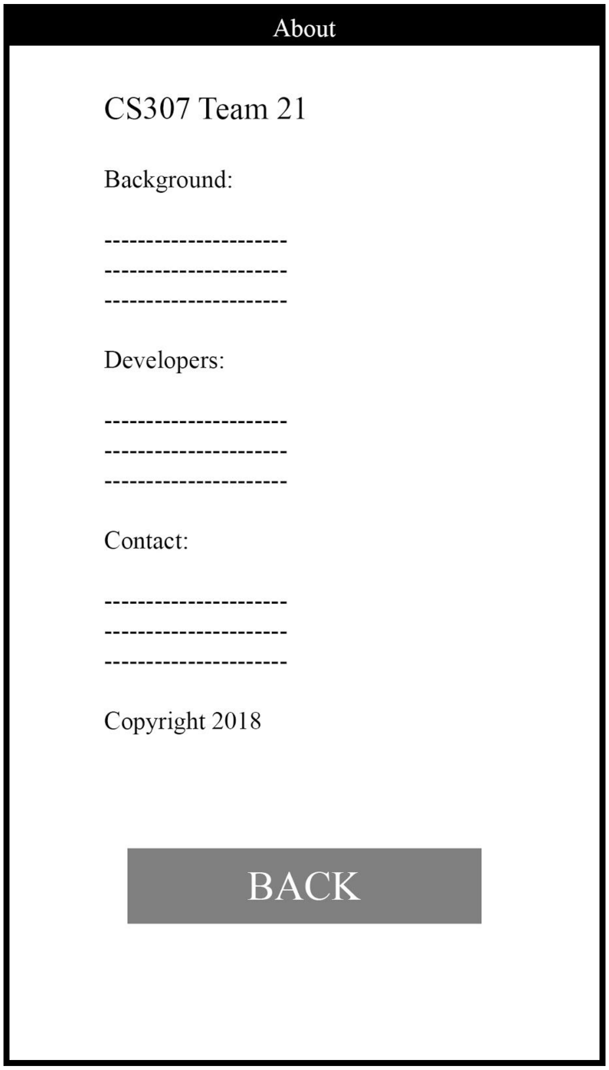
Below is the UI design flow with additional details as to what each UI Panel should contain (buttons, images, etc.). Users start off at the “Logo Start Screen”.




