**Evan MacHale**

N00150552 / Year 4 Creative Computing

Design Document / 31st January 2019

Generative Jewellery Web Application Utilising a Combination of 3D Subdivision and Data Structure Algorithms.

**Table of Contents**

[**01 Introduction**](#_9n94vlxf8upb) **3**

[**02 Functionality**](#_hbk6juzcgclf) **4**

[02.1 Input/Outputs](#_qv6iwqjd27xa) 4

[02.2 Use Case Diagram](#_m83otf2fudib) 5

[**03 Interface**](#_ynpnnrg23vkq) **5**

[03.1 Storyboard](#_bndlzojl63w1) 6

[03.2 Wireframe (Predicted Components)](#_pjf7vuq9ijvw) 7

[03.2.1 Login Page](#_l7l9rmm4rgd4) 7

[03.2.2 Playground Page](#_amsud5w25xsg) 8

[03.2.2 Saved Designs Page](#_r2c5yvd4rmz0) 10

[**04 Database**](#_i42jcvw80wcl) **11**

[04.1 Entity Relationship Diagram + Schema](#_2rzypqrn7a6l) 11

[**05 Conclusion**](#_xd3ug64wme4m) **13**

[**References**](#_oyabn3ctrmt2) **14**

# 

# 

# 01 Introduction

The tools that the application development will require are loosely based around an application model discussed in the requirements document. This primarily includes a web application framework with interface tools and a 3D environment in which to implement the project research. Again, as discussed in the technical document.

The framework chosen to build the application interface is React. The server-side technologies chosen to build the application are the Express framework and Node JavaScript Runtime. The database technology chosen is the MongoDB noSQL database. The 3D environment chosen to implement the project concept and research is the Three JavaScript library.

The system design involves utilising the database, server-side technologies, and interface to build an application that allows for interaction and data storage with the 3D environment. The environment will be incorporated into the interface via a library wrapper.

The system may then be described as a full-stack MongoDB, Express, React and Node application(MERN) with Three being wrapped in the React interface building system.

# 

# 02 Functionality

## 02.1 Input/Outputs

Inputs for the interface may be derived from the requirements document. The application is divided into three interfaces, or pages :

*Log in :*

1. Login form
2. Sign up form

*Playground :*

1. Select starting geometry (jewellery type)
2. Manipulate height, width, depth of geometry
3. Choose generative constraints
4. Export .STL file for 3D printing
5. Save model
6. View Account
7. Logout

*Account / Saved Designs :*

1. View list of designs
2. Load a previously saved design
3. Delete a previously saved design
4. Export selected design
5. View Playground
6. Logout

Each interface will be built in React and implement routing technologies for navigation.

The relationships between processes may be observed in the uml diagrams, and may be visualised with the help of the wireframes that follow.

## 

## 02.2 Use Case Diagram

The application has two types of users, defined in the requirements document. However, due to the limited size of the application after multiple development iterations, being only three pages, there is little differentiation. Both users will create an account, play with shapes and then perhaps download or save a design.

