ITMD 565 Rich Internet Applications
Project 2
Mahak Patil
A20323104

Project Objective:

Create a full NodeJS Express CRUD application with JSON API (2 Data models minimum, Data persistence stored in JSON files on the filesystem, RESTful url paths)

To run this project:

After unzipping it, open command line and execute these commands:
npm install
npm install supervisor
npm install method-override
npm install underscore
npm start

This application keeps a record of popular animated movies and dolls associated with them. User can create, read, update and delete a doll or a movie. This application is a full NodeJS CRUD application. The two models used here are dolls and movies. The 'dolls' object contains name, eyes, hair and clothes fields and the 'movies' object contains name of doll, name of movie, year of release and production house fields. They also contain a unique id field.
The home page has two options the user can choose: Disney Princesses or Movies.

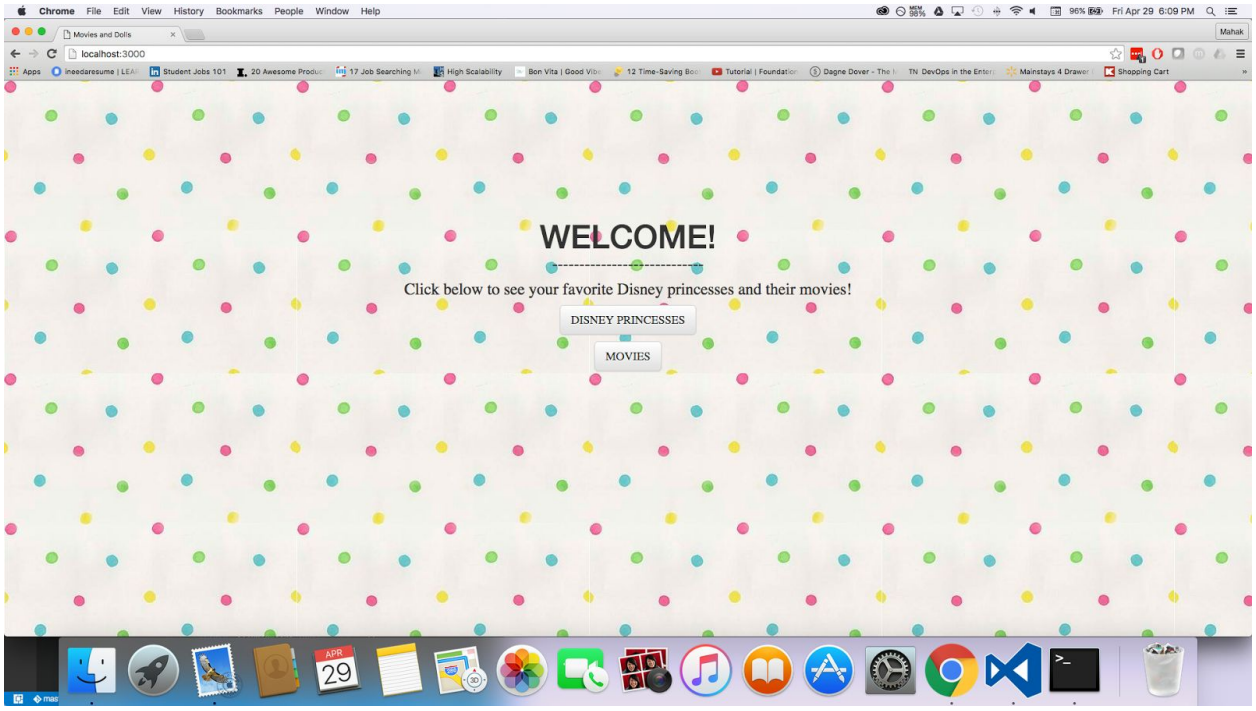
If the user clicks on the Disney Princesses button, a list of dolls is displayed. Here, the user has multiple options: he can either create a new entry, edit a current entry, go to the previous page or see raw JSON data. For each of these functions, a separate button is provided. If the user chooses to add a new entry, a form is presented, takes input from the user and updates data. Persistence is handled as a JSON file (data.json) is saved to the local filesystem. Deletion can be achieved by selecting a checkbox and pressing the Delete button. Paths used here:
'/dolls' for listing all the dolls available
'/dolls/1' or '/dolls/2' ('/dolls/#doll_id) for editing a doll with the specified id
'/dolls/api' for displaying raw JSON data. (Please close the window and restart application if you want to go back from this step)