Logo Start Screen
Buildr


Menu Screen
Buildr
START
LOGIN
ABOUT

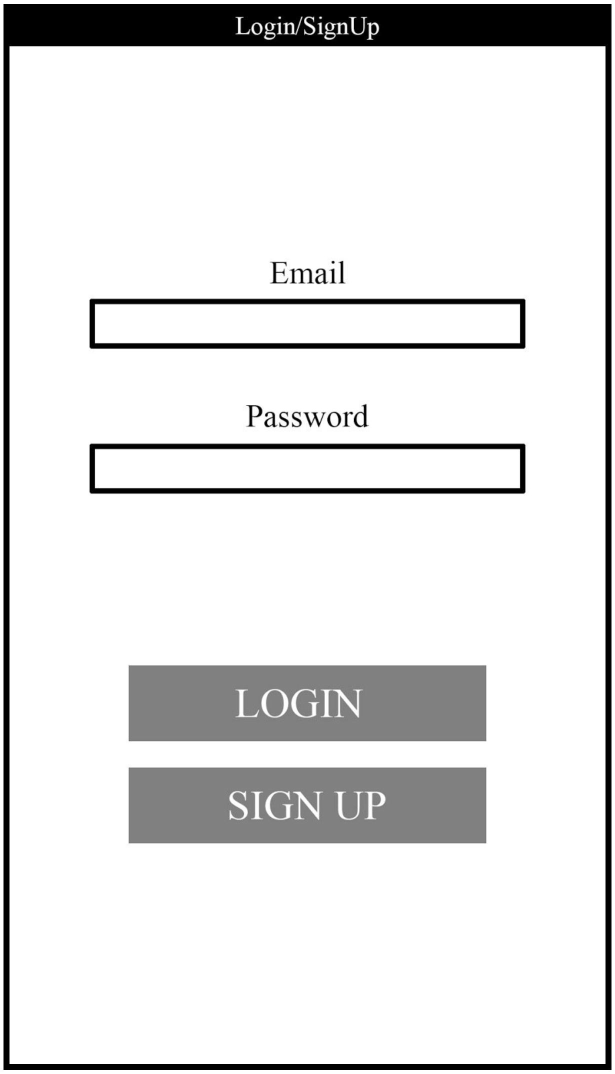

AR Mode (without Login)
(previous sculpture)
BACK
(sculpture being displayed live in Augmented Reality)
(douple tap screen to “Upvote”)
(next sculpture)

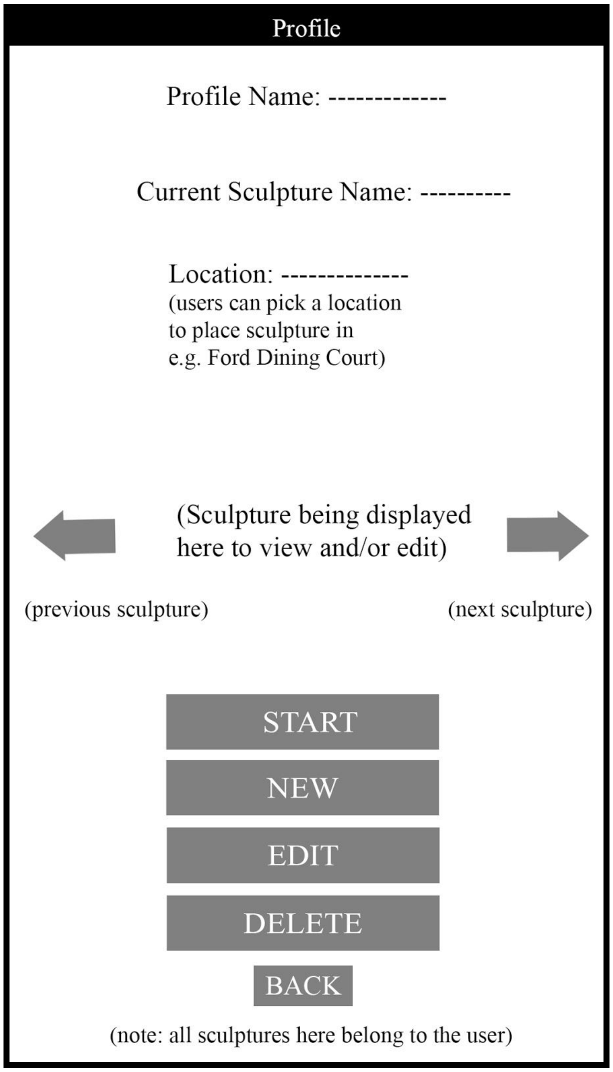

About
CS307 Team 21
Background:

Developers:

Contact:

Copyright 2018
BACK




Profile
Profile Name: -------------
Current Sculpture Name: ----------
Location: --------------

(users can pick a location
to place sculpture in
e.g. Ford Dining Court)

