CPEN321
Media Recommendation Engine

Design and Architecture Guidelines

Introduction

Our goal is to build our Media Recommendation Engine as a web application. The engine will seek to offer media suggestions to consumers based on their search queries, search history, and personal profile.

Architecture and Rationale

The application consists of the following major components:

1. The MEAN stack (MongoDB, Express, AngularJS, Node.js) to create the front and back ends of the web application
2. Several APIs from sites such as IMDb, Rotten Tomatoes, and Twitter along with a web crawler to gather information about media

MongoDB

The MEAN stack integrates MongoDB for the NoSQL database. MongoDB is an open-source, non-relational SQL database which we will use to store our media information and user data. The document store structure utilized by MongoDB is efficient at handling large amounts of data, which fits well for our case for media information.

Express.js

Express.js is a module framework for Node.js which simplifies development of secure, modular and fast applications. We are using this technology as a layer of abstraction on top of Node.js in order to more specialize in and make writing our web server easier by reduce the amount of code that must be rewritten or reimplemented.

AngularJS

AngularJS is a JavaScript framework that provides additional functionality used to develop dynamic web applications. Using AngularJS allows for extending existing HTML elements into more advanced web components. AngularJS implements a model-view-controller framework that describes the interactions between background web components and how they interface with the user. Using AngularJS will allow us to efficiently add new features or components to our web application.

Node.js

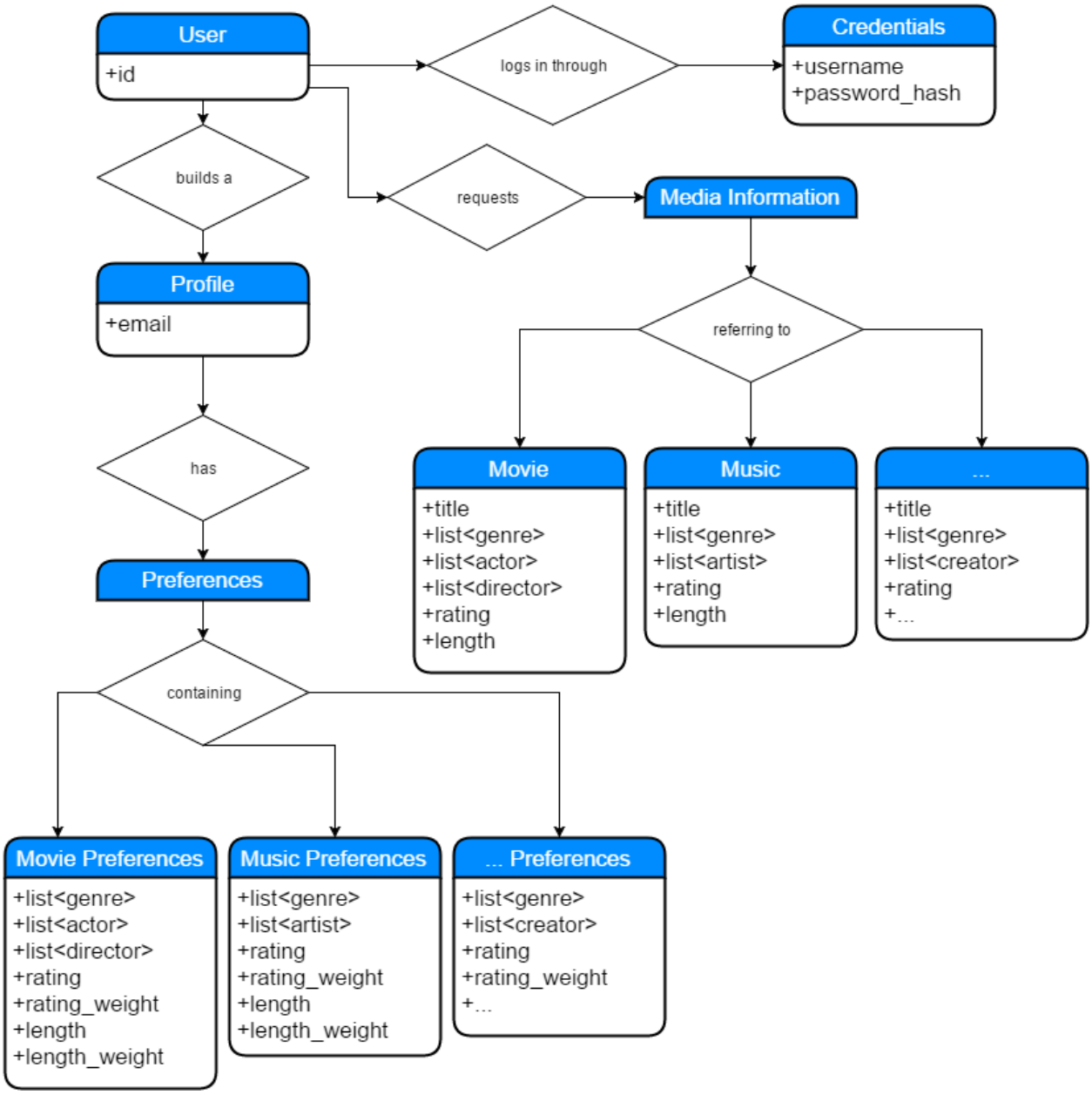
Node.js uses event driven, non-blocking I/O to be able to handle data-intensive applications while remaining lightweight. Since we are not asking for particularly CPU intensive tasks, we found that Node.js allows us the scalability that we need for our distributed web application. This technology enables us to use a client-server architecture which allows us to manage security,

data, and back-ups at one location. This also allows us to make upgrades and changes as necessary on the server without increasing resource use on the client-side.

