Seedit Design Document

Version 1.0

Table of Contents

Design Document Version History ….……..……..…….. 2
Summary ….....……..……..……..……...2
UI Mockups ...……..……..……..…….. 3
One Page Frame ...……...……...……...…….. 3
Home Page ……..…….....……...……..... 4
User Profile ……..…….....……...……..…… 5
Database Model …….. 5
Database Schematic ……... 7
Database Create Statements ..…….. 5
API Calls from Frontend to Backend …….. 9

Design Document Version History

Version	Date	Description	Author
1.0	12/14/2014	Initial Draft	The Seedit Team

Summary

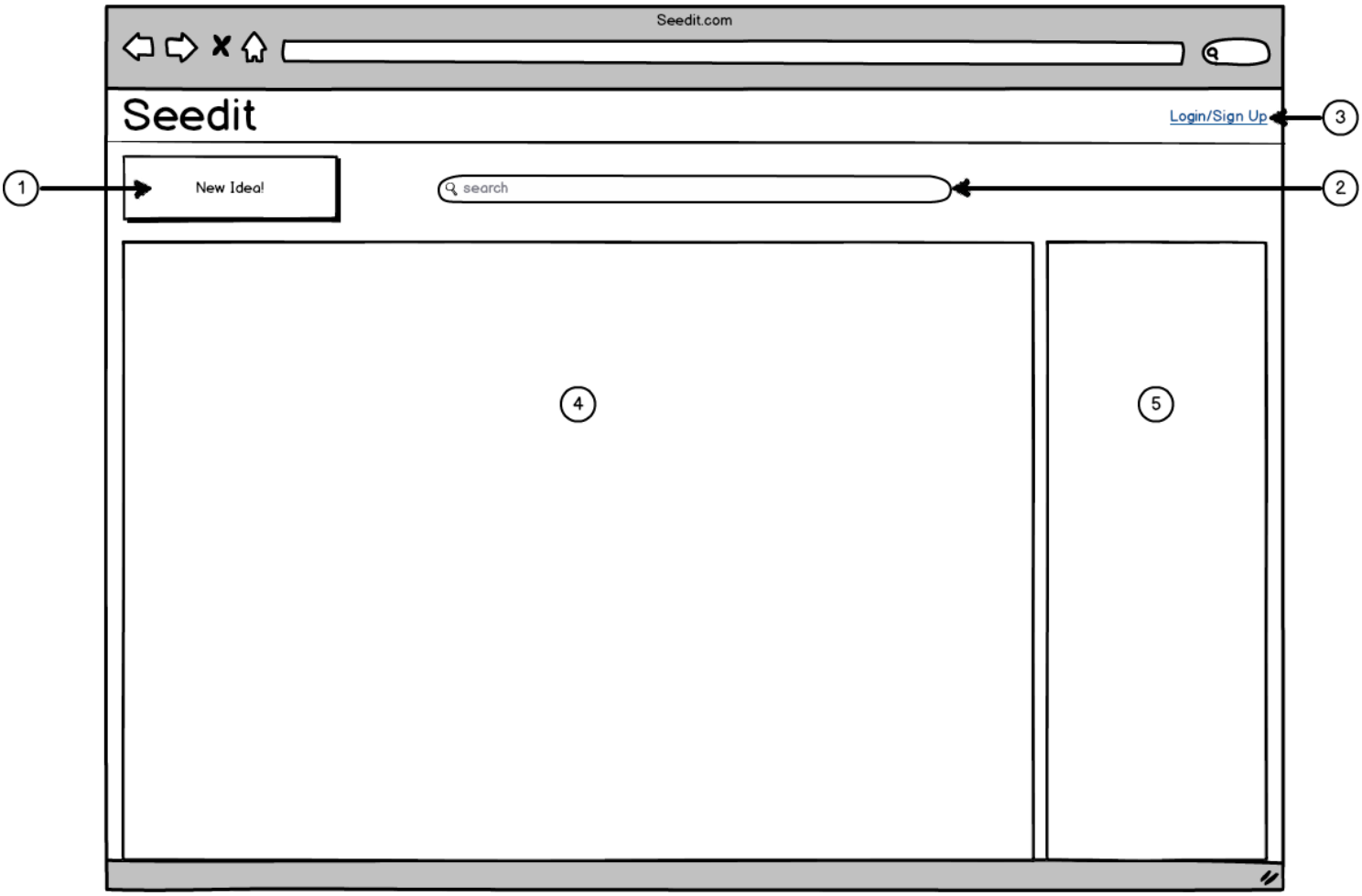
This design document has three points of focus: User Interface Mockups, a Database Model, and the Seedit API. Seedit will be a one page web application that will use AngularJS to dynamically update the page with needed information. Our MVP will have a Home Page and a User Profile so the mockup section first describes the outline of our one page application and then outlines the two main pages.

