VR Exploration Website

CMPE 280 SP18 Project Report

Bruce Decker, Akash Gupta, and Matthew Kwong

https://github.com/mtkwong/280-team-project

Table of Contents

1. Project Introduction & Description

Virtual reality (henceforth occasionally referred to as “VR”) is one of the hottest recent trends in software technology today. While most may be familiar with its manifestation in the gaming industry, it is also true that it can be (and has been) applied towards more constructive uses, such as flight simulation training, enhancing the design & performance of medical devices, and a bevy of applications within the education sector.

Through this project, our team seeks to provide a platform by which users can create, share, and participate in virtual reality experiences, both their own and those of others. At a high level, the project is a web application supporting user authentication, file uploads, and viewing of virtual realities (think of it like Youtube, but for VR).

2. Requirements

Given the broad description of our application above, we can distill it into several main requirements:

1. Catch users’ interest in VR from the moment they open the application
2. Provide a comfortable user experience, in an intuitive and collaborative environment
3. Make it simple for users to create their own VRs and for them to view others’ VRs

2.1. Project Requirements

The following are the detailed functional requirements for our application:

Req ID	Domain	Description
A00	Home page	Allow user to login or register
A01	Home page	Allow user to search
A02	Home page	Provide useful information about VR, to catch users’ interest
A03	Home page	Provide helpful hints about the website (e.g. via a chatbot)
A04	Home page	Add metadata to improve search engine optimization
B00	User profile	Allow user to logout if logged in
B01	User profile	Allow user to upload a new VR, with a description
B02	User profile	Allow user to view their personal details (provided upon registration)
B03	User profile	Allow user to view previews of their uploaded VRs
B04	User profile	Allow user to search
B05	User profile	Allow user to create a VR using an in-place editor
C00	VR details	Allow user to interact with the VR (mouse around; WASD movement for .gltf)
C01	VR details	Show user related or recent VRs as hyperlink previews

C02	VR details	Allow user to navigate to the profile of a user who uploaded a VR
C03	VR details	Allow user to “like” the VR, and display the change in real time
C04	VR details	Show the VR’s details (upload date, description, etc.)
C05	VR details	Show the VR’s comments, and allow the user to add a new comment
C06	VR details	Allow the user to search
D01	Search page	Show the search results of the user query (message if none found)
D02	Search page	Allow user to navigate to the profile of a user who uploaded a VR
D03	Search page	Show the VR’s description and upload date
D04	Search page	Allow the user to search
D05	Search page	Paginate results if there are too many to fit on the page

2.2. Web UI Requirement Principles

In order to increase the chances of the success of our application, our team utilized several web UI requirement principles, which are specialized research methods to gauge the needs and wants of particular demographics or people types who might be using the application.

2.2.1. Voice of the Customer (VOC) Analysis

Our first method of conducting market research was simply to poll our friends & family for their thoughts on a VR exploration website, what kinds of features they would expect, and what type of experience they would desire to have. Many of the people we talked to were non-tech-savvy, so once they understood the core concepts of our idea, they gravitated towards their experience with Youtube to give feedback. Others with more technical backgrounds were able to give more specific input, e.g. considering using blockchain to store certain types of data on our application. This VOC analysis was tremendously helpful in showing us what potential users of the application would want, and helped drive the creation of the requirements.

2.2.2. Focus Group

For our second method, we asked a small group of our peers at SJSU to discuss the VR app and possible features to include. These were mostly software engineering graduate students since we anticipated most of our user base being in that demographic, so their suggestions were more precise (even to the extent of suggesting specific libraries to use), which helped to complement the feedback we received from our initial VOC analysis.

2.2.3. Personas

Our third method did not involve directly interfacing with people; we created three main personas to typify the people who will be using our application. This helped us to identify specific functionalities that should be included, so as to appeal to one or more of the personas. They are as follows:

1. Engineer - The Engineer is a student or working professional who is interested in the software processing of virtual reality scenes, and thus would be concerned about the inner workings of VR. The Engineer would use the website to gain a better understanding of the technology.
2. 3D Artist - The 3D Artist is primarily concerned with modeling 3D scenes, and converting them into a VR format. The 3D Artist would be excited to share his or her scenes with others.
3. Layperson - The Layperson is a hobbyist or average person who is interested in finding something cool to look at, and may follow multiple hyperlinks as they catch their interest.

3. High-level Architecture

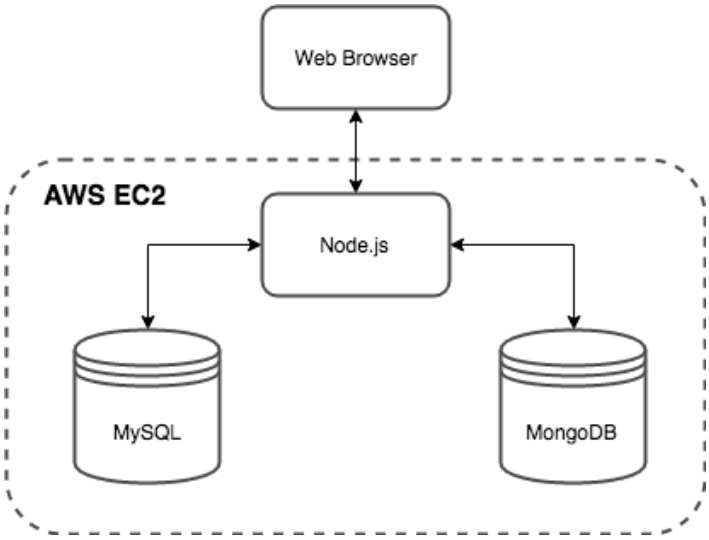

Figure 1. High-level architecture of the VR Exploration Website

At its highest level, our application has a simple architecture. The web browser client connects to a Node.js server, which manages data between MySQL and MongoDB databases. The server and databases are deployed on an AWS instance.

3.1. Components

Perhaps more instructive than the highest-level architecture are component-level diagrams. Below are diagrams & descriptions for each of the major components in the application.

3.1.1. Web Pages

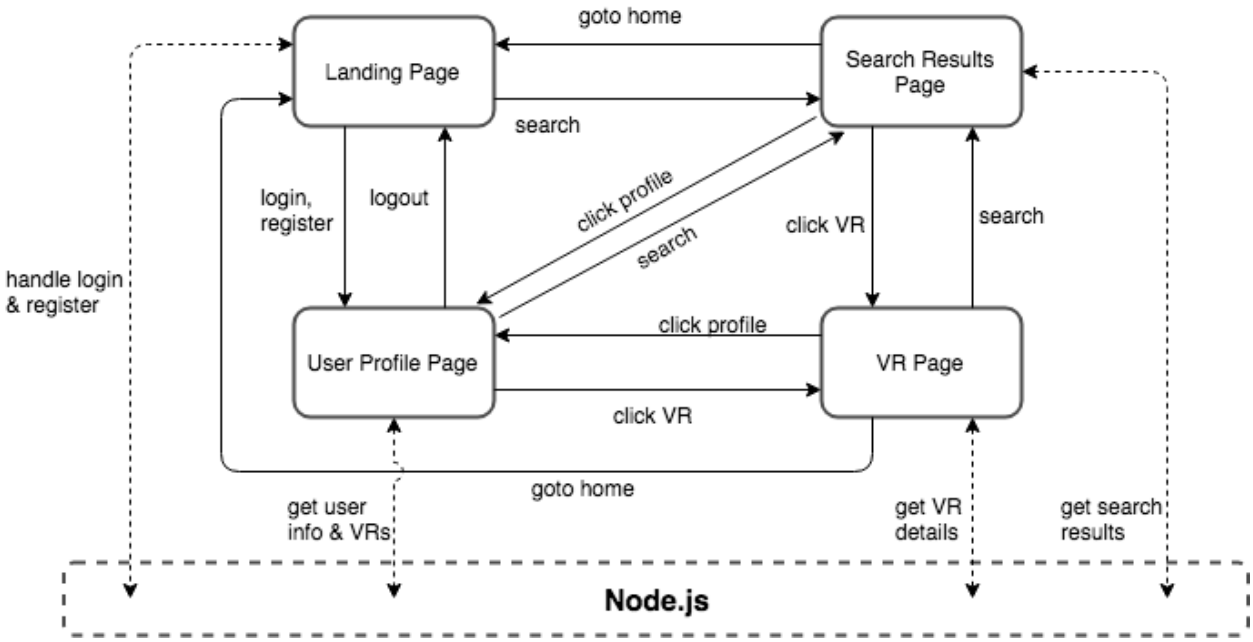

Figure 2. Web browser client architecture

In our web browser client, we support four main pages. The home page (i.e. landing page) is responsible for pulling in & welcoming users to the VR experience, and allows a user to login, register, or search. The user profile is where a user can view their VRs, upload a new VR, and view their profile settings/details (or to view the VRs of another user). The search results page is where a user can view the results for their search query. Finally, clicking on a VR from the user profile or search results page brings up the main VR page, where the user can interact with the VR and see a list of recently uploaded VRs.