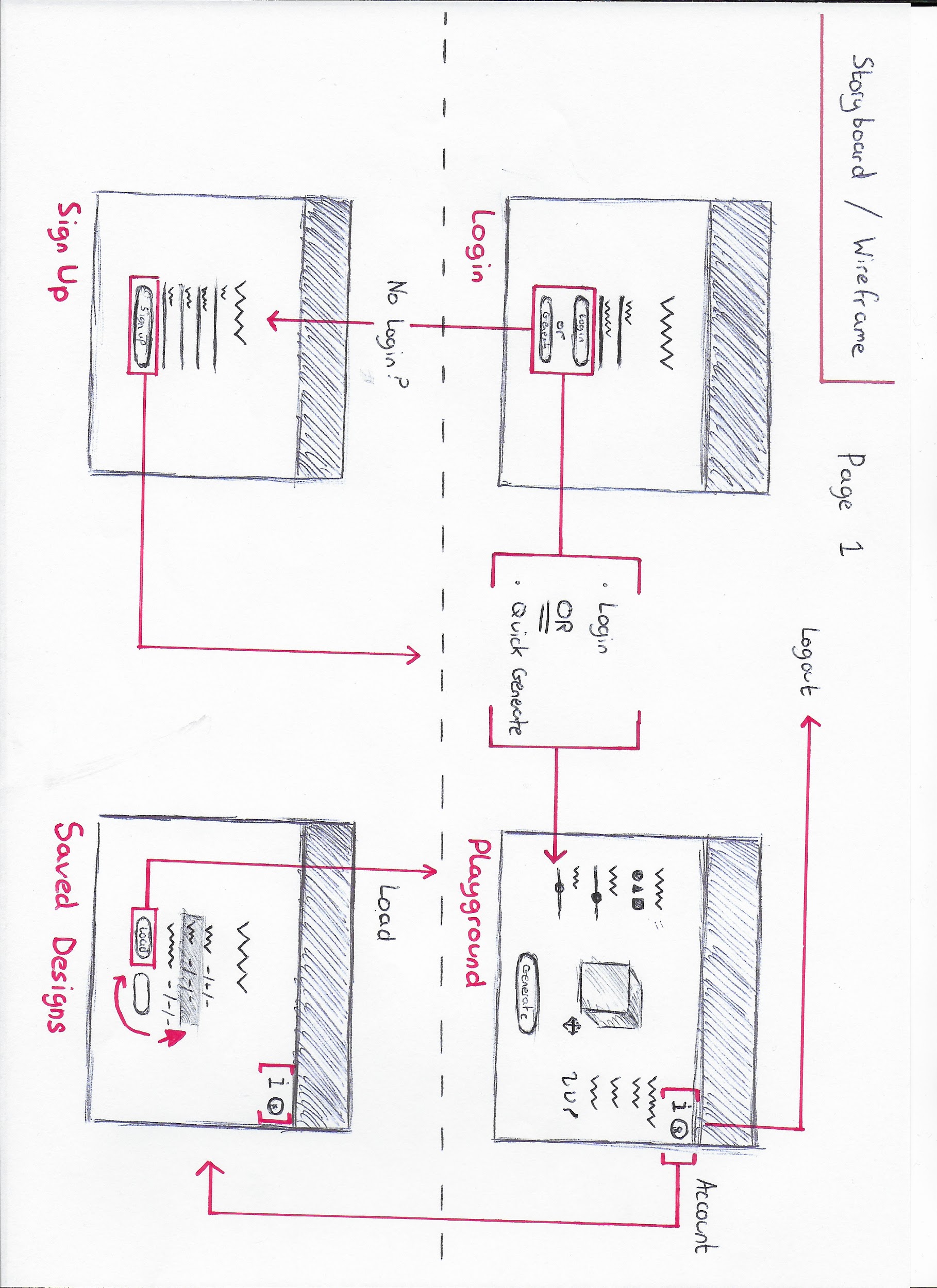
The following use case illustrates the primary tasks, alongside secondary tasks related to each primary.