The backend of Seedit uses Firebase for data synchronization and a NodeJS server which communicates with the client, firebase, and SQL. The Database Model consists of a UML Database Schematic that describes the relationship of the database tables. The Database Model section also defines our SQL Create Table statements.

As for the Seedit API, the last section of this document lays out every action that can be executed from the homepage of seedit and defines what will be sent to the server and received from the server. These queries and responses define the API and will be used as a guide when building the frontend and backend in parallel to ensure a smooth integration of components.

UI Mockups

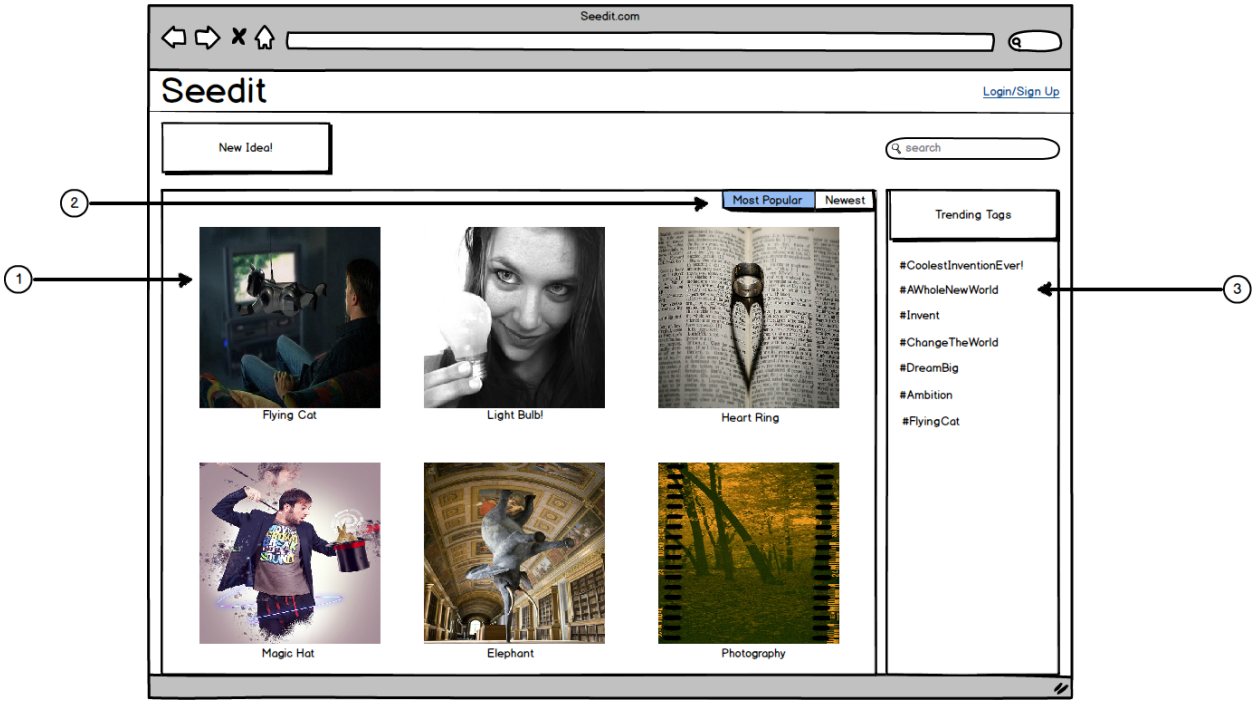
One Page Frame:



The SeedIt webpage will follow a consistent design with portions being switched out based on what a user is doing. This is the frame for the whole site.

