

Indie Market

Design Plan

By:

Paul Minniti

Carolina Zapata

Gabriel Ngankam Satchouk

Manuel Vargas

Christopher Hearl

James Waltz

11 Design Plan

11.1 Objective

11.2 Technologies

11.2.1 Languages

11.2.2 Node.js

11.2.3 Electron

11.2.4 Amazon Web Services

11.2.5 Google Identity Platform

11.2.6 Bootstrap Framework

11.2.7 MongoDB

11.2.8 Mongoose ORM

11.3 System Architecture

11.3.1 User to Application

11.3.2 Application to Database

11.3.3 Application to Media Files

11.4 User Interface

11.4.1 Login Page

11.4.2 Menu

11.4.3 Store

11.4.4 Library

11.4.5 Profile

11.4.6 Friends

11.4.7 Settings

11.5 Future Updates

11.5.1 Moving the Media Files to a Server

11.5.2 Implementing E-Commerce

11 Design Plan

11.1 Objective

The purpose of this section is to give an outline of the tools being used to build the program as well as outline the interface. The Developers will reference this section while building the Indie Market platform.

11.2 Technologies

11.2.1 Languages

- HTML/CSS
- JavaScript / JQuery
- JSON

11.2.2 Node.js

Node.js is an asynchronous event driven JavaScript runtime. Node will be used to make calls to the database and build out the backend of our application. This will allow us to push updates, package our platform and access the local file system. Additionally, we will be using node package manager to install our modules like electron.

11.2.3 Electron

Electron is a web API that can be integrated into Node.js. Electron lets you build cross platform desktop apps with web languages. In a way it's a minimal web browser that lets us talk to the local file system. It does this by creating an executable chromium window, giving users the desktop experience while allowing us to use the cross platform capabilities of internet browsers. In our case we are using electron solely for that purpose.

11.2.4 Amazon Web Services

Amazon Web Services (AWS) is a cloud based platform offering almost every service an online business requires. Our AWS cloud server will implement MongoDB for our database in order to set permissions for content that users purchase through our platform.

11.2.5 Google Sign In

Google Sign In or Google Identity Platform is an authentication system for web technologies. It can be easily implemented into our program to allow secure connections for our users. This will allow us to focus on the main functionality of our platform without having to worry about the security of our users.

Documentation Link: <https://developers.google.com/identity/>

11.2.6 Bootstrap Framework

Bootstrap Framework is a front-end component library and is compatible with all web languages. It allows developers to create responsive and crisp web Interfaces. We will be using Bootstrap for the front end styling of our user interface.

11.2.7 MongoDB

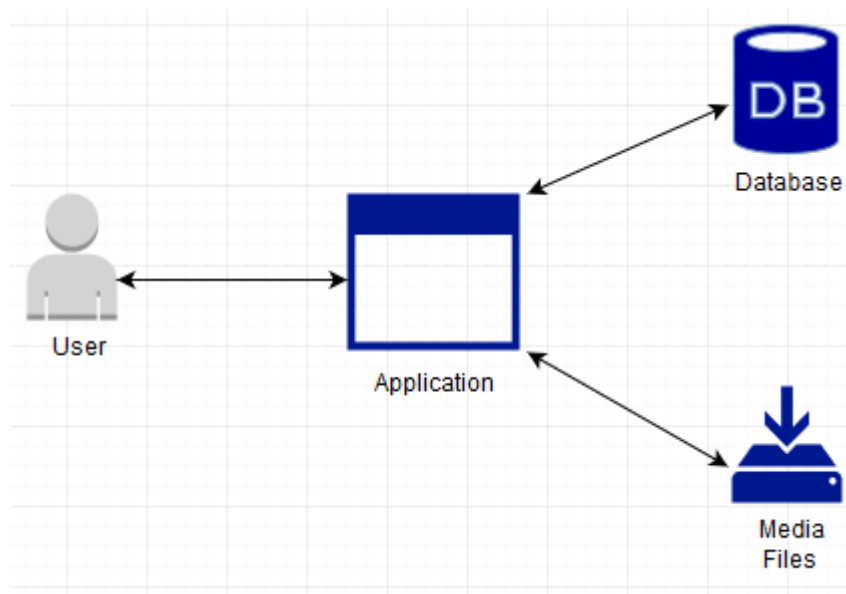
Mongo Database is an open source document-oriented database. It adapts JSON documents to have dynamic schemas. We will be implementing MongoDB as our database.