If the user clicks the Movies button, the layout and functioning on the application remains the same. A page listing movies with the corresponding dolls is presented. Again, the user has an option to add a new movie, edit an existing entry, go to the previous page or see raw JSON data. Here, persistence is handled as a JSON file (mdata.json) which is saved to the local filesystem. Deletion can be achieved by selecting a checkbox and pressing the Delete button. Paths used here are:
'/movies' for listing all movies available or previously entered.
'/movies/1' or 'movies/2' ('/movies/#movie_id) for editing a movie with that id
'/movies/mapi' for displaying raw JSON data.

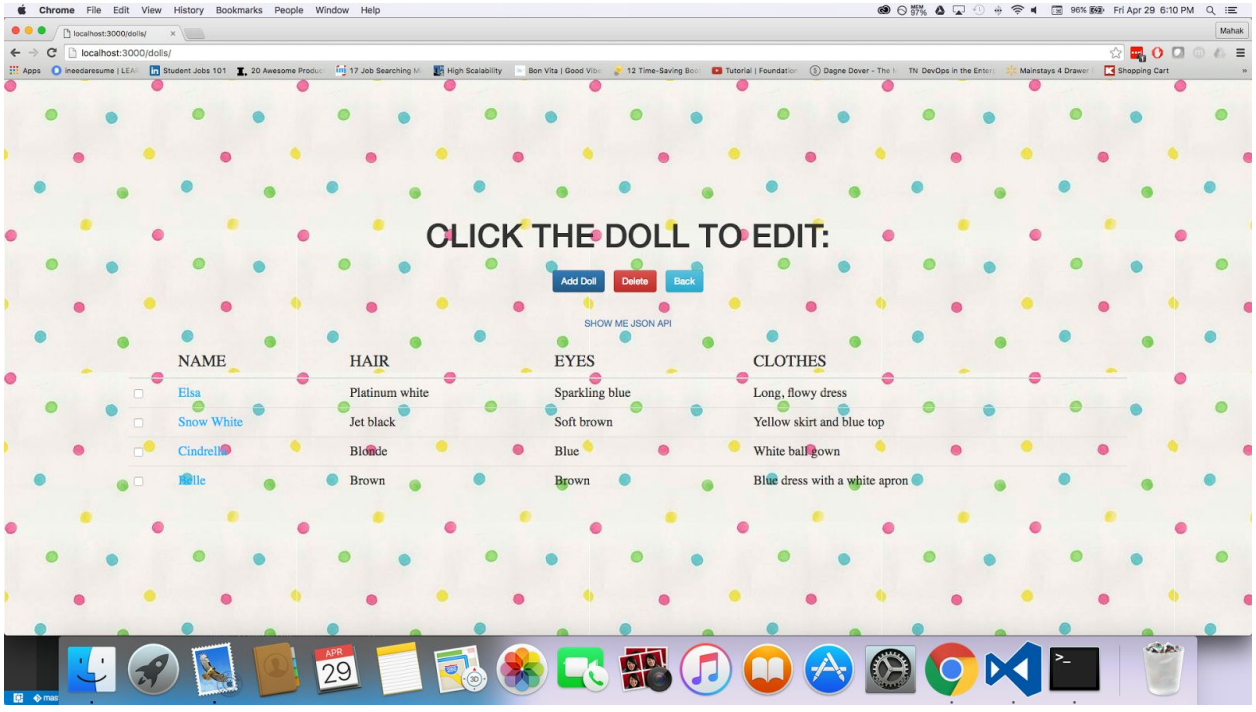
Each page has a back button which should be used to go back. PLEASE DO NOT USE THE BROWSER BACK BUTTON.

Below are some screenshots displaying the working application. For better visibility, I have included these in the “Screenshots” folder in the root directory.