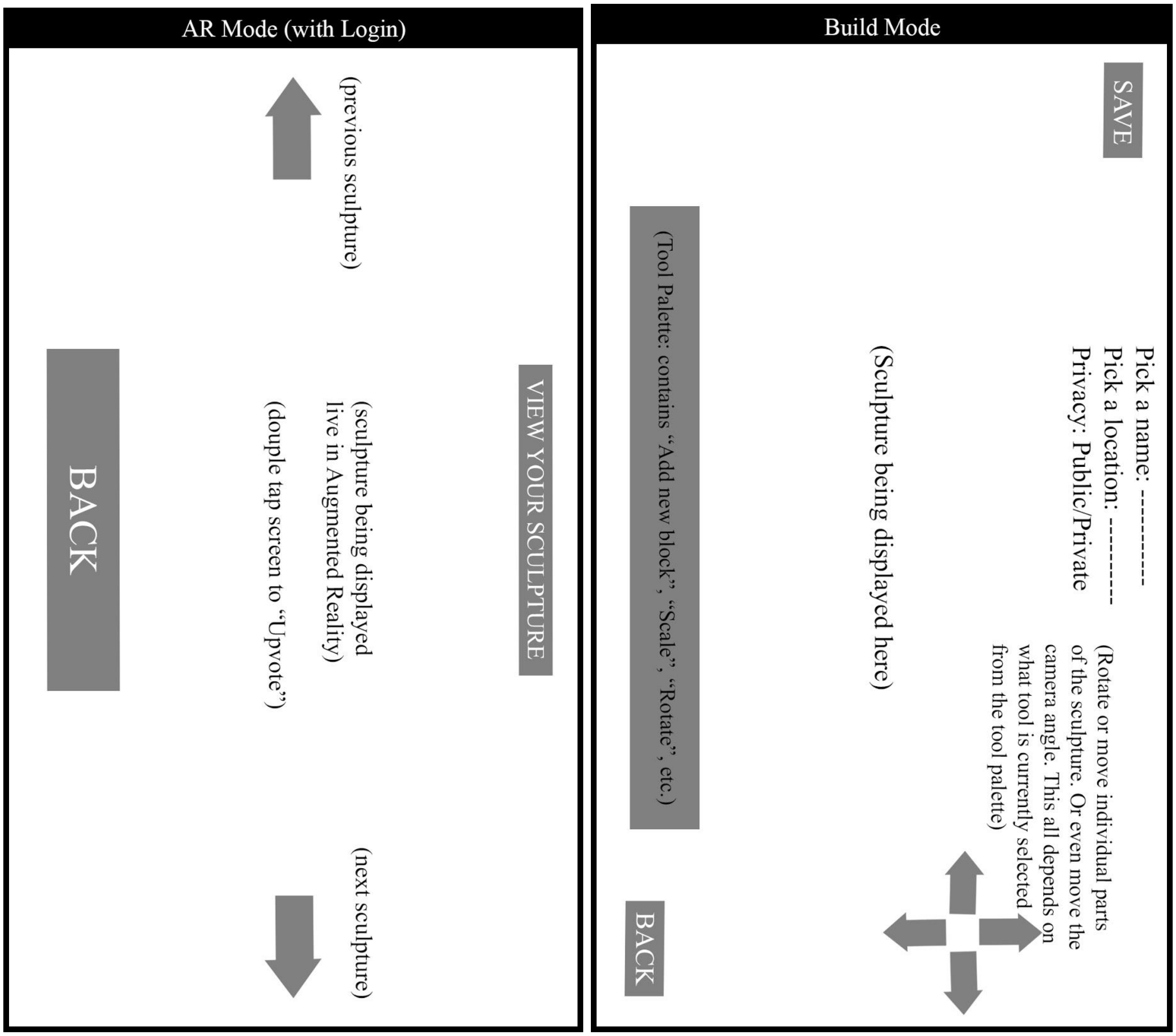
(Sculpture being displayed here to view and/or edit)

(previous sculpture)

(next sculpture)

START
NEW
EDIT
DELETE
BACK

(note: all sculptures here belong to the user)