11.2.8 Mongoose ORM

Mongoose ORM will be used in conjunction with MongoDB. Mongoose allows us to have built in automatic data validation and pre-defined events.

11.3 System Architecture

The Architecture of the system is based on a Model View Controller Framework (MVC). The Model would be represented by the Database which controls users access to the media files. The view of the system is going to be the front end or interface of the platform. The controller is the backend of the application which will be making calls to the database to give or check if a user has access to the file they are trying to interact with. Everything together will make up the core components of the application.



11.3.1 User to Application

This interaction will be the view of the system and will be responsible for supplying the user with access to the application interface. The application will be an executable file that has been downloaded onto the user's desktop.

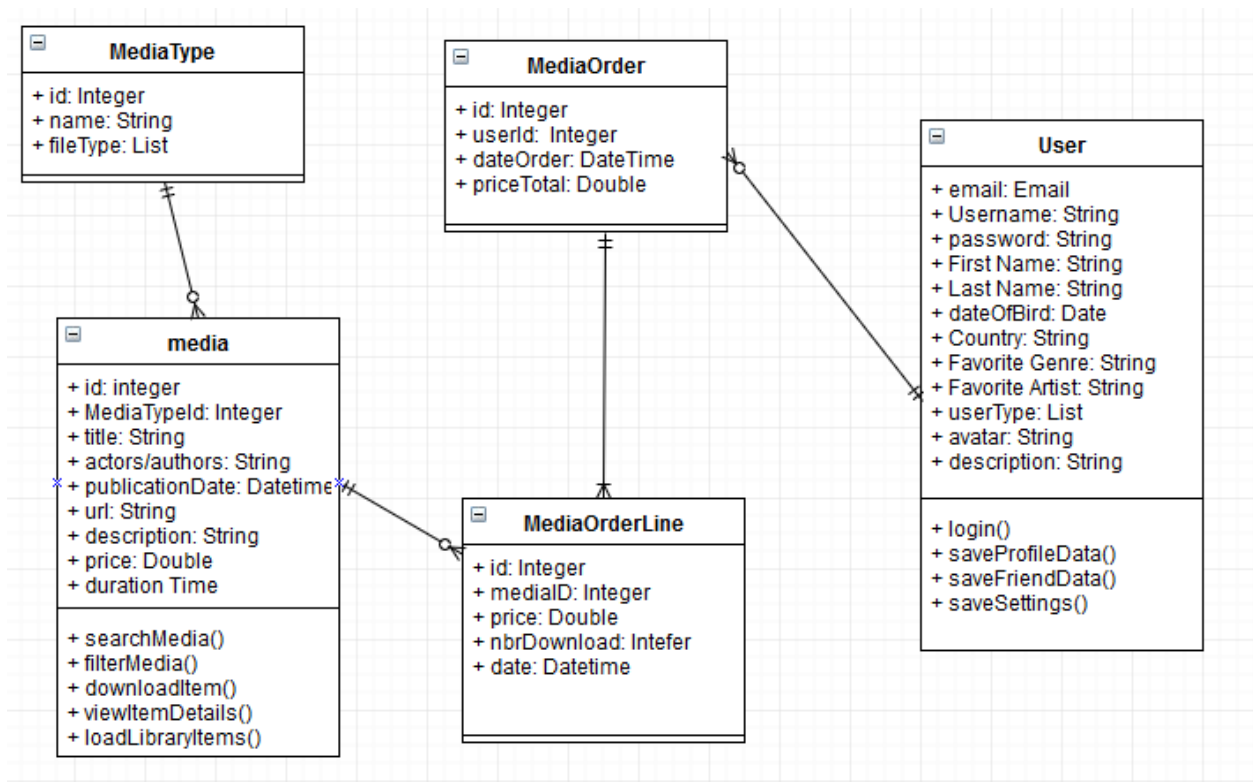
11.3.2 Application to Database

The interaction Between the Application and database is where all permission requests will be handled. If a user would like to add a media file to their library, a call to the database will be made and a check will be run to ensure the necessary prerequisites are met.

11.3.3 Application to Media Files

The Media files will be stored within the application and users will only be able to view them after they have added them to their library.