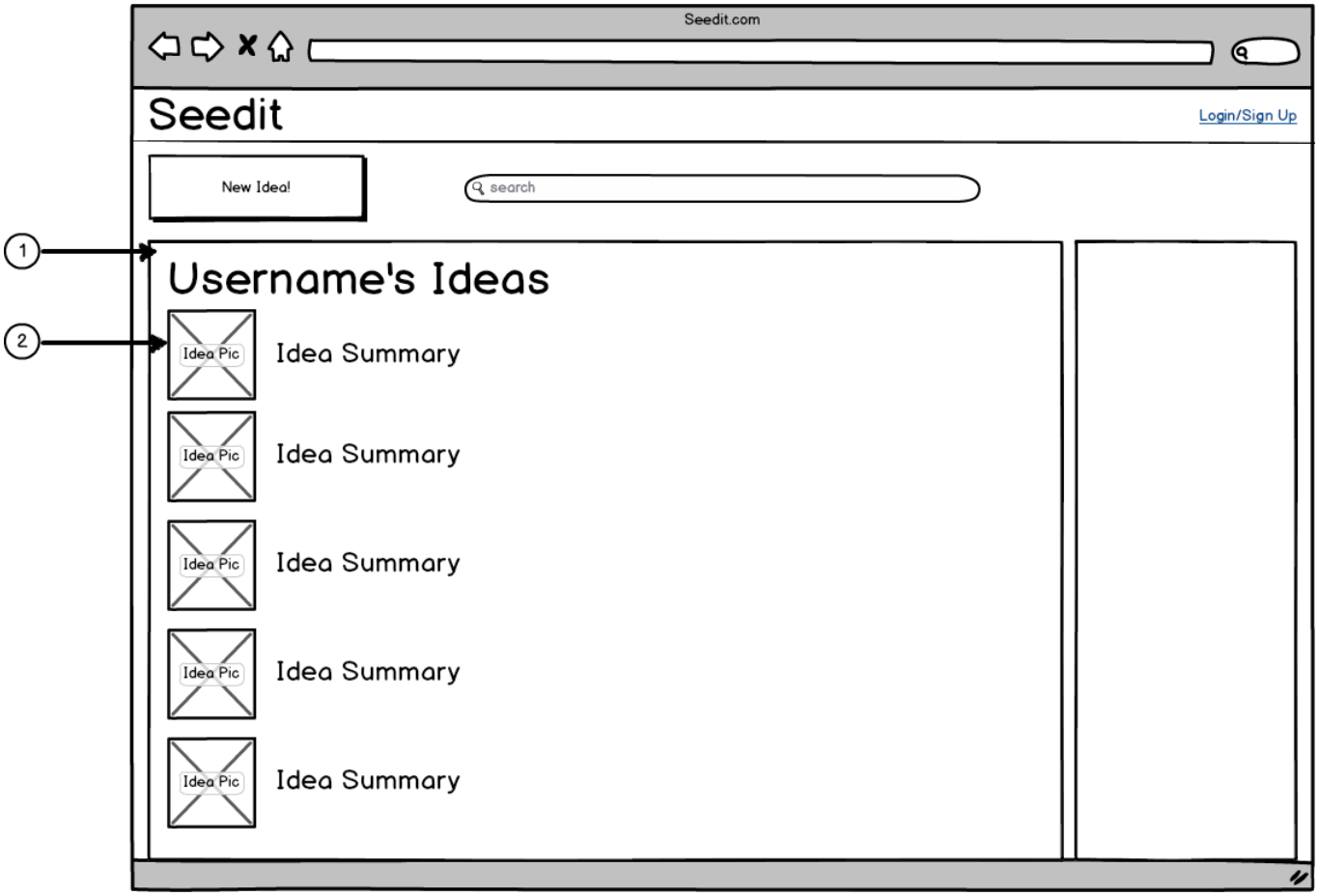
1. A persistent button for users to quickly get to the New Idea Submission page.
2. A search bar for searching different ideas based on their tags.
3. A login and signup link for users who want to create ideas and upvote other's ideas.
4. Main view area. This will be switched out with different content depending on where in the site a user is.
5. Side view area. This will be switched out with content such as Similar Ideas and Trending Tags.