3.1.2. Node.js Web Server

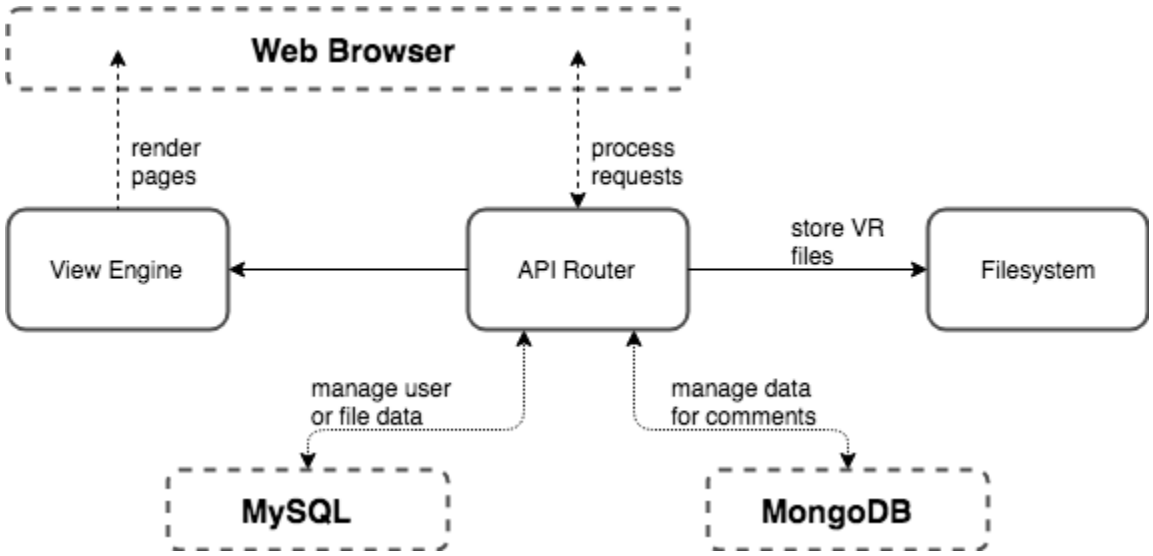

Figure 3. Node.js server architecture

The Node.js server is responsible for rendering the pages, and also contains many other APIs for handling users, files, and comments. The complete list of APIs is listed below in section 4.1. Notably,

uploaded files are stored on the local filesystem of the app, while the file paths are stored in the database. MySQL is used to store users and file data, and MongoDB is used to store comments.

3.1.3. MySQL Database

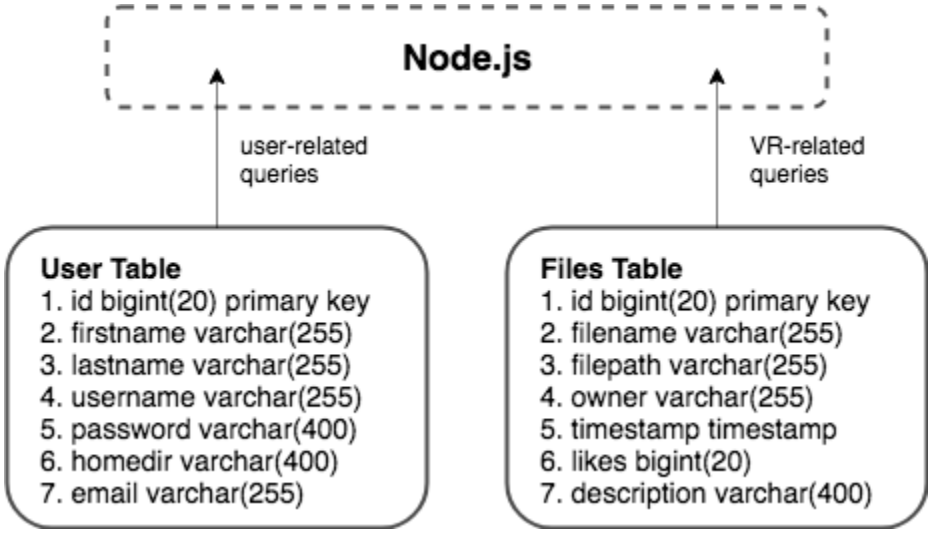

Figure 4. MySQL schema

Our MySQL database is setup with 2 tables - one for users, and one for files. The “owner” field in the files table essentially acts as a foreign key referencing the username from the user table.

3.1.4. MongoDB Database

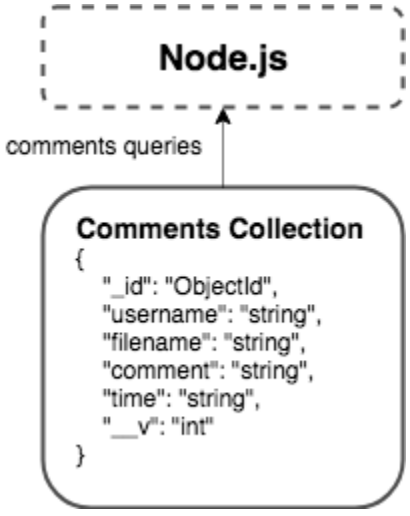

Figure 5. MongoDB schema

Our MongoDB database is responsible for storing comments, which are identified by a combination of username and filename so that they can be displayed on the correct VR.

3.2. Interfaces

Our application has several interfaces both between the UI and the server, and between the server and the database layer. These are best described by the following sequence diagrams of several of the major use cases of our application flow.