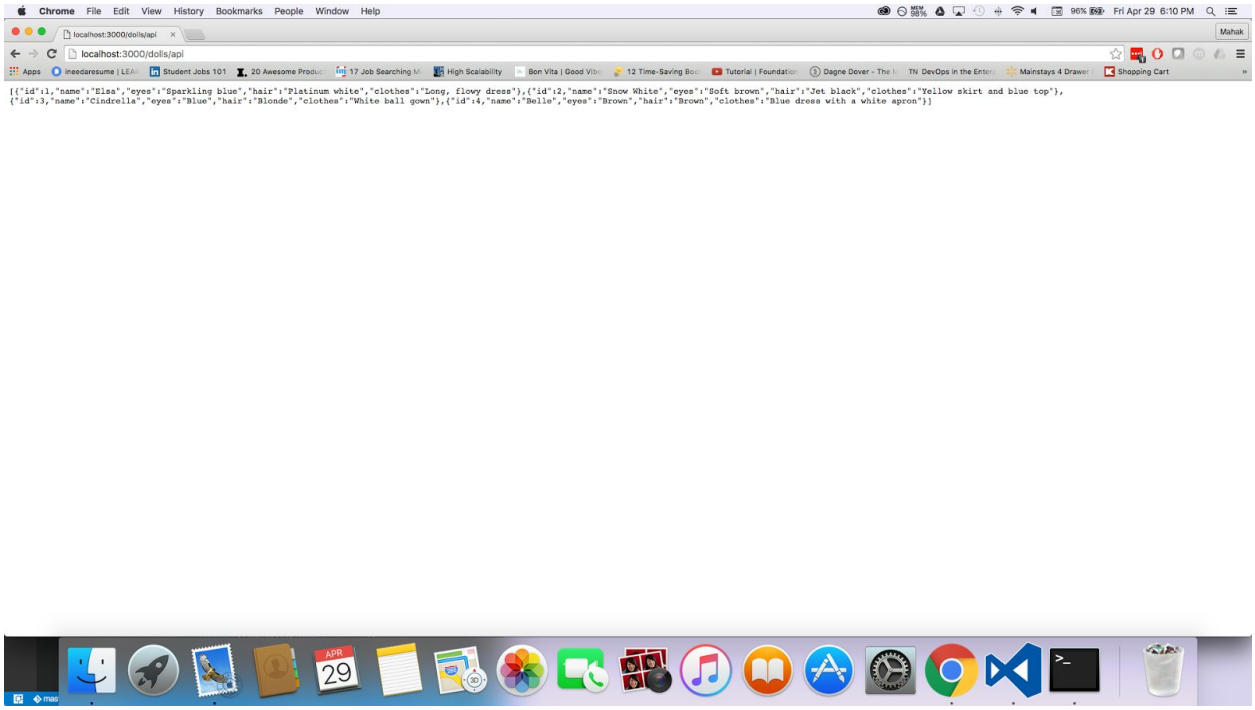
Homepage:



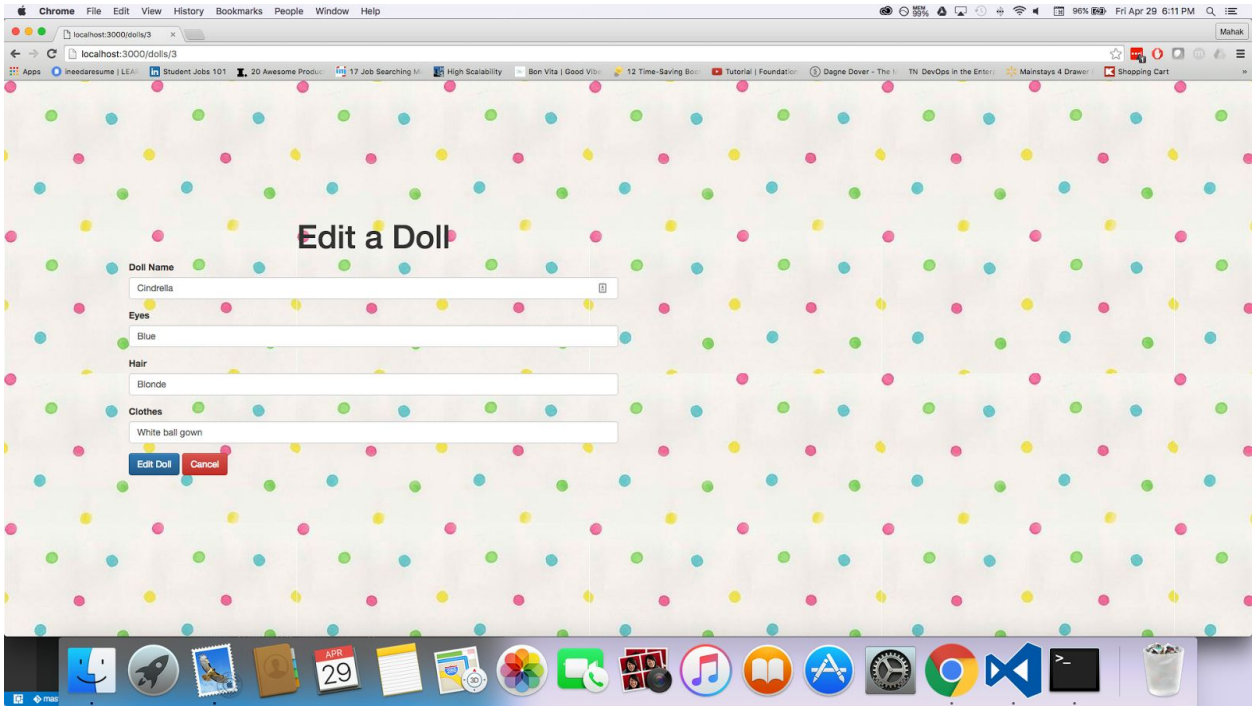
Click Disney Princesses:



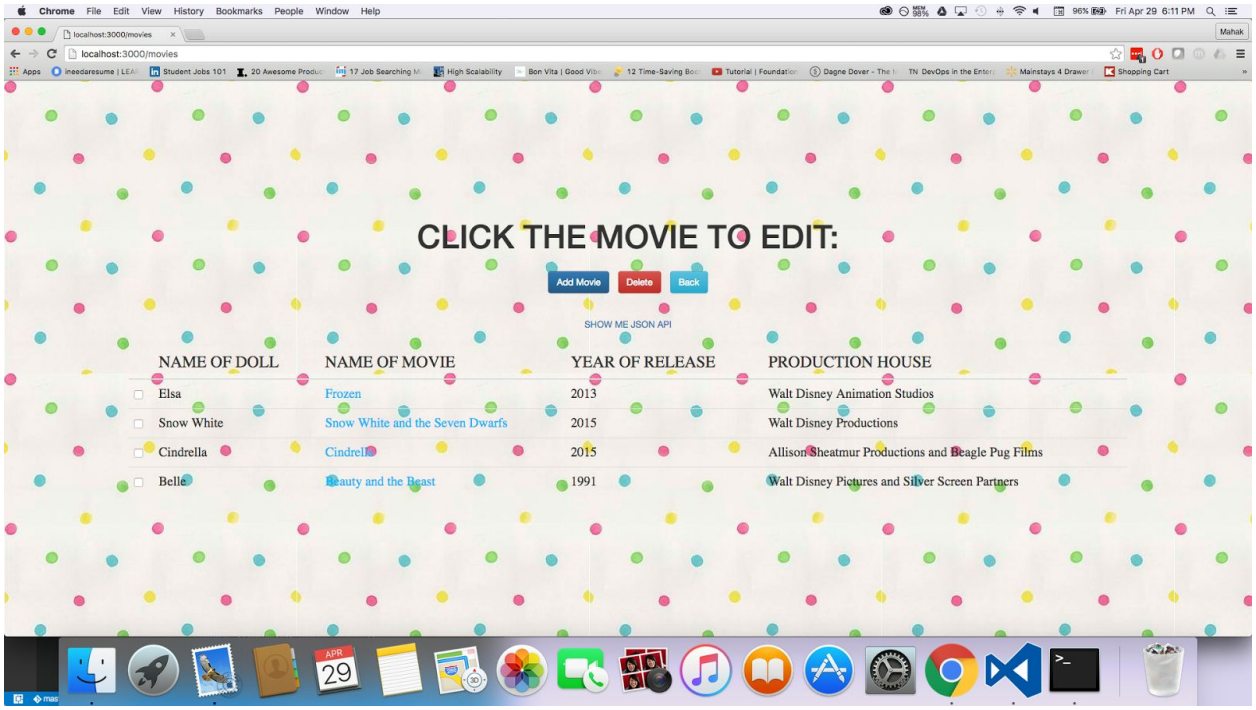
Click “Show JSON Data”:



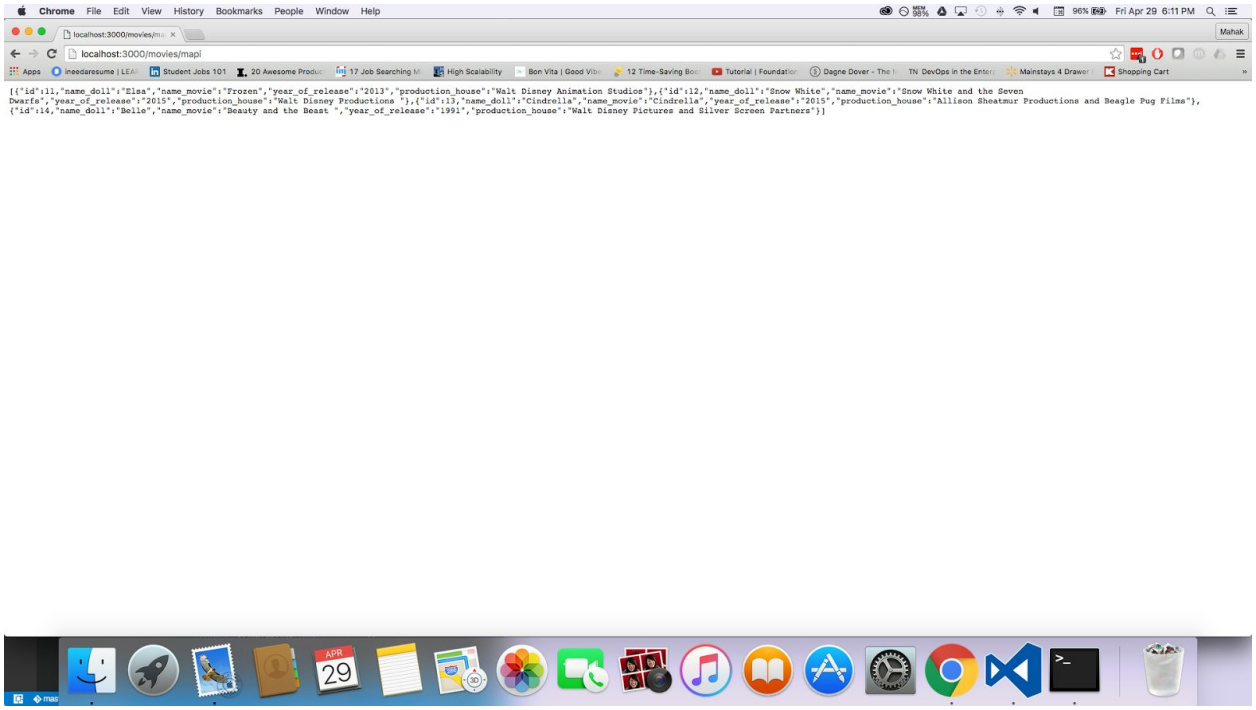
Go back to edit an entry:



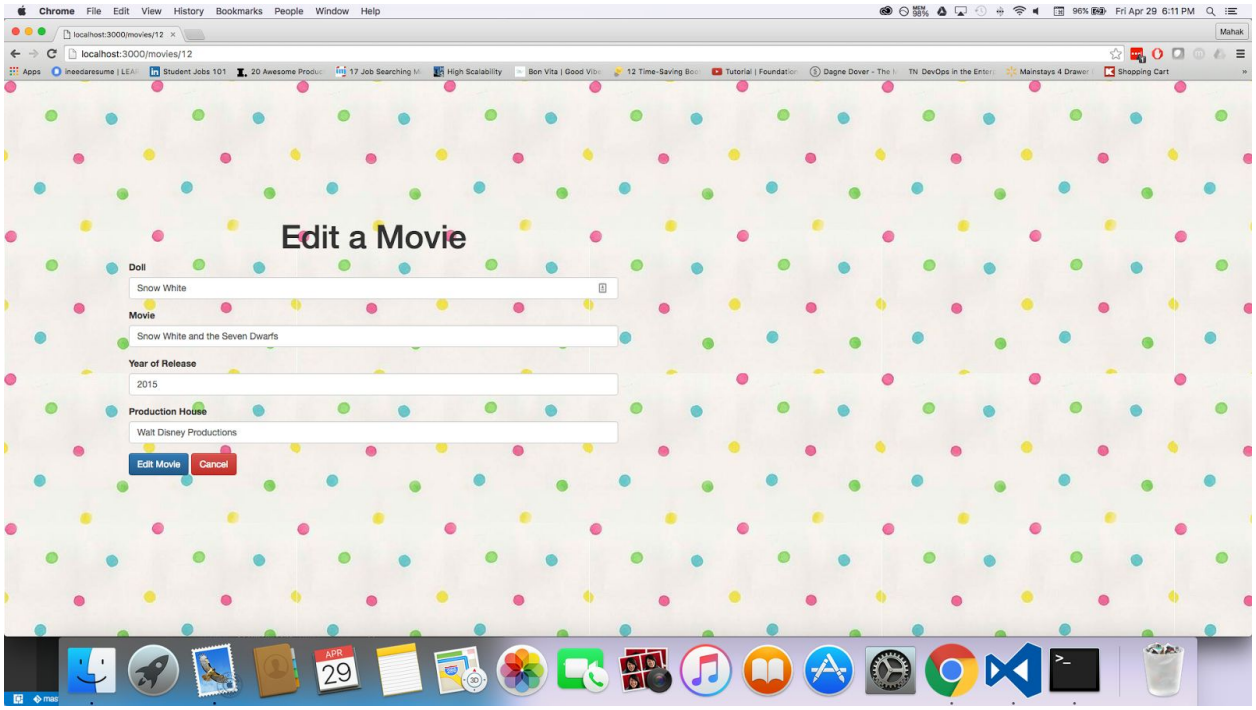
Go to homepage and select “Movies”:



Click “Show JSON Data”:



Go back to edit an entry:



In order to successfully create this application, I used a variety of tools. For compatibility across multiple screens and a visually pleasant interface, I used Bootstrap (v3.3.6). The bulk of my functionality I decided to do in Javascript. I also incorporated jquery for event handling and Ajax. Also using NodeJS (mentioned above) along with Express framework and decided to explore writing scripts in Jade. The purpose of the Jade scripts in this application is to fetch and display data in a specified way. So I have avoided setting up any logic in these files (files can be found in the routes directory). There are 3 sets of Jade scripts in the application: one set deals with getting and displaying data for the movies (addMovie.jade, editformMovie.jade, editMovie.jade and listMovie), other for dolls (add.jade, edit.jade, editform.jade and list,jade)and the last one contains scripts common to both (display.jade, error.jade, index.jade and layout.jade). They do not have their own folders but they are divided so. I used Visual Studio Code on my MacBook Air to develop this application (OSX El Capitan). I had integrated my Visual Studio Code with my github account.
(Navigate to
https://github.com/Mahak-Patil/ITMD-565-Assignments/tree/master/mpatil_a20323104_Project2
to track my progress!)

While working on this project, I explored many functionalities of Jade. I got to play around with RESTful apis, developing new ones, modifying the existing ones etc. Working with REST and CRUD applications, I got to see how they differ from each other. What I found when working with REST, interaction with a working system can be achieved. Whereas CRUD provides basic data manipulation. In CRUD, basic operations can be done on a data repository. This data can be either store on the local filesystem or a database like MongoDB, Cassandra or even MySQL.

REST, I learned, operated on resource representations. What this means is, some form of abstraction and its relationship with other data objects. Notice that in CRUD, we can handle data objects as they are whereas in REST some level of abstraction is achieved. In REST, every representation has a URL. A major issue I faced while implementing this project was getting the APIs right! I initially had trouble with getting the levels the way I wanted. But, since I had worked on this before, I was able to solve it and get the APIs the way I wanted.