Home Page:



1. A list of ideas displayed by image tiles and idea names. Can be sorted to show the “Most Popular” ideas or the “Newest” ideas.
2. Button bar to toggle between showing the “Most Popular” ideas or the “Newest” ideas.
3. List of the top trending tags associated with ideas. Clickable to view ideas with that tag.

User Profile:



This page is associated with each user and can be accessed by clicking on their name on the Idea page, or if it is the logged on user, by clicking on their name in the upper right corner.

1.This page is contained in the main viewing area like others.
2. It contains a time sorted list of ideas, showing their picture and main summary.

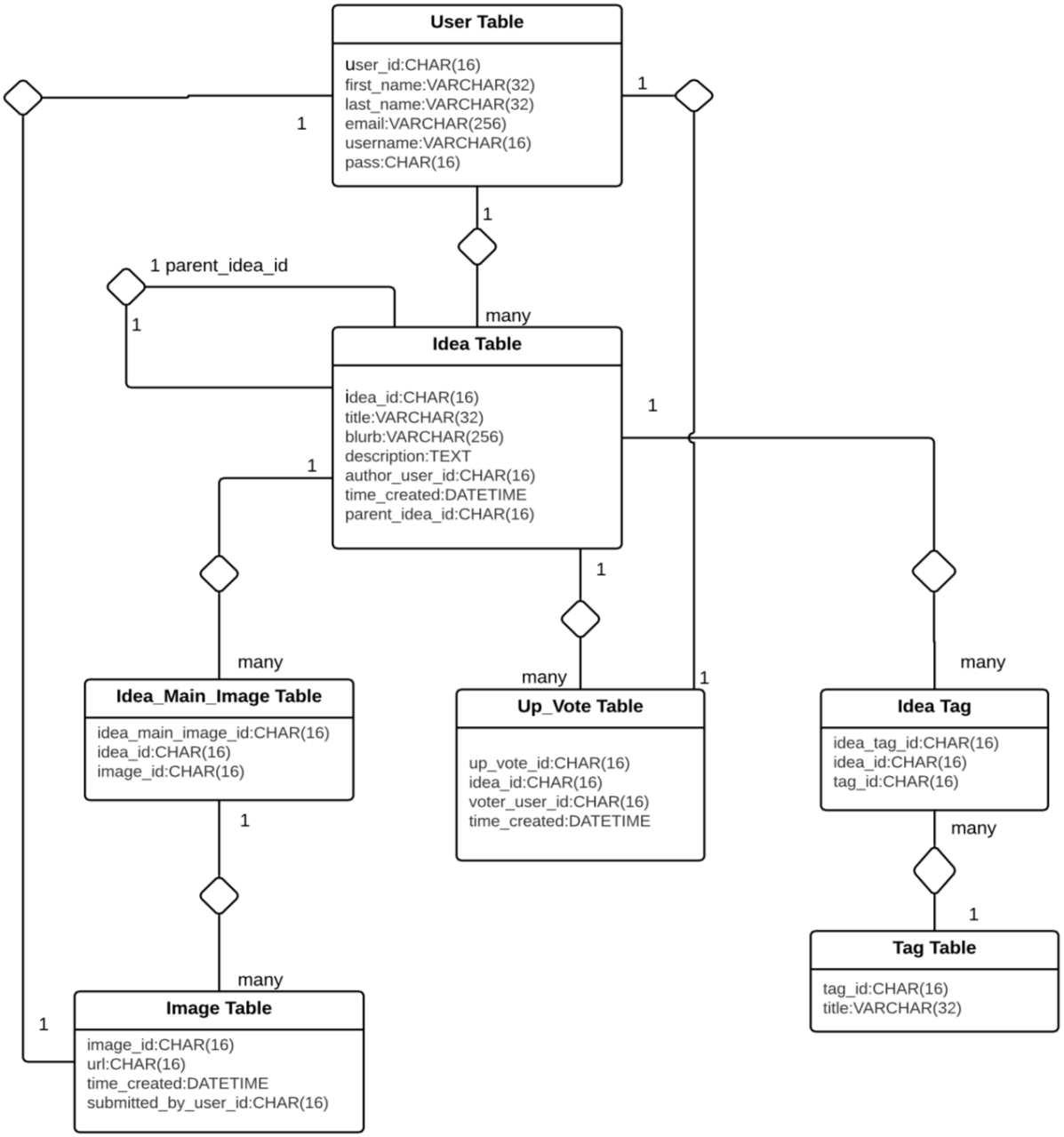
Database Model

The SeedIt database model will consist of a Node.js infrastructure communicating to an SQL database. All endpoints will be handled by express and handed down to Node.js. All responses will be handed back using express as well.

Database schematic:

The SeedIt database will have relationships defined as the following:

1. An idea_id in the idea table will be owned by one user_id.
2. An idea_id in the idea table will have a link to a parent_idea_id.
3. A idea_main_image_id in the idea_Main_Image table belongs to an idea_id.
4. An up_vote_id in the Up_Vote table belongs to a idea_id.
5. An up_vote_id in the Up_Vote table is owned by a user_id.
6. An idea_tag_id from the Idea tag table belongs to an idea_id.
7. An tag_id from the Tag Table belongs to an idea_tag_id.
8. An image_id from the Image Table belongs to and idea_main_image_id.
9. And image_id from the Image Table is owned by a user_id.