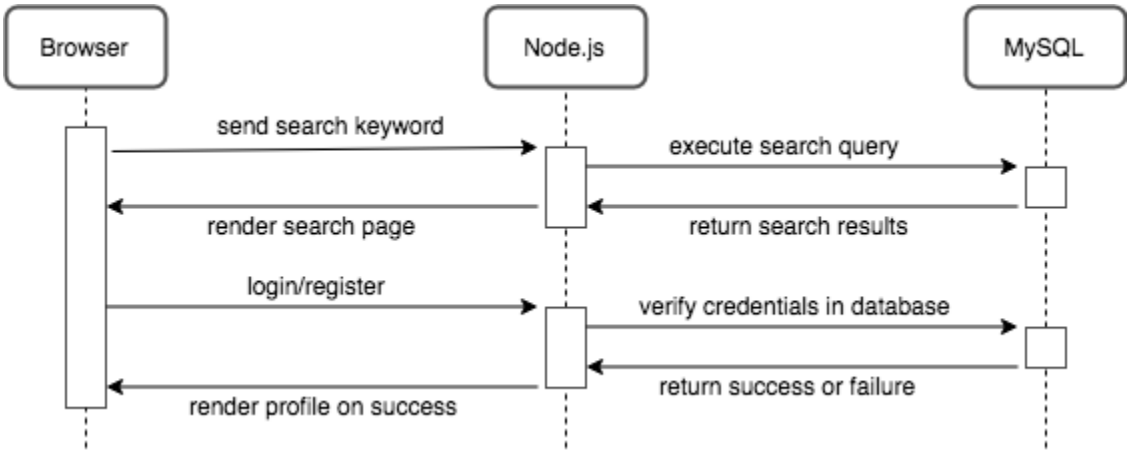

Figure 6. Home page sequences

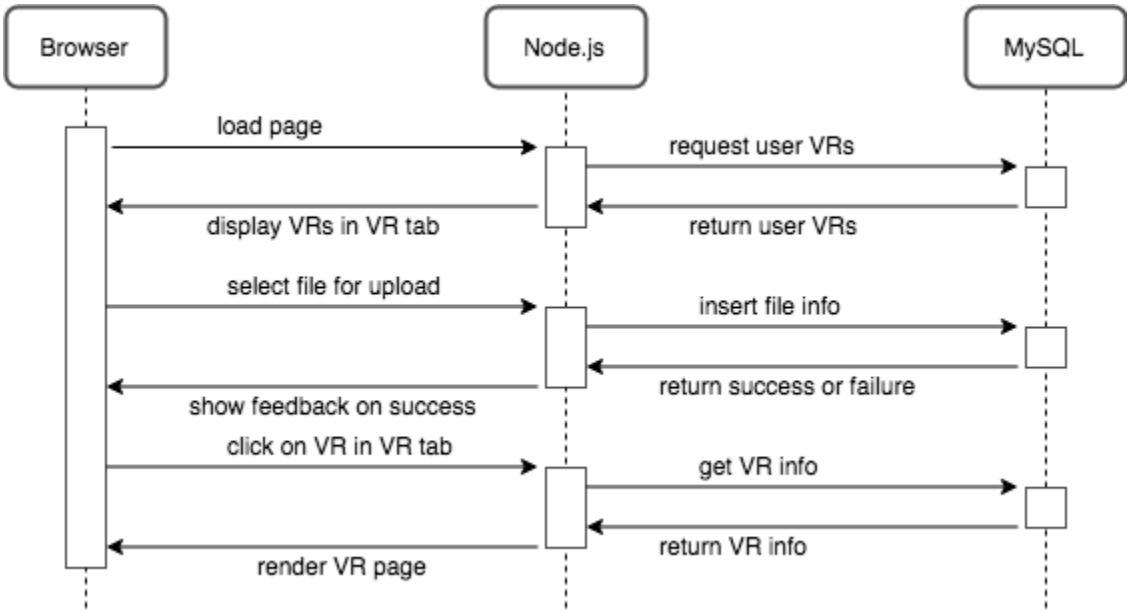

Figure 7. User profile page sequences

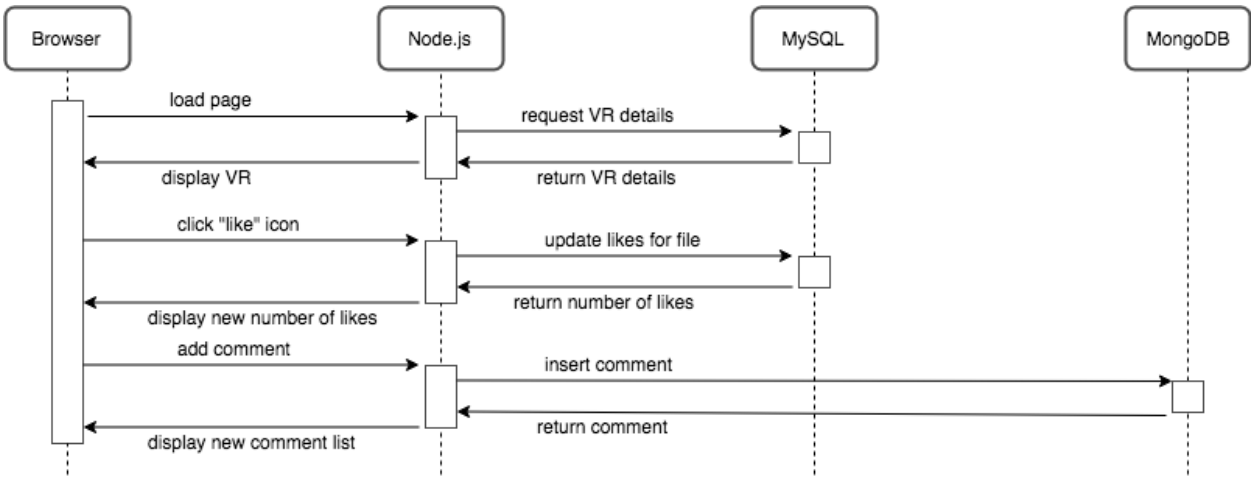

Figure 8. VR details page sequences

4. Server-side Design

The server for our website is a traditional Node.js application, rendering a collection of views, and responsible for serving a collection of APIs. These APIs interface with the MySQL and MongoDB databases, and both the server and databases are hosted on an AWS EC2 instance, to provide our website with a publicly available URL as well as the option for scalability in the future. Here we will list all the APIs called in the app.

4.1. REST APIs

Verb	Route	Description
GET	/	Display landing page
GET	/users	Display a particular user's profile
GET	/users/getUserInfo	Get the information for the currently logged-in user (if any)
POST	/users/signup	Register a new user
POST	/users/login	Log in an existing user
GET	/users/logout	Log out a logged-in user
POST	/search	Display search page
POST	/files/upload	Upload a new VR file
POST	/files/download	Download an existing VR file
GET	/files/uservrs	Get the VRs for a particular user
POST	/files/search	Get the files that match a search query keyword
POST	/files/like	"Like" a VR

GET	/vrs	Display a particular VR
GET	/vrs/getVrInfo	Get the details for a particular VR
GET	/vrs/getOtherVrs	Get the summary of the 10 most recently uploaded VRs
GET	/vrs/getComments	Get the comments for a particular VR
POST	/vrs/submitComment	Submit a new comment for a particular VR

Table 1. List of server APIs

5. Client-side Design

The client makes use of several basic webpages to handle all the functionality needed for our VR exploration website. Most of the pages in the app dynamically generate the DOM based on the data returned by the server. The VR editor on the user page is handled by the three.js editor running in an <iframe> element.

5.1. Web UI Design Principles

In order to plan the layout and design of our application, we employed several Web UI Design Principles, which are designed to abstract the main ideas of the app prior to any coding. Below are the storyboard and several wireframes that we created while in the design phase.

5.1.1. Storyboard

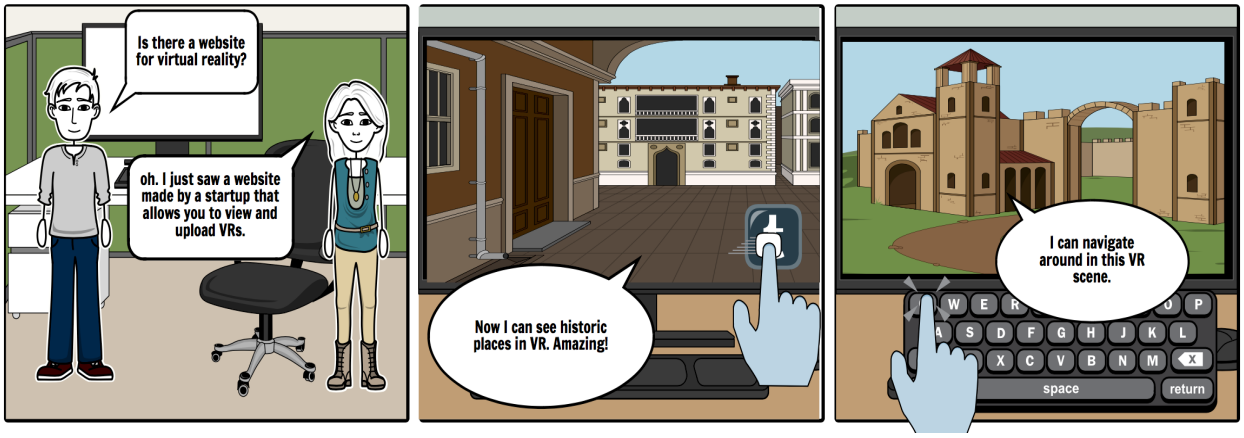

Figure 9. Storyboard of the website concept

Left panel: generate user interest
Middle panel: browsing VR
Right panel: exploring VR

5.1.2. Wireframes

For our application, we have generated the following wireframes to describe the general structure of each page, as well as different views within each page.