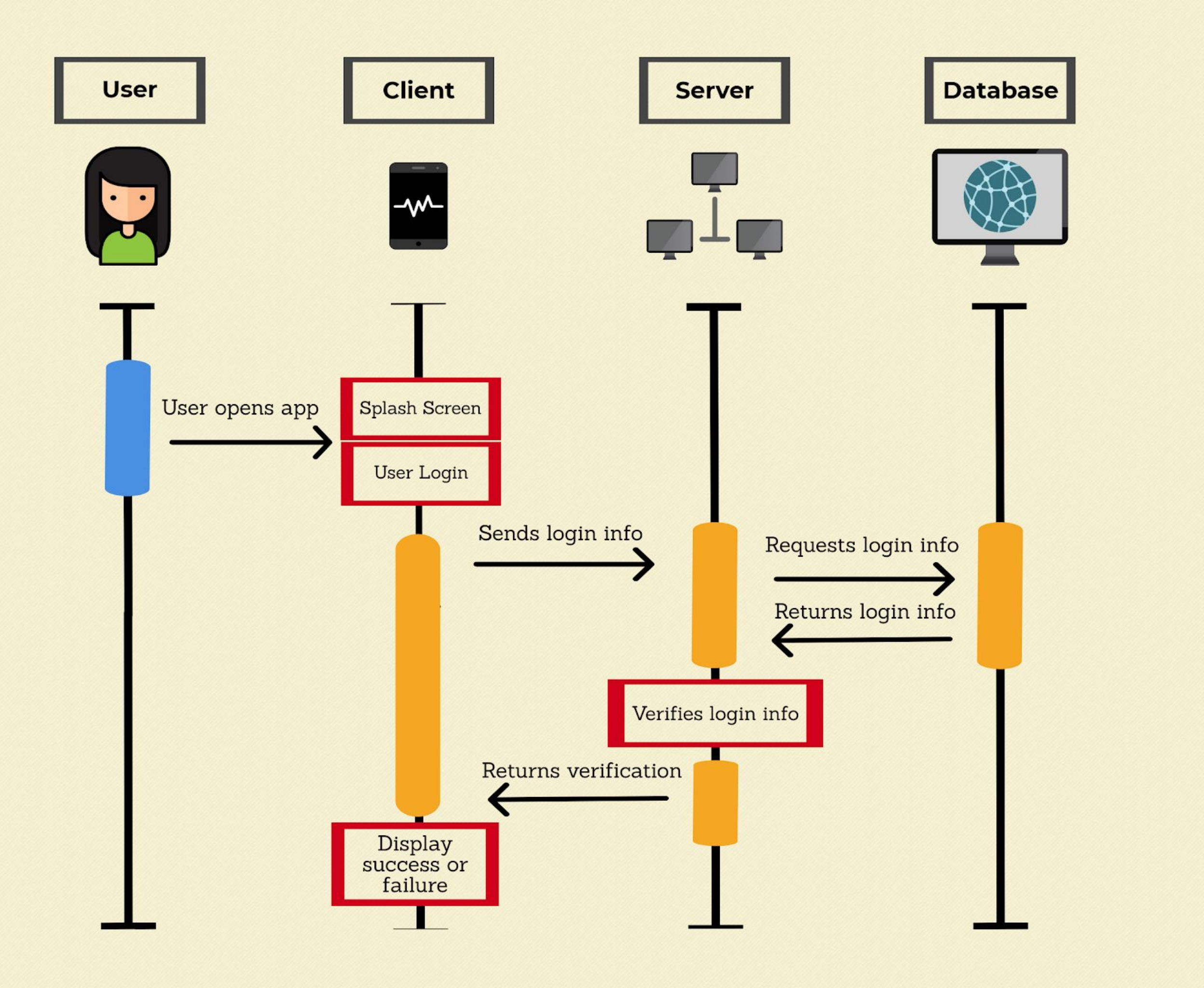

AR Mode (with Login)
(previous sculpture)
VIEW YOUR SCULPTURE
(sculpture being displayed live in Augmented Reality)
(douple tap screen to “Upvote”)
BACK
(next sculpture)
Build Mode
SAVE
Pick a name: ----------
Pick a location: ---------
Privacy: Public/Private
(Sculpture being displayed here)
(Rotate or move individual parts of the sculpture. Or even move the camera angle. This all depends on what tool is currently selected from the tool palette)
(Tool Palette: contains “Add new block”, “Scale”, “Rotate”, etc.)
BACK

Note: The UI design plans here are simply rough ideas of what each panel should contain, actual design may change during development