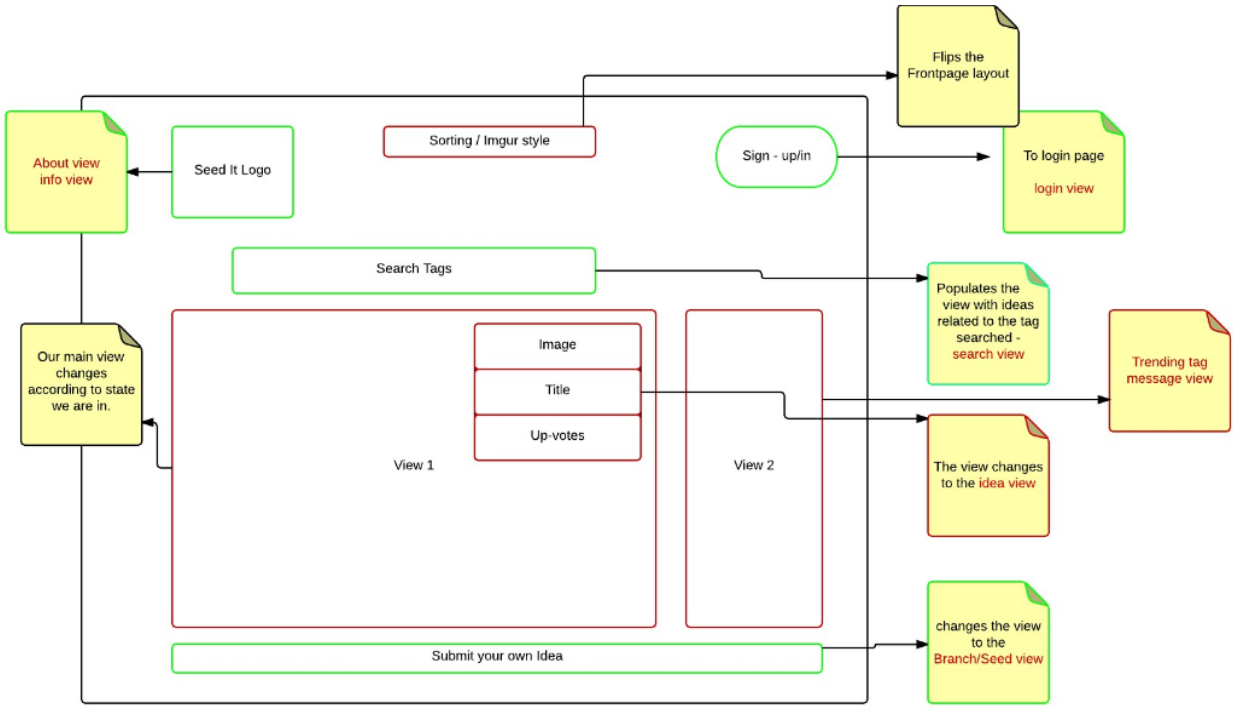


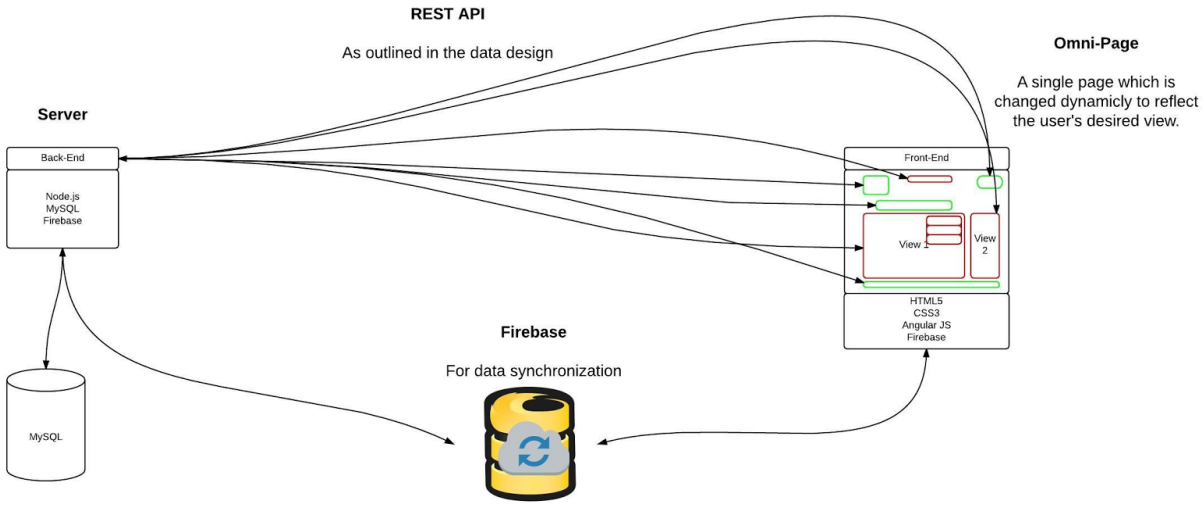
Database Create Statements:

```
DROP DATABASE SeedIt;

CREATE DATABASE SeedIt;

USE SeedIt;

CREATE TABLE User (
  user_id CHAR(16) UNIQUE NOT NULL,
  first_name VARCHAR(32) NOT NULL,
  last_name VARCHAR(32) NOT NULL,
  email VARCHAR(256) UNIQUE NOT NULL,
  username VARCHAR(16) UNIQUE NOT NULL,
  pass CHAR(60) NOT NULL,
  PRIMARY KEY(user_id)
);

CREATE TABLE Idea (
  idea_id CHAR(16) UNIQUE NOT NULL,
  title VARCHAR(32) NOT NULL,
  blurb VARCHAR(256) NOT NULL,
  description TEXT NOT NULL,
  author_user_id CHAR(16) UNIQUE NOT NULL,
  time_created DATETIME NOT NULL,
  parent_idea_id CHAR(16) NULL,
  PRIMARY KEY(idea_id),
  FOREIGN KEY(author_user_id) REFERENCES User(user_id),
  FOREIGN KEY(parent_idea_id) REFERENCES Idea(idea_id)
);

CREATE TABLE Up_Vote (
  up_vote_id CHAR(16) UNIQUE NOT NULL,
  idea_id CHAR(16) NOT NULL,
  voter_user_id CHAR(16) NOT NULL,
  time_created DATETIME NOT NULL,
  PRIMARY KEY(up_vote_id),
  FOREIGN KEY(idea_id) REFERENCES Idea(idea_id),