11.3.4 Application UML



11.4 User Interface

11.4.1 Login Page

Welcome to Indie Market

Login

Email

Sign In With Google

Password

Login

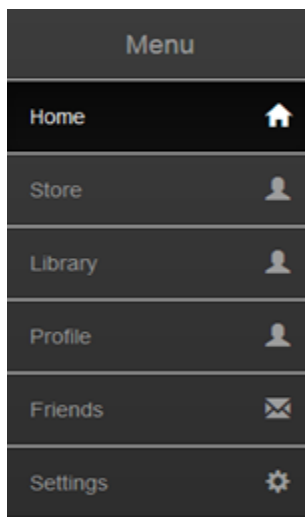
Register

The Login page will be the first page user are brought to when executing the program. The login page has four main elements, email and password fields as well as a login and register buttons. The login page will also implement google sign in or google identity platform for user login authentication. Below is an example of google Sign in's API from their website.

```
<div class="g-signin2" data-onsuccess="onSignIn" data-theme="dark"></div>
<script>
  function onSignIn(googleUser) {
    // Useful data for your client-side scripts:
    var profile = googleUser.getBasicProfile();
    console.log("ID: " + profile.getId()); // Don't send this directly to your server!
    console.log('Full Name: ' + profile.getName());
    console.log('Given Name: ' + profile.getGivenName());
    console.log('Family Name: ' + profile.getFamilyName());
    console.log("Image URL: " + profile.getImageUrl());
    console.log("Email: " + profile.getEmail());

    // The ID token you need to pass to your backend:
    var id_token = googleUser.getAuthResponse().id_token;
    console.log("ID Token: " + id_token);
  };
</script>
```

11.4.2 Menu



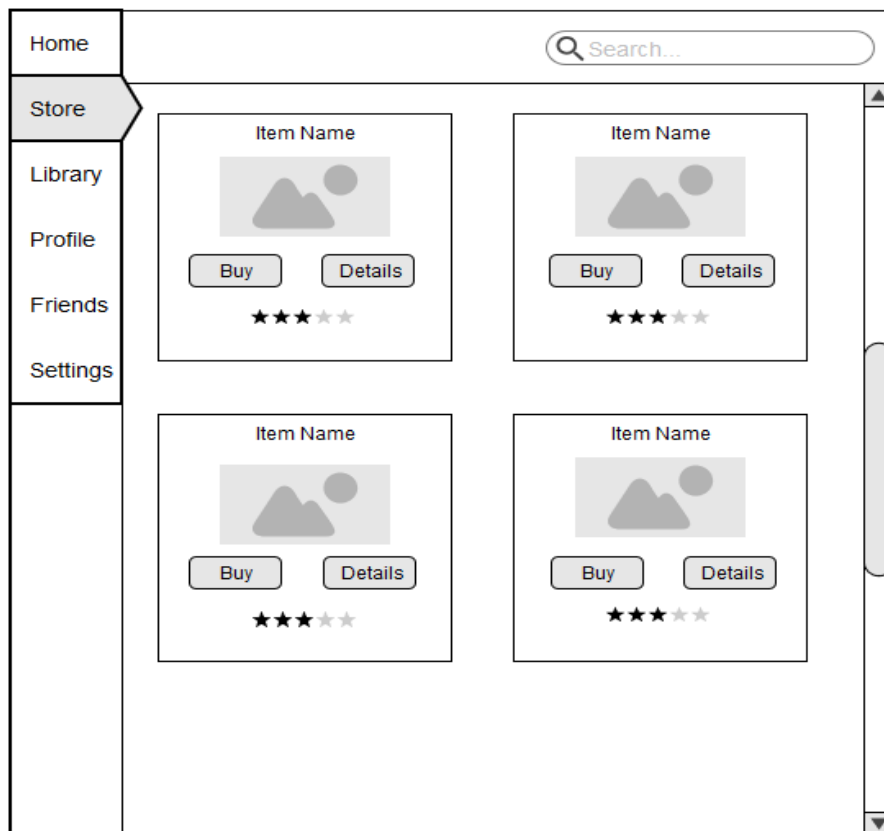
The Menu is how users will navigate the application in order to access all of its functionality. After logging in the default page will be the

home page. In order to improve performance, the content will be loaded in one HTML file and will be hidden and displayed using JQuery scripts.