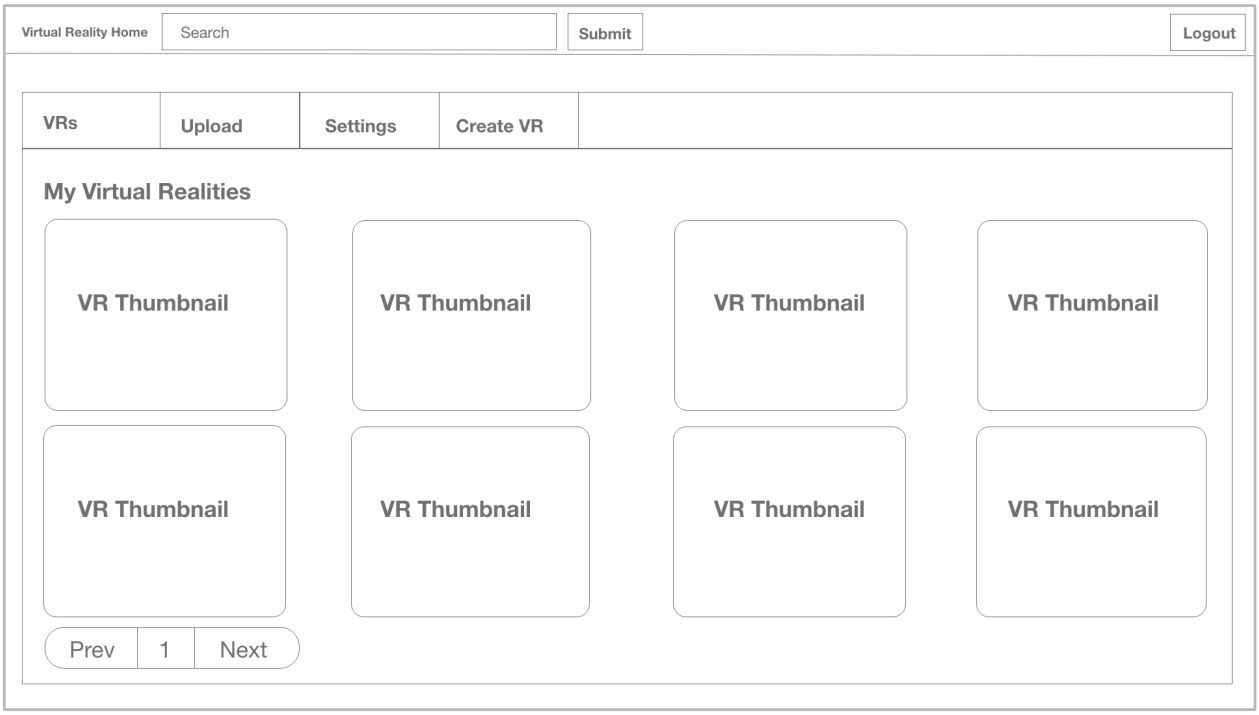

Figure 10. Virtual Reality Home - My VRs

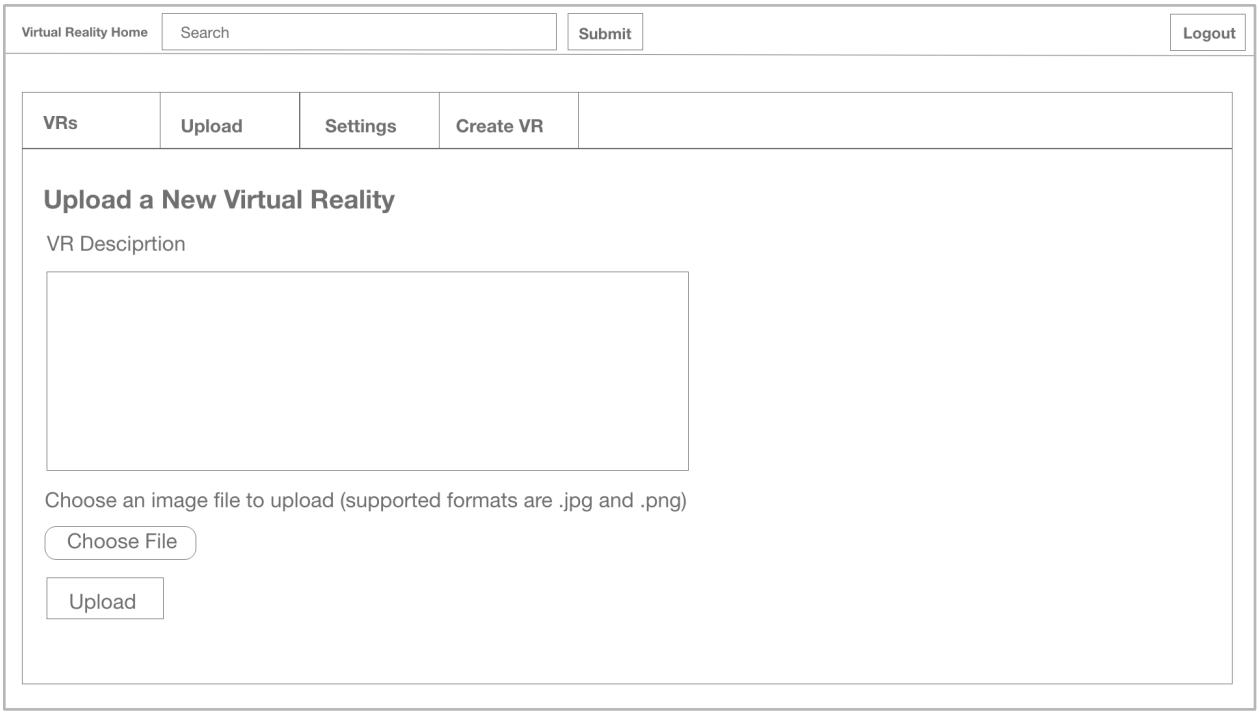

Figure 11. Virtual Reality Home - Upload

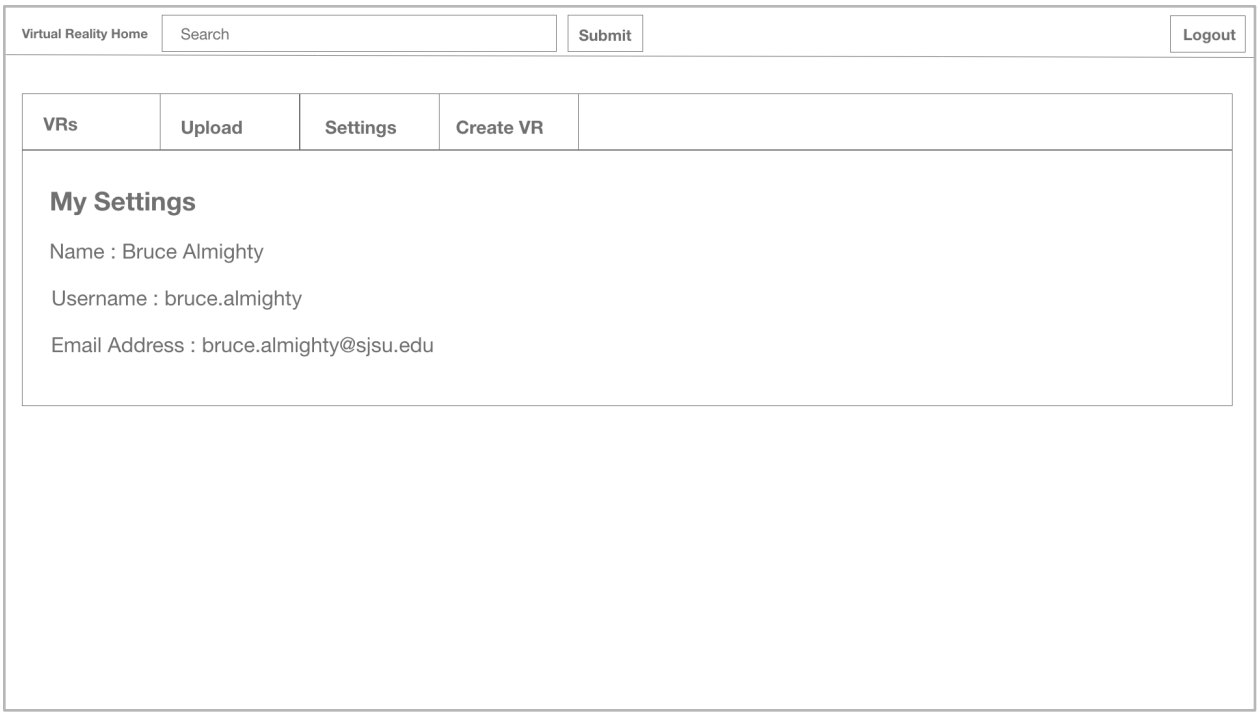

Figure 12. Virtual Reality Home - Settings

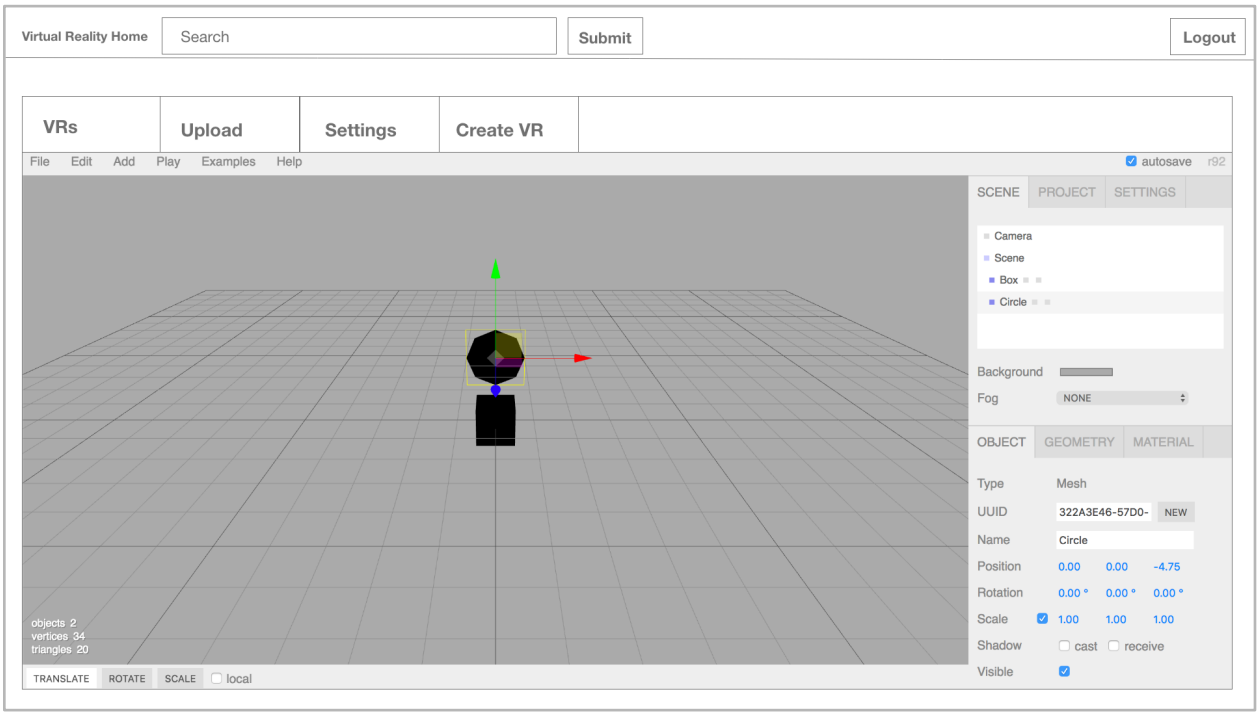

Figure 13. Virtual Reality Home - Create VR

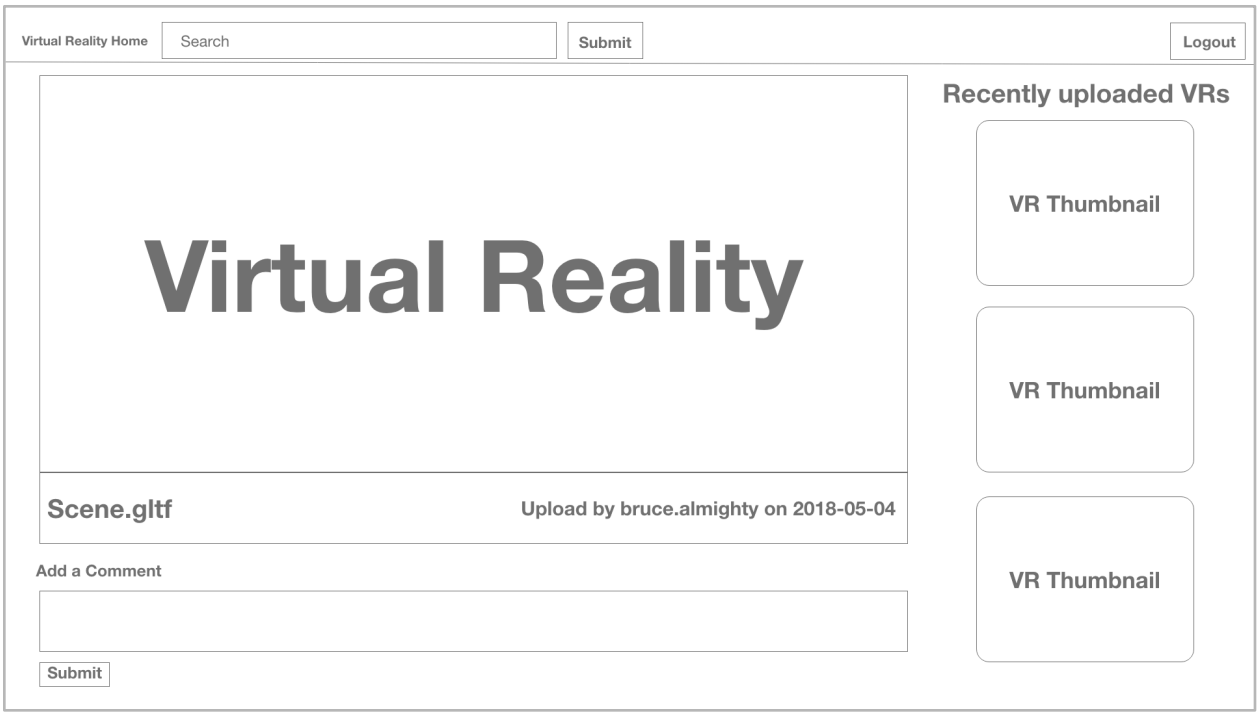

Figure 14. Virtual Reality Page

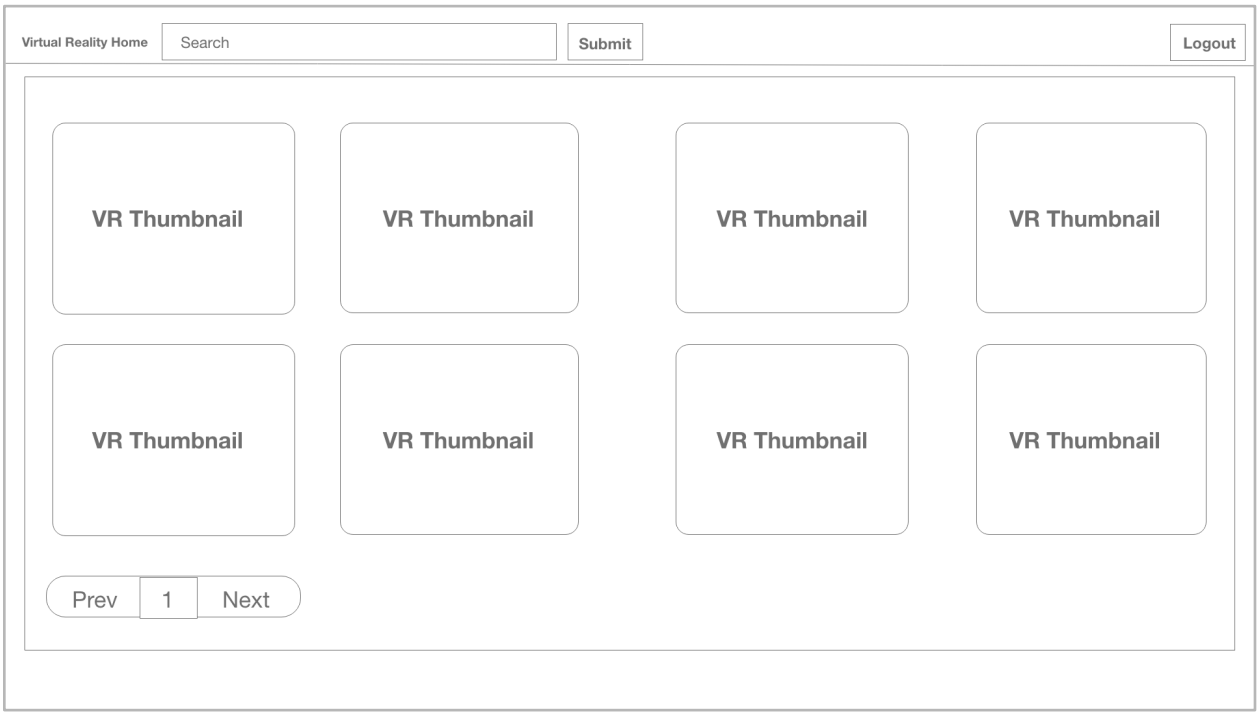

Figure 15. Search results

5.2. Bootstrap

Bootstrap styling is used for most of the landing page (to create the carousel of images, and the contact form at the bottom of the page, for example). It is also used for the header menu that is common across

all the pages. Smaller parts of the Bootstrap style suite were also used to style individual HTML elements such as buttons of input fields, and the Bootstrap column layout was used in several places as well.

5.3. HTML5

The primary HTML5 elements we used in our app were <nav> for the navigation bar header links, <canvas> to display the VRs to users (this is handled automatically by our VR library), and custom “data-” attributes to assist with some navigation bar functionality.

5.4. JavaScript Libraries

5.4.1. jQuery

JQuery is used heavily to dynamically generated DOM elements after the server returns some data. For example, we use jQuery to create the table of VRs whenever a user views a user’s uploaded VRs, or obtains a search result. The other major usage of jQuery comes from the many AJAX requests made by the UI to the server; we use jQuery’s “ajax” method to make those GET and POST requests, and then to move on to the aforementioned process of dynamically processing the data on a successful response.

5.4.2. A-Frame

For our project, we decided to use A-Frame as the JavaScript virtual reality library. We chose A-Frame because of its simple integration with the HTML DOM, and thus easy manipulation through jQuery. A-Frame is also widely supported on multiple browsers & platforms, since it is built upon the common WebVR baseline. The other benefit A-Frame brings is its ability to handle both 360 degree image files as well as full 3D scene GLTF files.