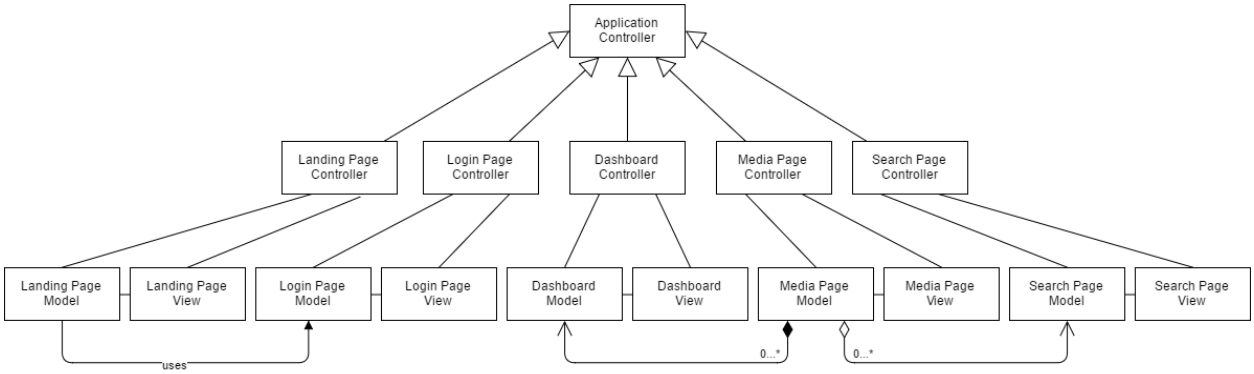
API Accesses and Web Crawler

In order to provide users with media recommendations and suggestions, a database containing media will be needed for drawing information from. To collect this data, we will use APIs from various sites such as IMDb and Rotten Tomatoes and call for the data when requested.

Data

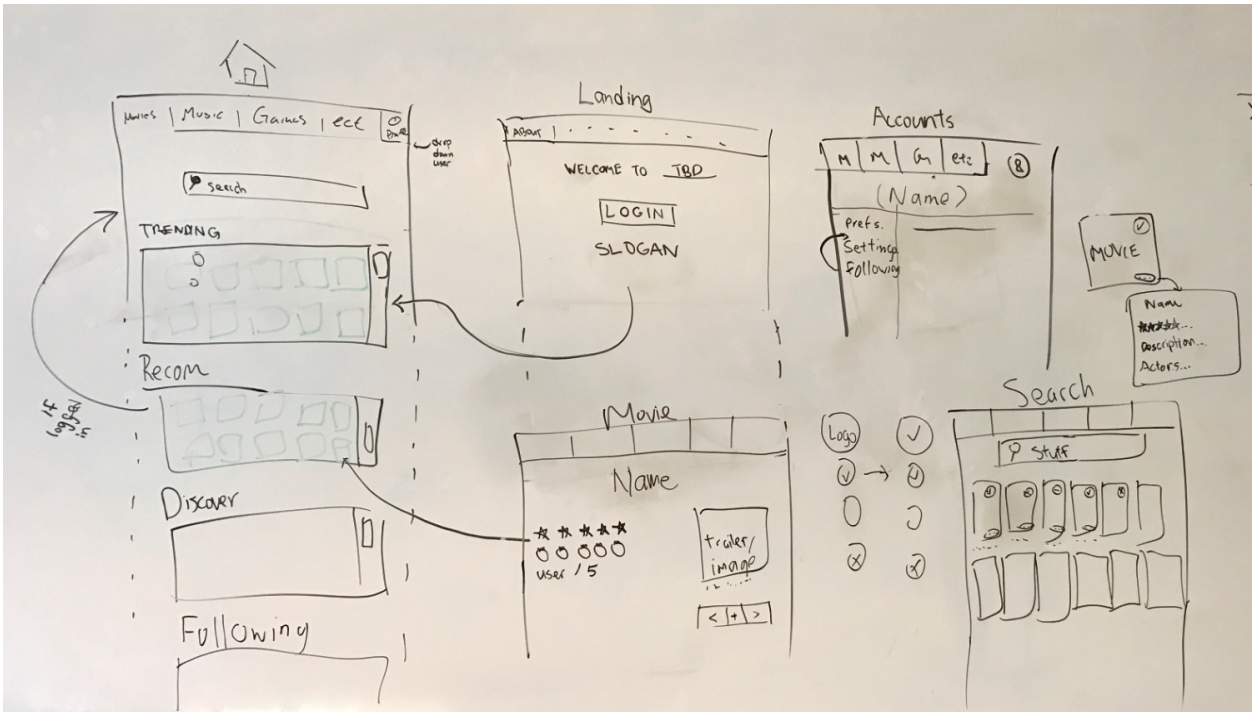


Detailed Design



GUI

We plan on constructing the web application from scratch (without the use of templates). This will be built using HTML, CSS (with Bootstrap), and JavaScript frameworks such as AngularJS and Express. A rough sketch of what our web application may look like is shown below.



Validation

The UI validation will be continuously worked on and improved on; using feedback from beta testers, we can decide on new features and fixes to implement. We will also be running tests on our design, ensuring the flow is succinct to users of varying computer knowledge.