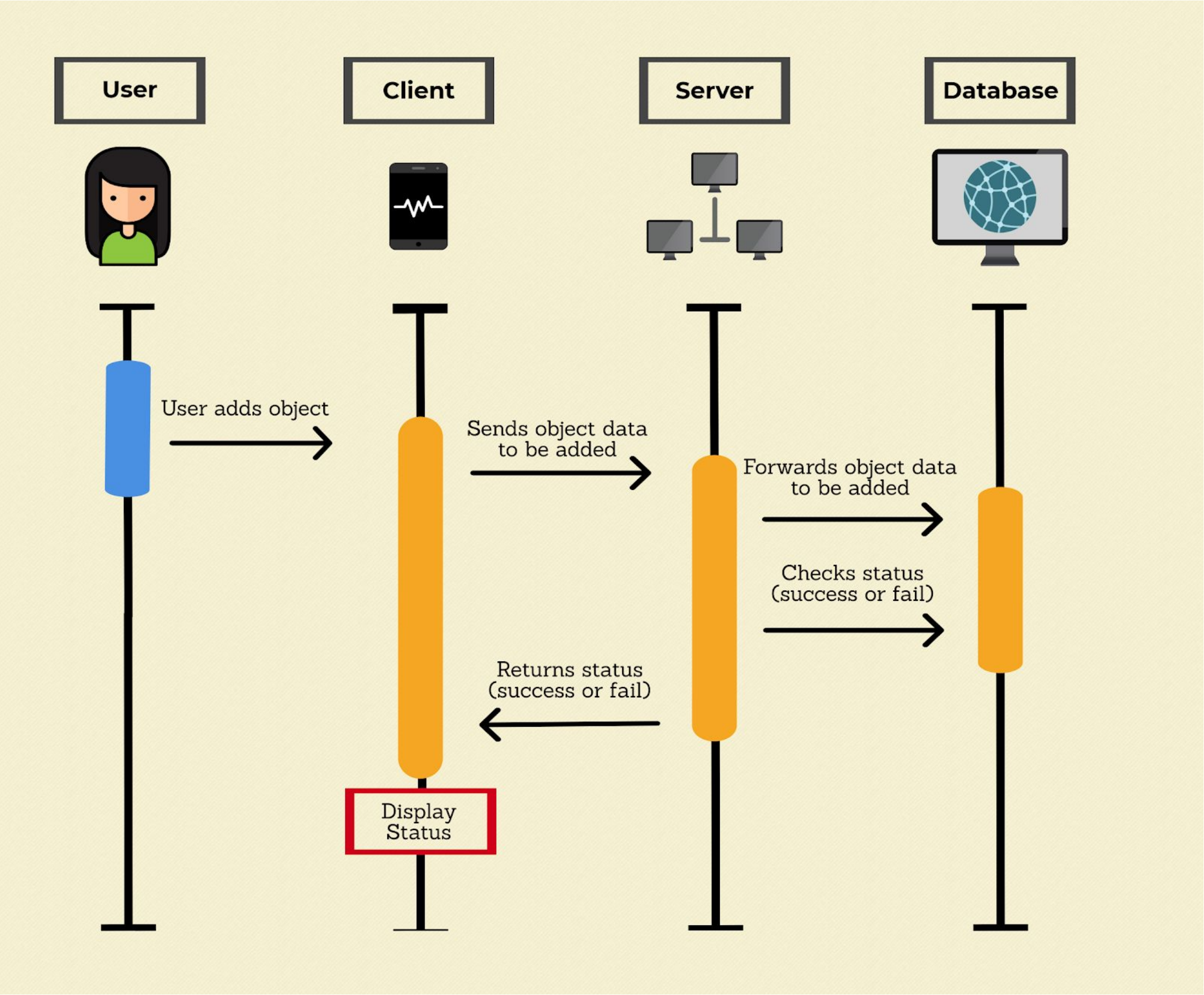
Sequence Diagrams

The following sequence diagrams show the major sequence of events that users will experience as they use Buildr. The sequences depict how a user will interact with the client, server, and database through our application.