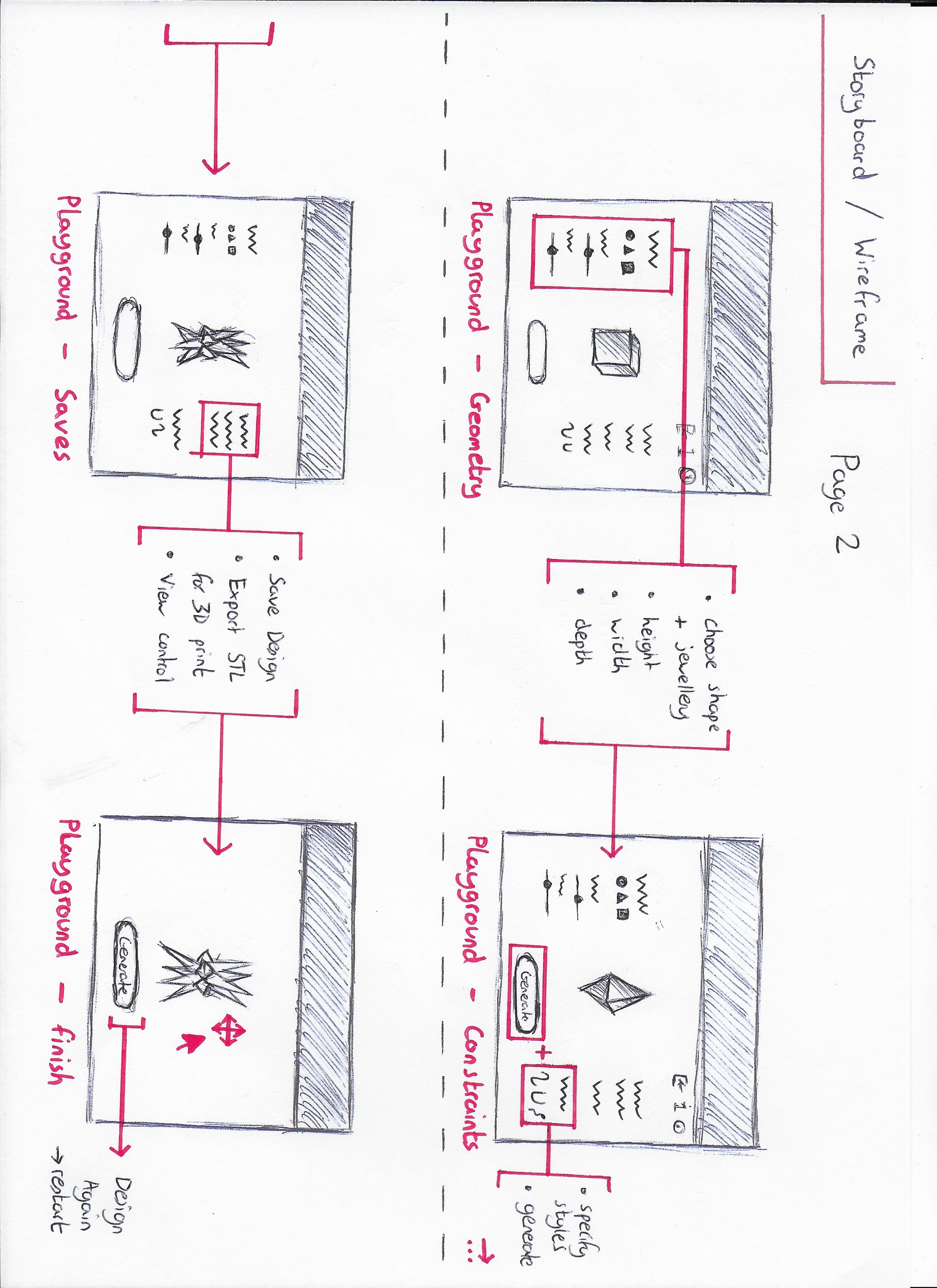
# 03 Interface

Interface design for this project is implemented with the user interaction principles of the Material Design team. This project utilises components from the *Material Design Components(MDC) React* framework.

## 03.1 Storyboard

Referenced from the requirements document :





## 03.2 Wireframe (Predicted Components)

Three pages may be derived from the storyboard :

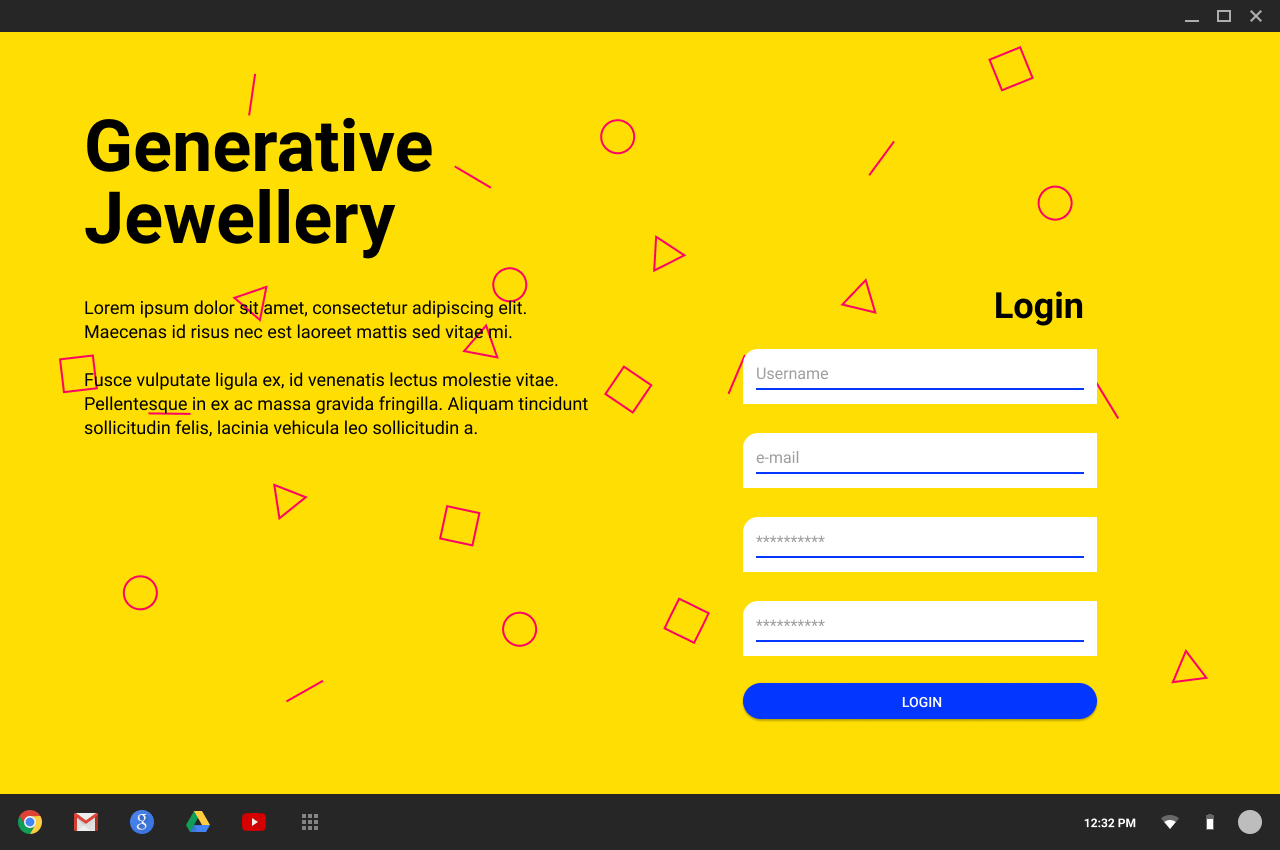
1. Log In / Sign up
2. Playground
3. Saved Designs