Ex.

```
<script>
  $(document).ready(function(){
    $("element").click(function(){
      $("this").hide();
      $(".newelement").show();
    });
  });
</script>
```

11.4.3 Store



The Store is where users will be able to browse and add media content to their library. Each item will display its name and image of the content as well as buttons to add to a user's library and view the details of the content. Search functionality will be implemented to allow users to search for a specific piece of content. In order to control what users have

access to, database calls will be made when a user wants to add content to their library. Below is an example of a database call using Node.JS.

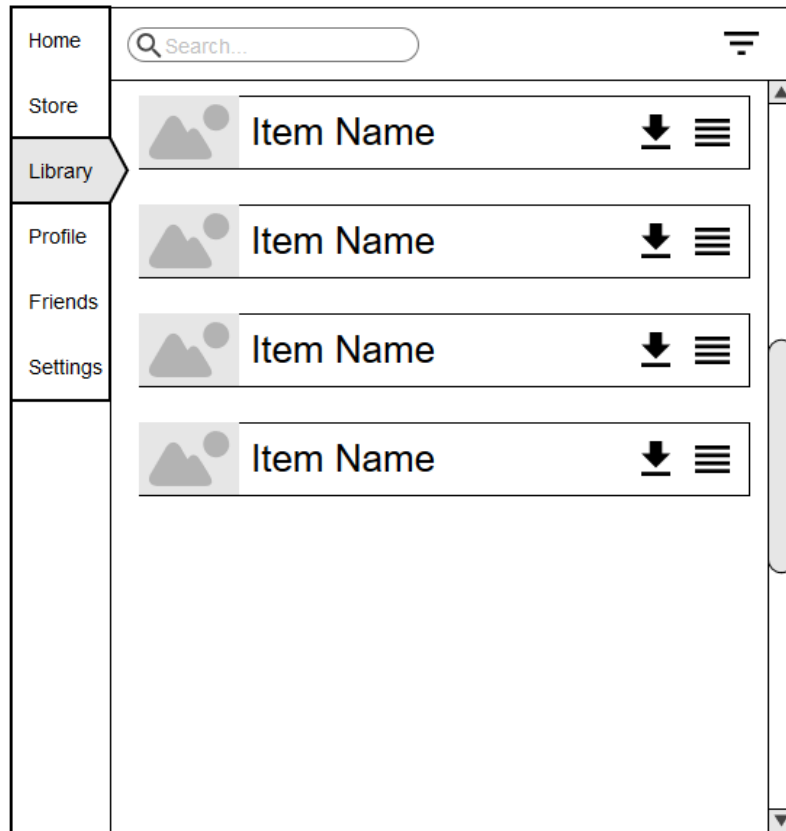
```
function callDatabase(id) {
  var result;
  var connection = mysql.createConnection(
    {
      host      : 'localhost',
      user      : 'username',
      password  : 'password',
      database  : 'myDB'
    }
  );

  connection.connect();
  var queryString = 'SELECT name FROM test WHERE id = 1';

  connection.query(queryString, function(err, rows, fields) {
    if (err) throw err;

    for (var i in rows) {
      result = rows[i].name;
    }
  });
  connection.end();
  return result;
}
```

11.4.4 Library



The library is where users can access the content that they have added to their library through the store. The functionality on this page will allow users to: download, view details, filter and search for content in the user's library. Items in the library will be loaded based upon what permissions are set for the user in the database. An example of the search function is below.

```

<input type="text" id="search" onkeyup="SearchFunction()" placeholder="Type here to Search... ">

<script>
function SearchFunction() {


    var input, filter, ul, li, a, i;
    input = document.getElementById('search');
    filter = input.value.toUpperCase();
    ul = document.getElementById("myUL");
    li = ul.getElementsByTagName('li');

    // Loop through all list items, and hide those who don't match the search query
    for (i = 0; i < li.length; i++) {
        a = li[i].getElementsByTagName("a")[0];
        if (a.innerHTML.toUpperCase().indexOf(filter) > -1) {
            li[i].style.display = "";
        } else {
            li[i].style.display = "none";
        }
    }
}
}
</script>


```

11.4.5 Profiles

Home
Store
Library
Profile
Friends
Settings



First Name:
Last Name:
Age: ▼
Country: ▼
Favorite Genre: ▼
Favorite Artist:



The profile tab is where users will be able to add information about themselves to the system. The Information they input in the fields will be passed to the database. The data will be used to obtain information on our users and their preferred media content, this information will be used when considering what media content to add to the platform in the future. An example of how the information will be passed to the database is below

```
function InsertDatabase(id) {
  var result;
  var connection = mysql.createConnection(
    {
      host      : 'localhost',
      user      : 'username',
      password  : 'password',
      database  : 'myDB'
    }
  );

  connection.connect(function(err) {
    if (err) throw err;
    console.log("Connected!");
    var sql = "INSERT INTO customers (name, address) VALUES ('Company Inc', 'Highway 37')";
    connection.query(sql, function (err, result) {
      if (err) throw err;
      console.log("1 record inserted");
    });
  });
}
```

11.4.6 Friends

Home

Store

Library

Profile

Friends

Settings

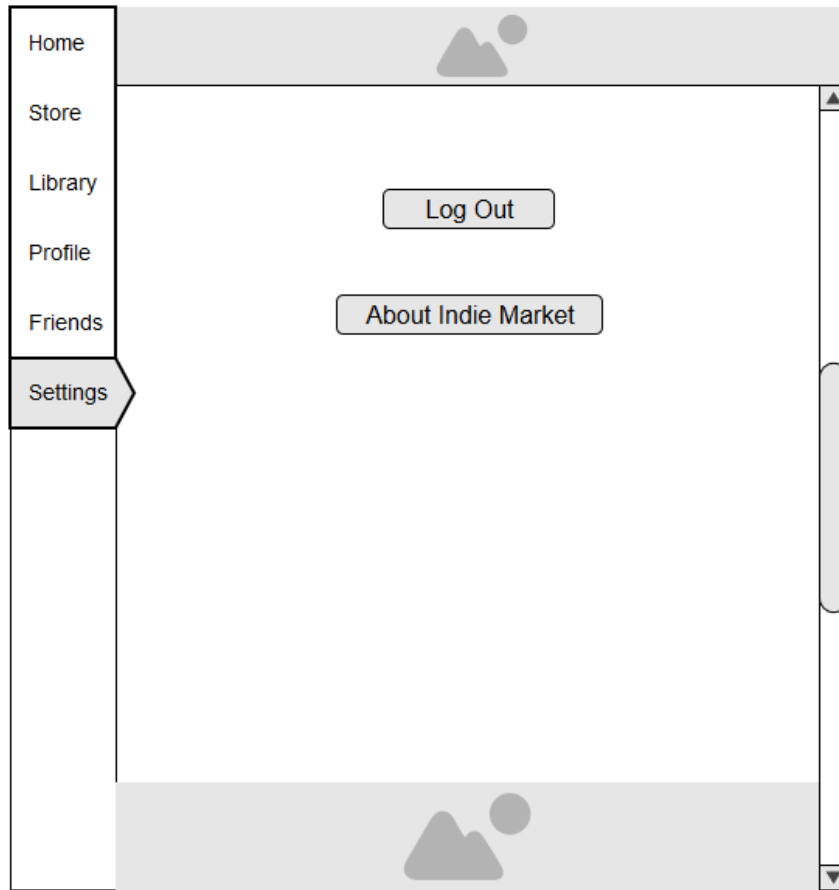
Bio

Connect Social Accounts ▼

Save

The friends tab will allow the user to upload a photo of themselves, add a short Bio and connect their other social media accounts. This information is again passed to the database to store information on our users and what accounts they have through various social network platforms. The program will not be able to access their connected accounts, this functionality is just used to gain insight on our customers.

11.4.7 Settings



The Settings tab has the least amount of functionality but is still critical to the design of our program. Users will use this page to call google sign in's API in order to log the m out of the system. Users will also be able to learn more about the project in a pop up after clicking the About Indie Market button.

11.5 Future Updates

11.6.1 Moving the Media Files to a server

In order to expand upon the systems, proof of concept and move it towards a deliverable product, the Media Files will have to be moved out of the program files and onto their own server. This will allow Indie Market to deliver a larger option of content without hindering the performance of the system as it grows.

11.6.2 Implementing E-Commerce

The Indie Market store would need an E-Commerce framework or API to be implemented in order to handle transactions. This would allow Indie Market to monetize itself by selling licensed content. This would allow Indie Market to grow its team and expand the platform with more updates and content.