6. Tests

As a whole, our application was thoroughly tested at every stage of development because we kept thinking of new features to add, and every feature (especially those that required a server code change) needed to be tested end-to-end to make sure things worked well.

6.1. UI Testing

The majority of our UI testing was accomplished via manual testing of each feature as it was implemented along with stress testing of previously existing features when new ones were implemented. The modular structure of our UI (split between several independent pages) gave us the freedom not to have to worry about changes in one page affecting other pages. We also performed automated testing with Selenium.

6.2. Server Testing

For our application, we wrote unit test cases for testing some of our important backend server APIs using the Mocha and Chai Javascript test frameworks. Here is a sample of the mocha test output:

```
akashs-mbp:280-team-project akash$ npm test

> backend@0.0.0 test /Users/akash/Desktop/280-team-project
> mocha


  Search VR API
200
    ✓ Should return 200 response

  Like VR API
200
    ✓ Should return 200 response

  List VRs owned by a user
200
    ✓ Should return 200 response

  Signup API
200
    ✓ Should return 200 response

  Login API
200
    ✓ Should return 200 response

  Render user page API
200
    ✓ Should return 200 response


  6 passing (81ms)
```

Figure 16. Mocha test output

6.3. Interface Testing

Interfaces were tested early, since we made sure to define the server API routes first prior to working on the UI. Once the interfaces were defined, we tested those interfaces thoroughly as the UI was implemented. We often performed interface tests through Postman to ensure that the APIs were working correctly, then ensured that the UI was passing data to the server in the same format that the APIs were expecting.

6.4. Automated Testing using Selenium

For our application, we wrote a few automated tests in Selenium using the Python API. The tests go through a basic use case of the application: searching for a keyword that will give many results; navigating to the next page of results; “clicking” on a VR on that page; and “liking” the VR.