  FOREIGN KEY(voter_user_id) REFERENCES User(user_id),
  UNIQUE(idea_id, voter_user_id)
);

CREATE TABLE Tag (
```

```
  tag_id CHAR(16) UNIQUE NOT NULL,
  title VARCHAR(32) UNIQUE NOT NULL,
  PRIMARY KEY(tag_id)
);

CREATE TABLE Idea_Tag(
  idea_tag_id CHAR(16) UNIQUE NOT NULL,
  idea_id CHAR(16) UNIQUE NOT NULL,
  tag_id CHAR(16) UNIQUE NOT NULL,
  PRIMARY KEY(idea_tag_id),
  FOREIGN KEY(idea_id) REFERENCES Idea(idea_id),
  FOREIGN KEY(tag_id) REFERENCES Tag(tag_id),
  UNIQUE(idea_id, tag_id)
);

CREATE TABLE Image(
  image_id CHAR(16) UNIQUE NOT NULL,
  url CHAR(16) NOT NULL,
  time_created DATETIME NOT NULL,
  submitted_by_user_id CHAR(16) NOT NULL,
  PRIMARY KEY(image_id),
  FOREIGN KEY(submitted_by_user_id) REFERENCES User(user_id)
);

CREATE TABLE Idea_Main_Image(
  idea_main_image_id CHAR(16) UNIQUE NOT NULL,
  idea_id CHAR(16) UNIQUE NOT NULL,
  image_id CHAR(16) NOT NULL,
  PRIMARY KEY(idea_main_image_id),
  FOREIGN KEY(idea_id) REFERENCES Idea(idea_id),
  FOREIGN KEY(image_id) REFERENCES Image(image_id)
);
```