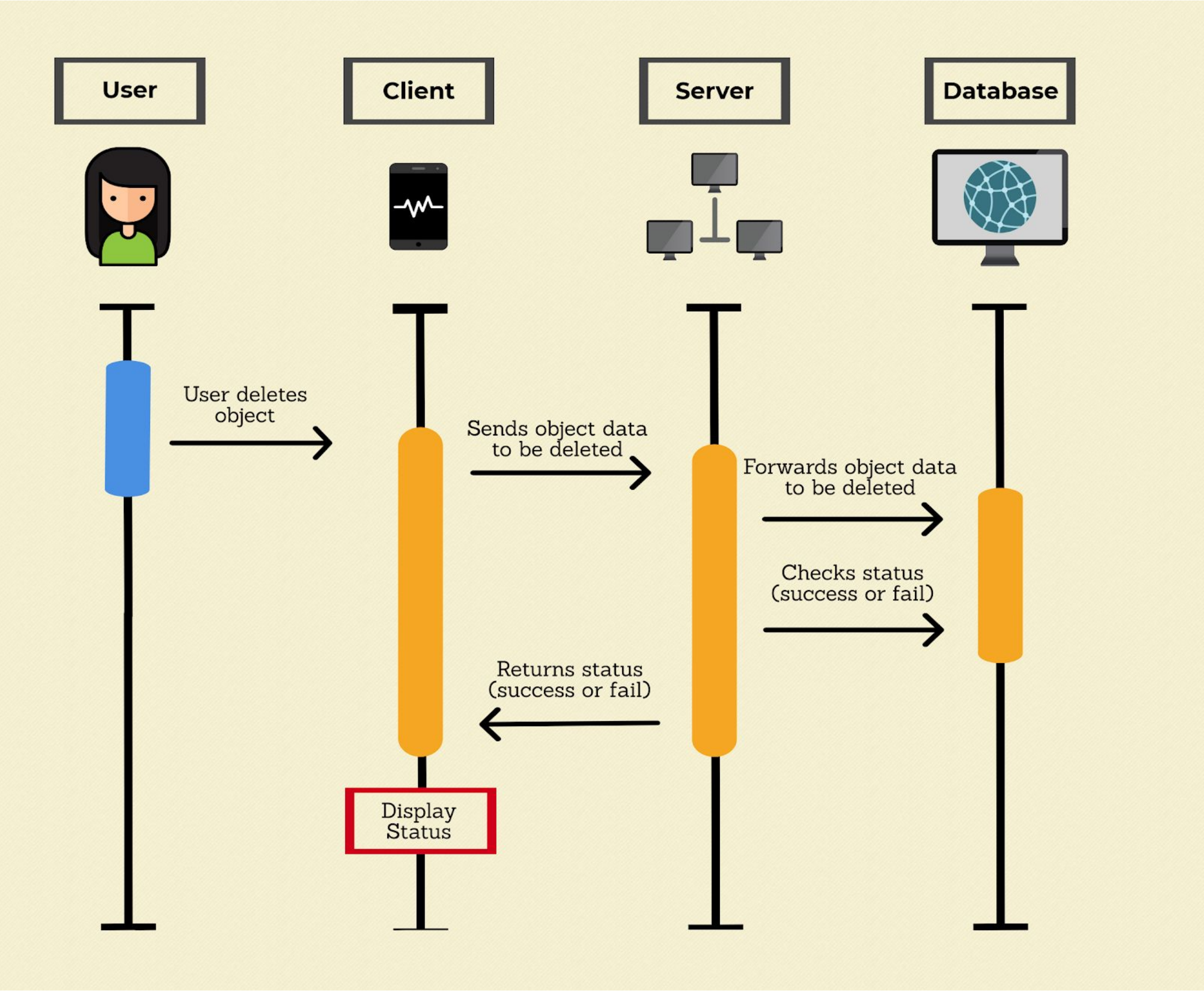
Application Initialization/User Login



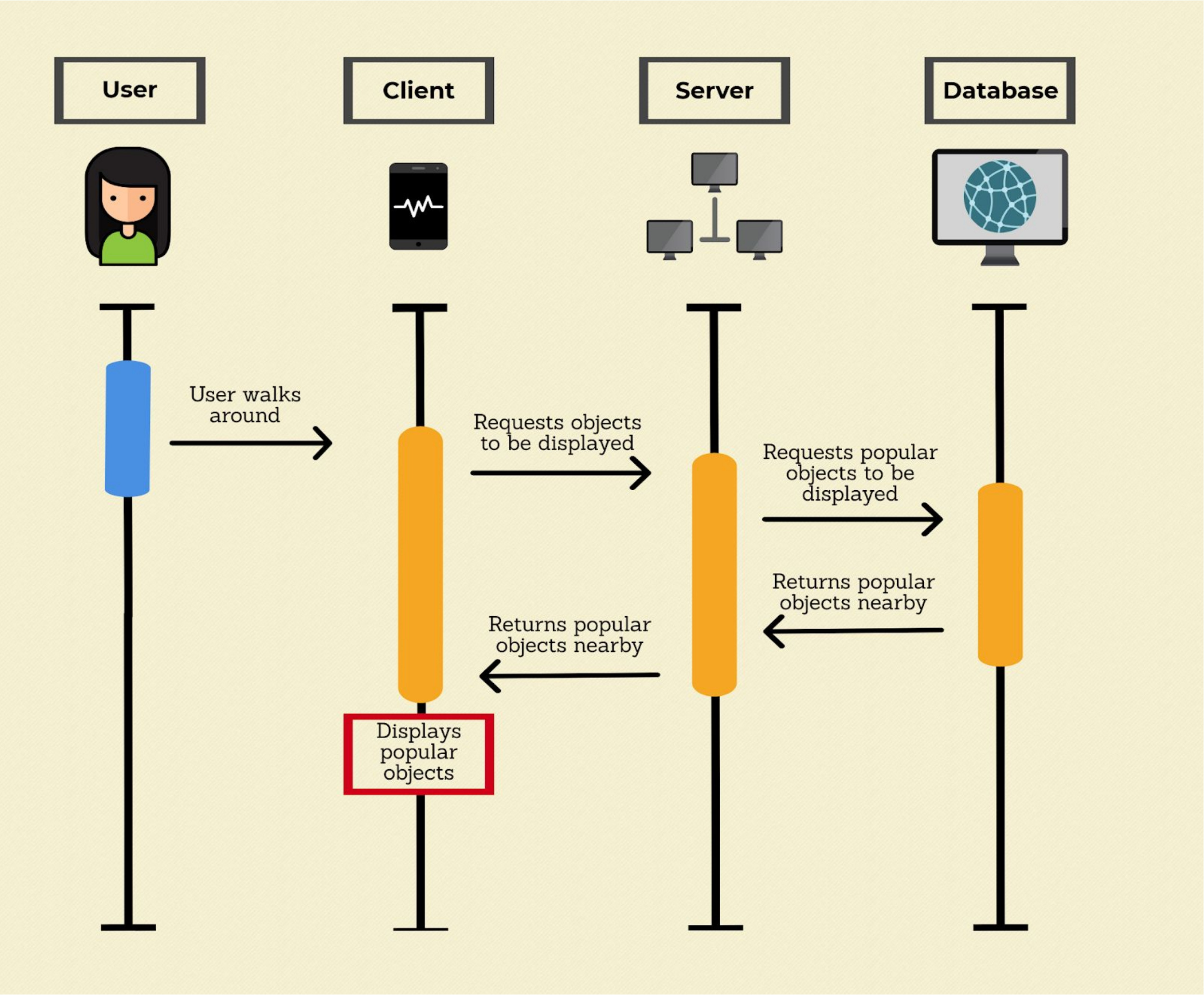
User Adding Objects


User
Client
Server
Database
User adds object
Sends object data to be added
Forwards object data to be added
Checks status (success or fail)
Returns status (success or fail)
Display Status