7. Other Features

7.1. Cross-Browser & Cross-Platform Compatibility

Our application supports multiple browsers, not using any browser-specific technologies (and all the libraries we incorporated are widely supported). The web exploration website also works on multiple platforms, with page elements rearranging themselves intuitively on mobile devices, for example. For this we made use of such features as Bootstrap’s collapsing menu item feature in the navigation bar.

7.2. Search Engine Optimization

In order to better optimize our application to be discoverable by search engines, we used a few techniques. The first is that we made sure to create the <meta> description tag in the header of our home page, and to set the <h1> and <title> tags to “Virtual Reality”. In fact, we have placed that phrase in several key locations in the page content in the app to increase its keyword density. Finally, we added support for robots.txt to ensure that bots have access to the landing page, user page, and VR page, but not the search page (since it contains only dynamic content).

7.3. Profiling

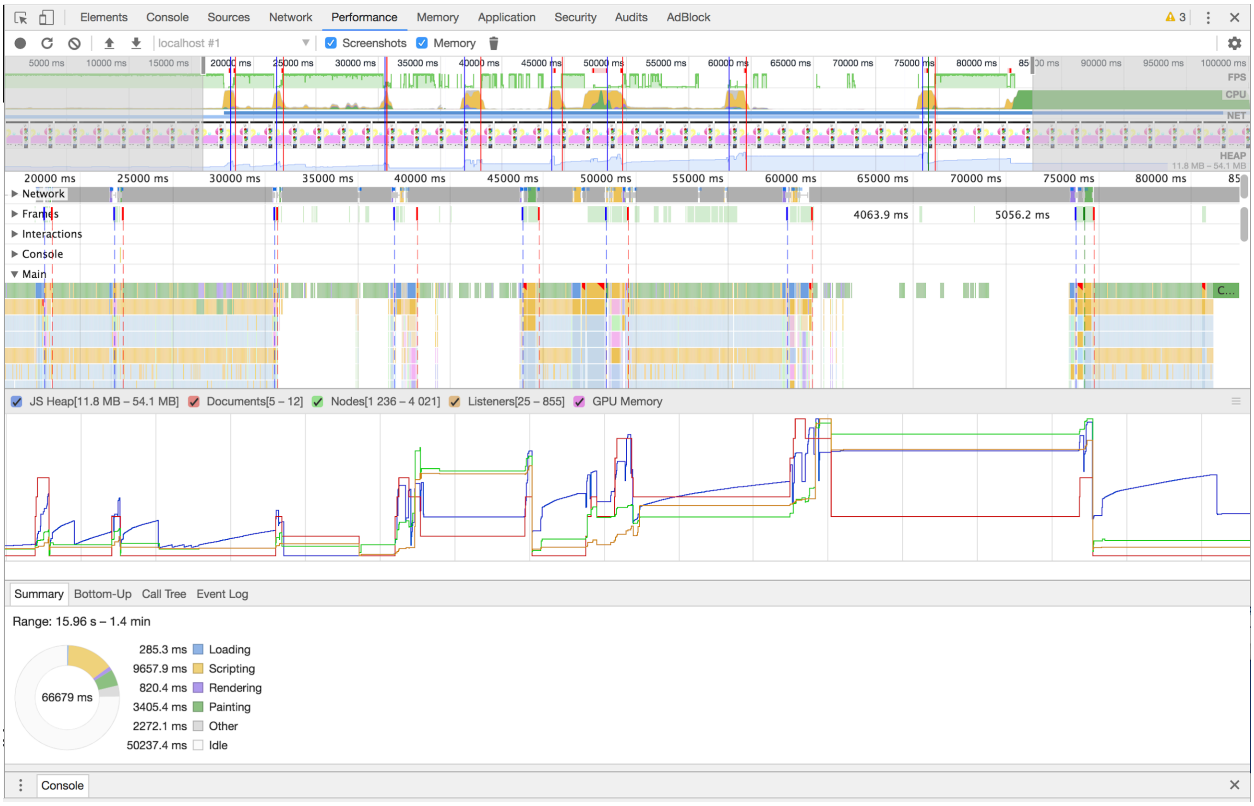

Figure 17. Profile of application workflow

7.4. Chatbot

DialogFlow, formerly known as API.AI, was used in this project to help users understand this website and learn more about VR. It is powered by Google’s machine learning running on the Google Cloud Platform, and can be scaled up to support many users. This chatbot was trained to answer the most common questions related our website, so that users don’t have to email or call a customer service representative to learn about the website. The implementation of this chatbot allows users to query information with ease. Once the user enters a request, the request is sent to DialogFlow, which handles all requests and formulates a response, which will then be displayed in the chat window.

In addition, DialogFlow supports 14+ languages including Brazilian, Portuguese, etc. Google bought the startup API.ai in 2016. Google has been developing it as a tool supporting Google Home Assistant. We came up with a list of questions and answers for chatbot to answer, and we then trained the chatbot to respond to different sorts of questions.

List of questions answered by the Chatbot:

1. What is virtual reality?
2. Can you tell me more about virtual reality?
3. What is virtual reality used for?
4. What framework in VR are you using?
5. Why is VR going to be the future?
6. What kind of VR files can I share?
7. How can I create 3D models for virtual reality?

8. Team Contributions

8.1. Bruce

1. Wrote user comments and saved user comments in MongoDB
2. Designed and wrote landing page
3. Did initial research into VR frameworks like a-frame and experimented with a-frame
4. Made a chatbot for the website and trained it
5. Implemented a blockchain for contact requests made from the landing page

8.2. Akash

1. Helped come up with the initial backend architecture (how users uploads are going to be managed on the server).
2. Wrote backend APIs dealing with users, files and search.
3. Implemented connection pooling in MySQL to improve application performance with higher number of concurrent users.
4. Wrote unit test cases for backend APIs using Mocha and Chai JS frameworks.
5. Deployed the application on an AWS EC2 instance.

8.3. Matthew

1. Wrote the user profile, VR details, and search result pages
2. Integrated most of the server APIs with the web pages, including login, register, file upload, search, and obtaining a list of VRs
3. Wrote the Selenium automated tests
4. Helped come up with the final MySQL database schema
5. Created architecture diagrams & wrote most of the project report
6. Setup Google Analytics for the website running on the EC2 instance