These pages are defined based on user research from the requirements document.

### 03.2.1 Login Page

The landing page of the application consists of a simple form utilising multiple *<TextField/>* components. The login and saved designs pages have React form interactions and will both be wrapped in a *<LayoutGrid/>* component for styling and layout.

Potentially there will be a creative-coding simulation in the background of the login page to contribute to an app identity. If doable, this will utilise the *p5* JavaScript library wrapper components for React.



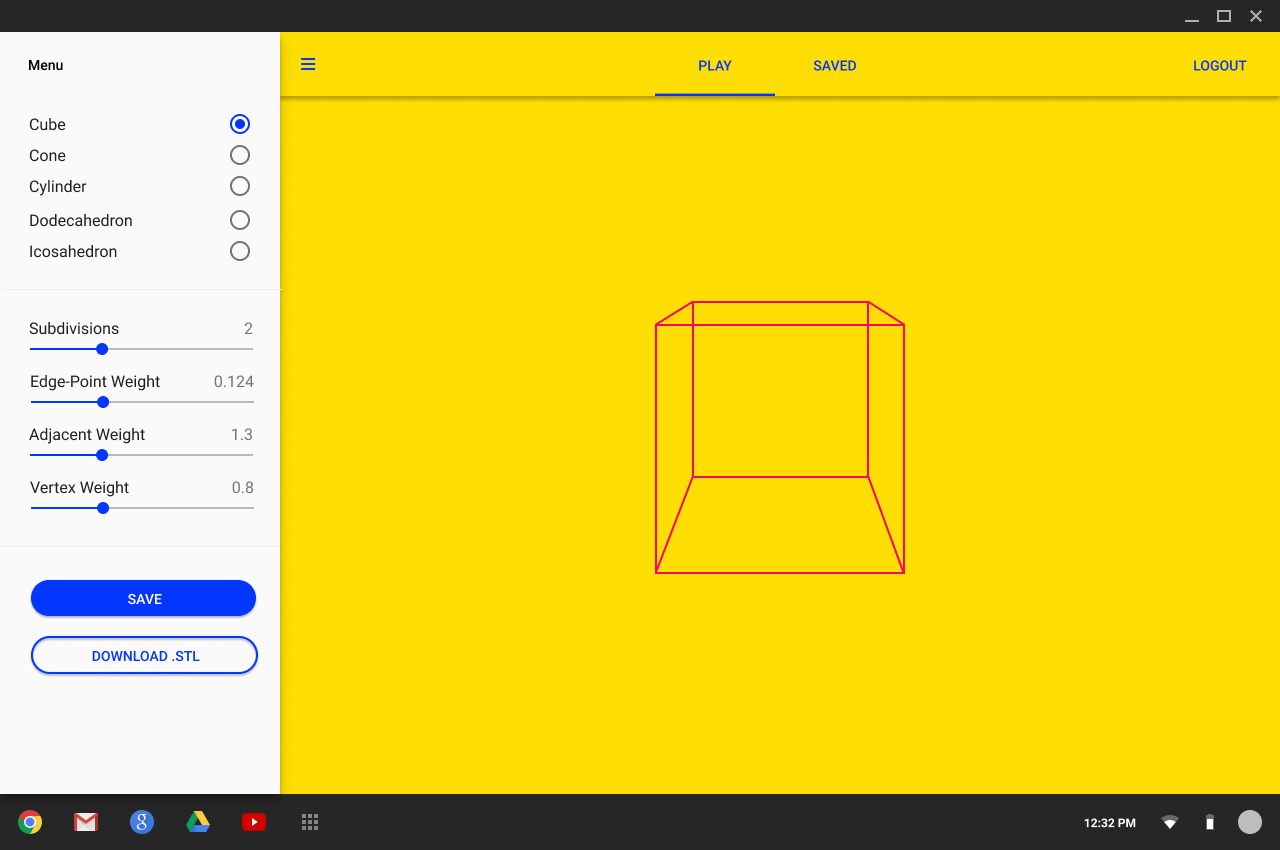
### 03.2.2 Playground Page

All page components in the application will be wrapped in *<HashRouter/>* from *react-router* which will acts as a container for the app.