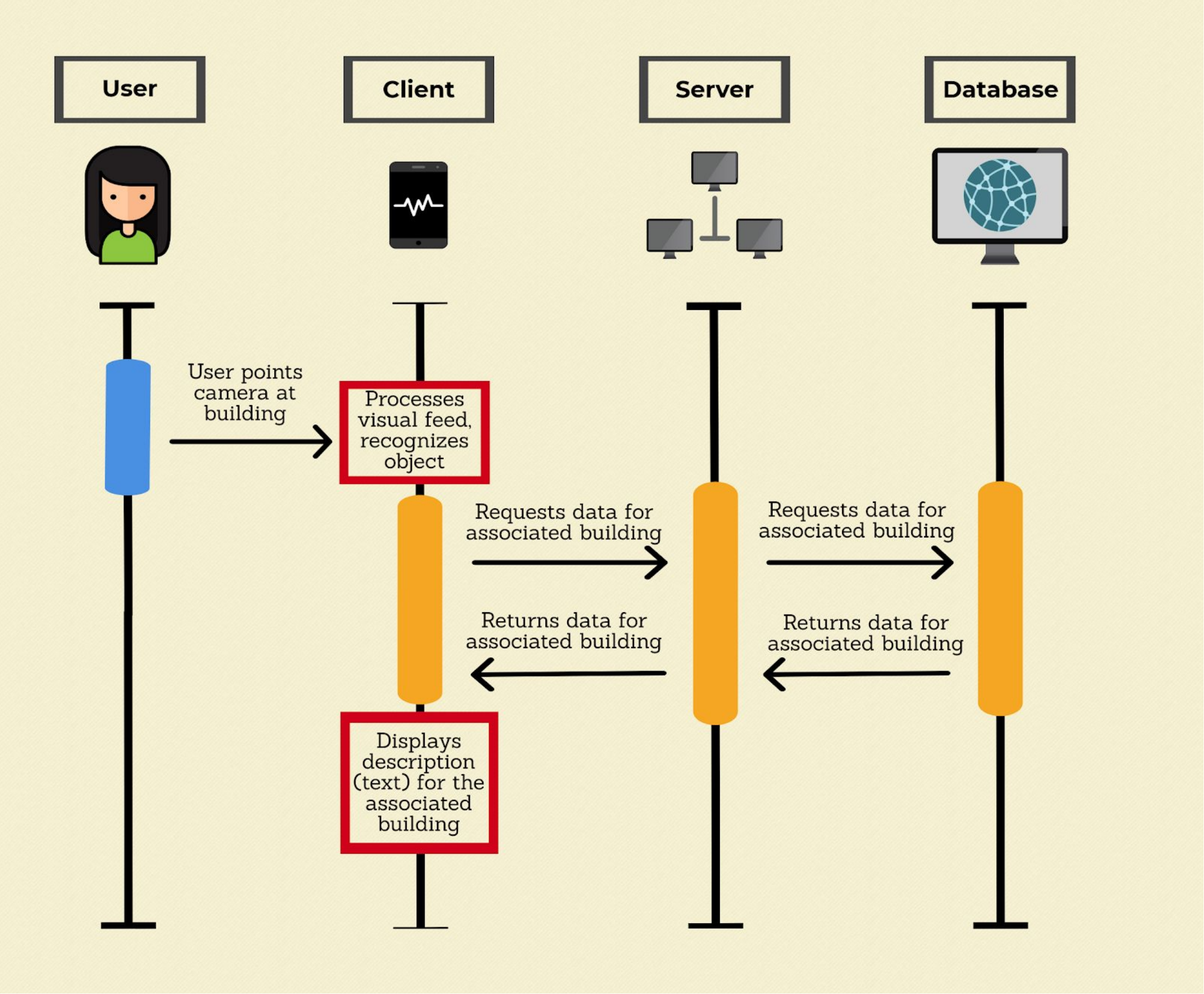
User Deleting Objects


User
Client
Server
Database
User deletes object
Sends object data to be deleted
Forwards object data to be deleted
Checks status (success or fail)
Returns status (success or fail)
Display Status

User Viewing Objects


User
Client
Server
Database
User walks around
Requests objects to be displayed
Requests popular objects to be displayed
Returns popular objects nearby
Returns popular objects nearby
Displays popular objects

User Viewing Historical Buildings


User
Client
Server
Database
User points camera at building
Processes visual feed, recognizes object
Requests data for associated building
Requests data for associated building
Returns data for associated building
Returns data for associated building
Displays description (text) for the associated building