9. Application Screenshots

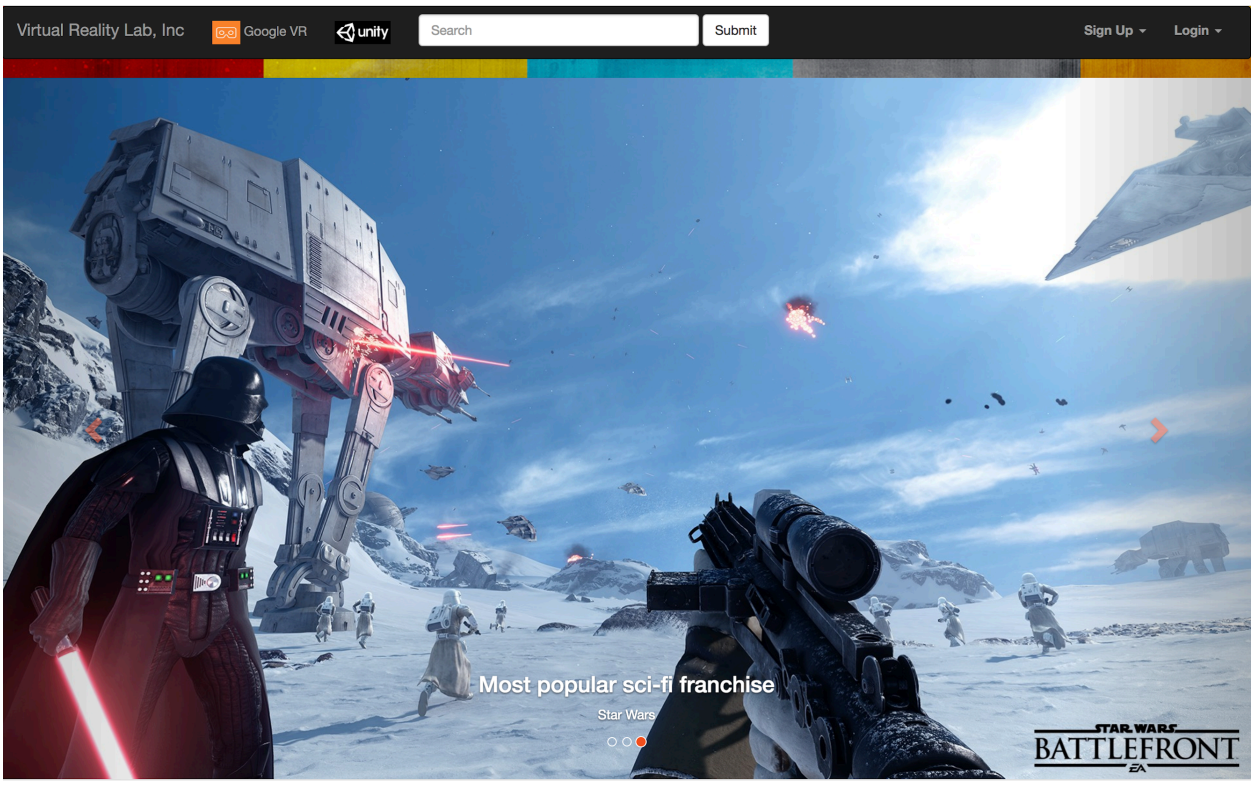

Figure 18. Landing page carousel

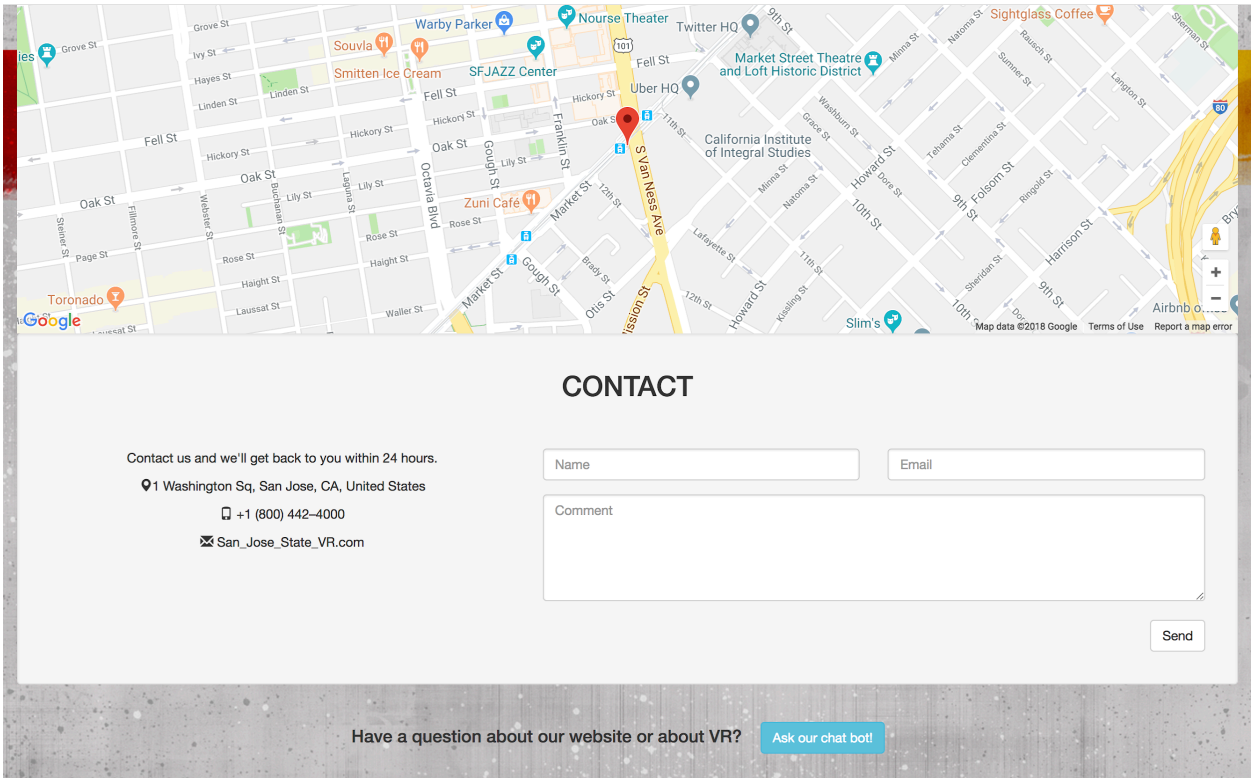

FIgure 19. Landing page map & contact form

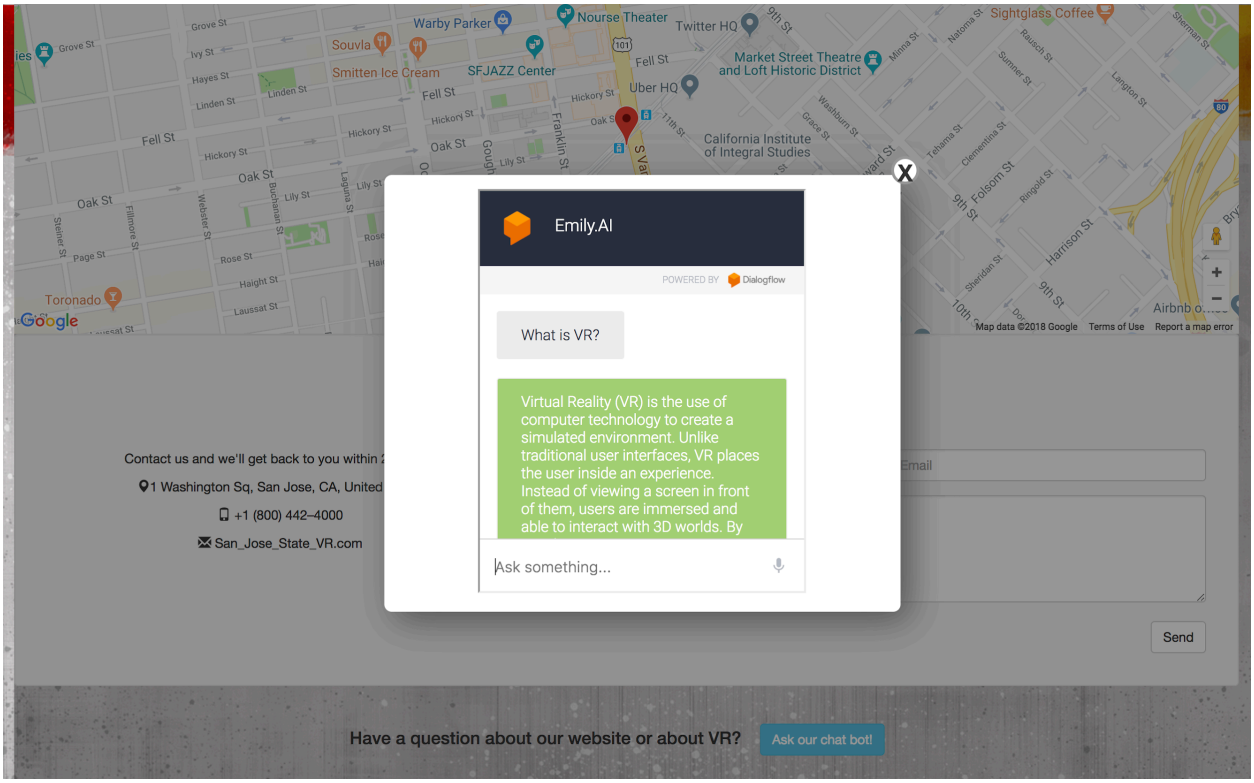

Figure 20. Landing page chatbot