Inside this may be found MDC *<Drawer/>* and *<DrawerAppContent/>.* These will not be visible from the login page based on conditional rendering.

*<Drawer/>* contains options in an MDC *<List/>* to manipulate shape, this will only be available from the playground page.

Below is *<DrawerAppContent/>*, where the MDC *<TopAppBar/>* lives with react-router tab *<Links/>* and *<Routes/>*, rendering the different pages.



Pseudo-code to visualise how content will be wrapped :



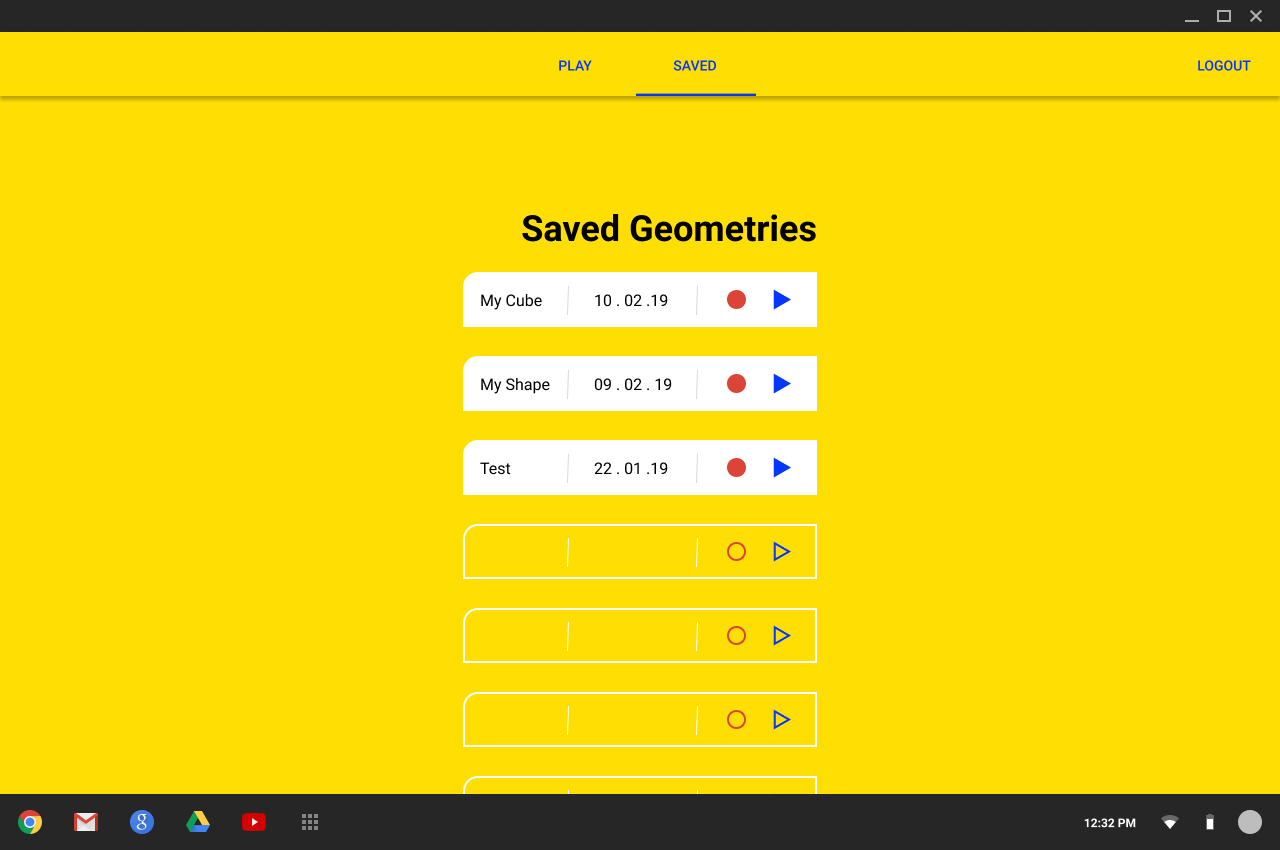
This approach to application architecture allows each page to have access to react-router navigation without having to repeat all this code. Different pages are conditionally rendered in <TopAppBarFixedAdjust/> based on what tab <Link/> is selected.