API Calls from Frontend to Backend





The front end is loaded for the user in one HTML page. The rest of the information that the server will send to the user will be done through JSON documents and not actual HTML. These JSON documents are sent from the server to the user when the front-end makes API calls to the server. This will make the page much more responsive than a traditional website that sends different HTML pages.

The following paragraphs describe what API calls our main page will make to our backend server. The headers of each paragraph indicate the component in the above view that is making the API call.

Component #1:
Component #1 as outlined in the home page diagram will make the following API calls when triggered by the following events:

1. **Whenever the page loads and "Most Popular" is selected:** Request the top 50 most popular ideas on the server. Each idea will be represented as an entry with its description, its number of upvotes, and the size of its idea tree.
2. **Whenever the user scrolls to the bottom of the list:** Request the next top 50 most popular/newest ideas from the backend with the same JSON format as indicated above
3. **Whenever the user clicks on "Newest":** Request the top 50 newest ideas on the server. Each idea will be represented as as the same JSON format described above
4. **When an IDEA is clicked on:** Request a JSON document containing the idea object from the server. This will include the description, content, number of upvotes, a list of direct children and their JSON urls. The IDEA view will then open up.
5. **When the seed it logo is pressed:** No data will be required. The page will save the data for this view.
6. **During the sign-in when the user submits Info:** We will validate the data input (user name and password) on the frontend to make sure it is in the correct form. Then the data will be sent to the backend to be authorized. The application will get back an acknowledgement with users basic information.
7. **When a search query is submitted:** The query string will be sent to the backend and a Json object will be sent back with all of the relevant branches data which have tags similar to the query string. The view will then display the results in a similar form to the trending ideas page which is loaded upon initialization.

Component #3:
Component #3 as outlined in the diagram will make the following API calls when triggered by the following events:

1. **Whenever the page loads:** Request the top 20 trending tags in JSON format. The JSON document will have a list of tags sorted from most popular to least popular, along with their number of occurrences.
2. **Whenever the user scrolls down to the bottom of the view:** Request the next top 20 trending tags in JSON format. The format will be the same format as described above.

IDEA view:

1. **Whenever a user clicks the “add branch” button:** After validating the input fields, the front-end will send a POST request to the server with its fields containing information for the IDEA object to be saved to the server. This would include the description and the content of the idea.