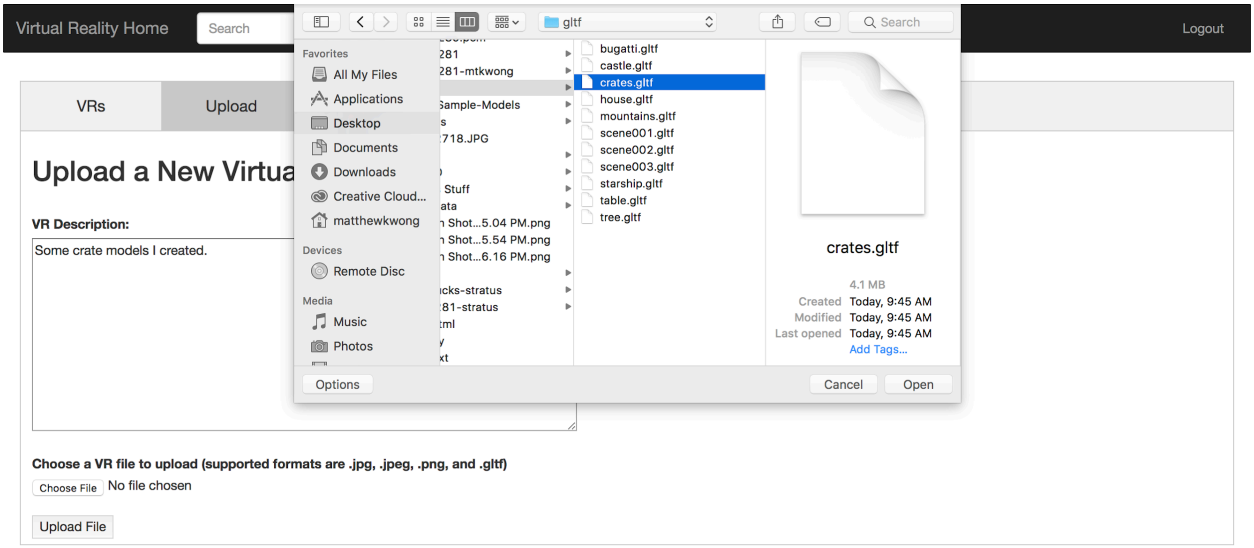

Figure 21. User profile upload

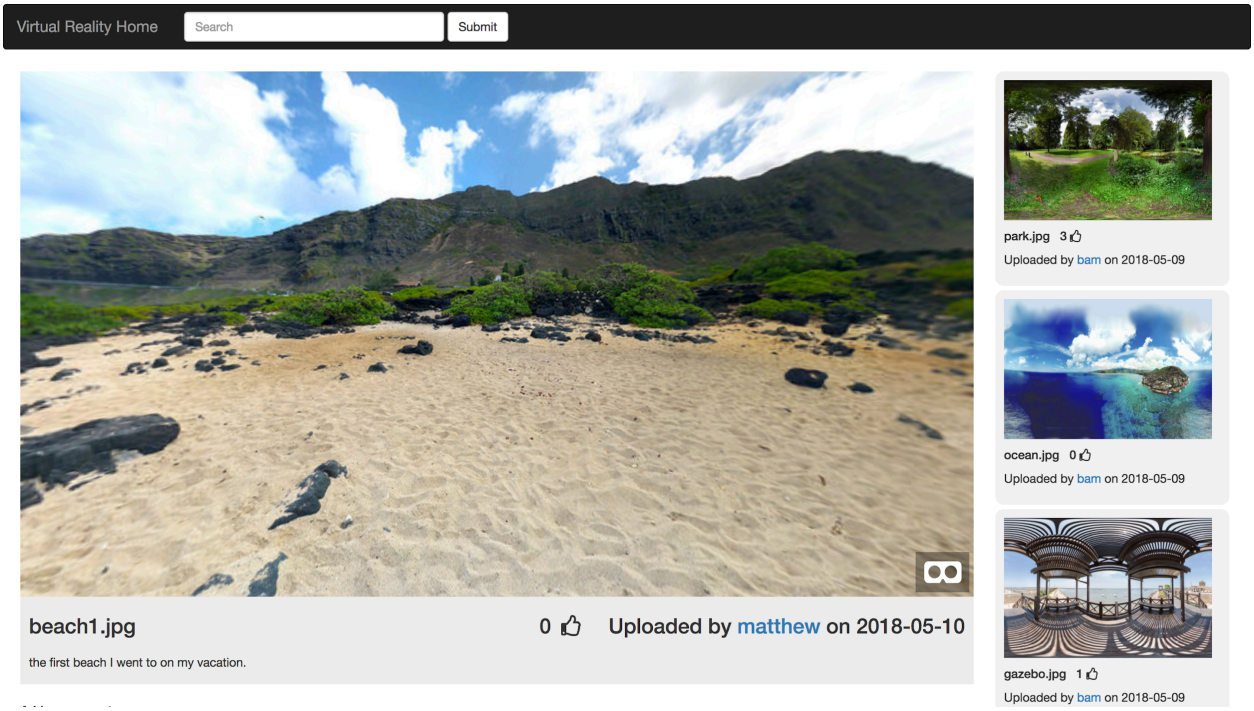

Figure 22. VR details

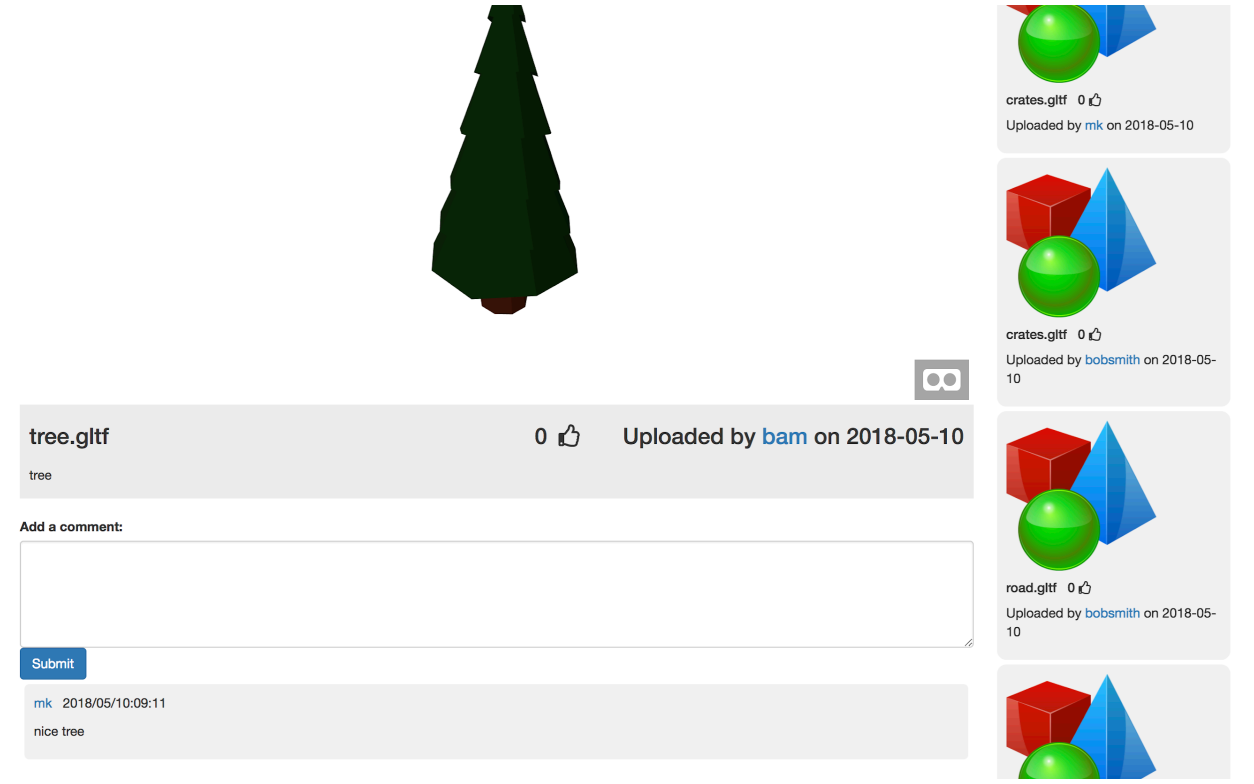

Figure 23. Adding a comment to a VR

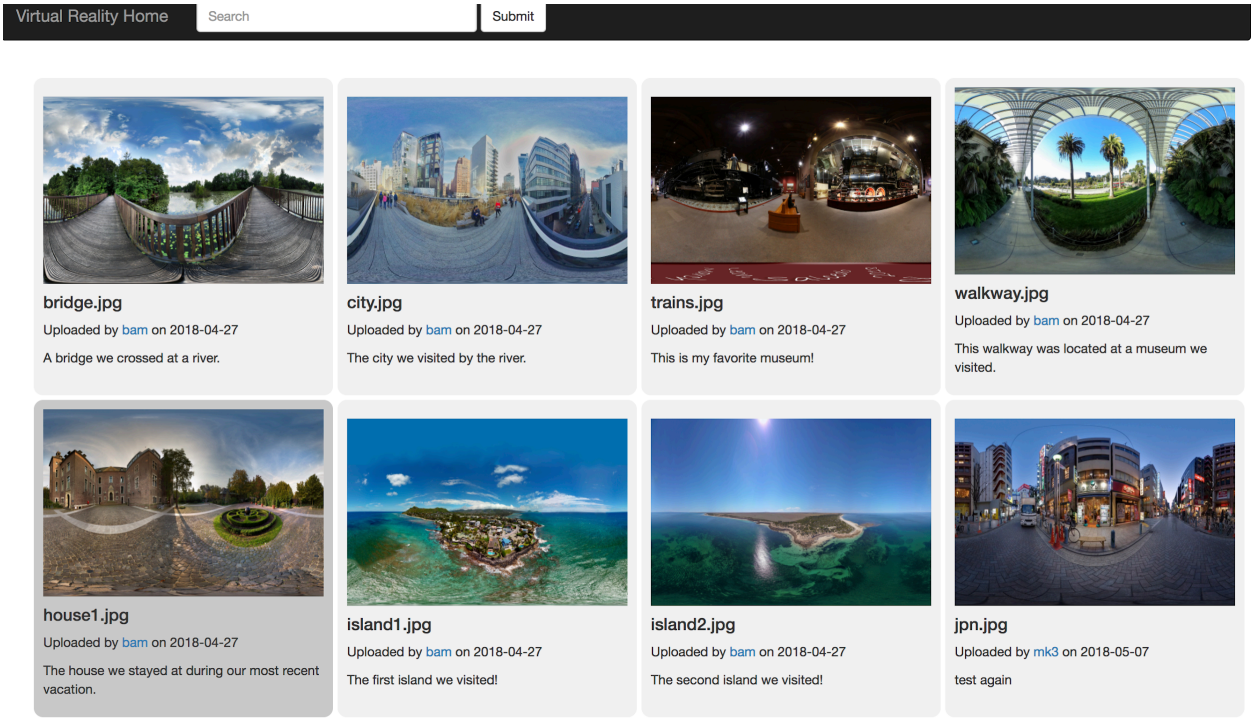

Figure 24. Search results page