### 

### 03.2.2 Saved Designs Page

Like the login page, saved designs page does not have access to the <DrawerContent/> component. The app bar for navigation is still inherited.

This page will use a <List/> component to render geometries saved from the playground page component. There will be a *bin* and *load* button trailing each list item.



# 

# 04 Database

The server-side technologies of Express and Node allow for use of JavaScript in communicating with the database. The database will hold a collection of user login information, and information for 3D environment geometries. The database in use is MongoDB.

## 04.1 Entity Relationship Diagram + Schema

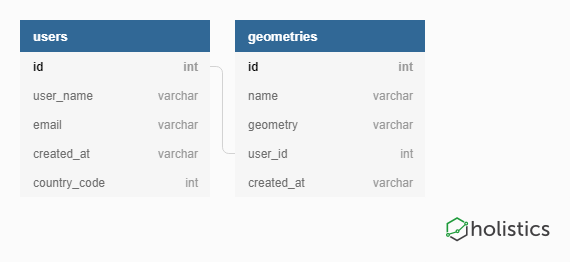
An Entity Relationship Diagram(ERD) is simply a mapping of the data you intend to store and the relations amongst that data. You can still make an ERD with MongoDB as you still want to track the data and the relations. The big difference between traditional SQL databases and MongoDB is that it has no joins.

Unlike SQL databases, where you must determine and declare a table’s schema before inserting data, MongoDB collections, by default, does not require its documents to have the same schema.

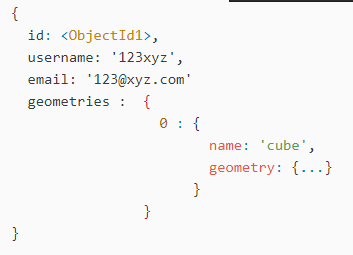
The key decision in designing data models for MongoDB applications revolves around the structure of documents and how the application represents relationships between data. MongoDB allows related data to be embedded within a single document.

So to take advantage of MongoDB we should try to store as much as possible in a single document allowing for efficient querying of data. However, MongoDB imposes a 4MB size limit on a single document. This may cause problems when storing the large geometry data structures and will be subject to change as the project progresses.

So that this :



May be expressed in a single document, as this:



This system of storage is known as *embedded data* in MongoDB. As mentioned due to predictions on file size, when storing a geometry, a reference system may be preferable :



# 

# 05 Conclusion

The list of processes based on the use case diagrams will be built using MDC and the React framework for both interface design and application functionalities. Visualised through the wireframes these components will be divided into three pages alongside a number of utility components such as input fields and application drawers.

The server side will be built using MongoDB data storage principles and connected to React using Express and Node functionalities.

These MERN characteristics will revolve around the Three library component built into React and will make up the majority of application interactions.

# 

# References

MongoDB. (n.d). Data Modeling Introduction. Retrieved from : https://docs.mongodb.com/manual/core/data-modeling-introduction/

Stack Overflow. (2017). MongoDB relationships: embed or reference?. Retrieved from : https://stackoverflow.com/questions/5373198/mongodb-relationships-embed-or-reference

Material Components. (2019). Material Components for React (MDC React). Retrieved from : https://github.com/material-components/material-components-web-react

React Training. (2019). React Router. Retrieved from : https://reacttraining.com/react-router/web/guides/philosophy