**WEB DEVELOPMENT - SERVER END (Node.js)**

We have implemented the server level using node.js + Express and MongoDB.

**Node.js:**

**Node.js** is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side. Node.js lets developers use JavaScript for server-side scripting, running scripts server-side to produce dynamic web page content *before* the page is sent to the user's web browser.

Consequently, Node.js represents a "JavaScript everywhere" paradigm,unifying web-application development around a single programming language, rather than different languages for server side and client side scripts.

We will be creating a Node application, that will allow users to input data that they want to store in a MongoDB database. It will also show all items that have been entered into the database.

**RESTful API:**

A RESTful API is an application program interface that uses HTTP requests to GET, PUT, POST and DELETE data. We will be using an API to define when we add data to our database and when we read from the database.

**npm:**

**npm** is a package manager for the JavaScript programming language. It is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm.

Along with this, we installed few other packages,

1. **Express :** ‘npm install express --save'
2. **body-parser :** ‘npm install body-parser --save’
3. **mongoose :** **‘**npm install mongoose --save'

**Express:**

**Express.js**, or simply **Express**, is a web application framework for Node.js, released as free and open-source software.  It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.

**Express Middleware**: To fill out the contents of our endpoint, we want to store the values entered by the user into the database. The values for are in the body of the request that we send to the server. We want to capture that data, convert it to JSON and store it into the database.

Express.js version 4 removed all middleware. To parse the data in the body we will need to add middleware into our application to provide this functionality. We will be using the body-parser module.

**Body-parser:**

To handle HTTP POST request in **Express.js** version 4 and above, you need to install middleware module called body-parser.

body-parser extract the entire body portion of an incoming request stream and exposes it on req.body.

**Mongoose :**

**mongoose** is an Object modeling tool for **MongoDB**. **Mongoose** is built upon the **MongoDB** driver to provide programmers with a way to model their data.

**Loading data to database:**

Mongoose provides a save function that will take a JSON object and store it in the database. Our body-parser middleware, will convert the user’s input into the JSON format for us.

In webpage, we are connected to the server with: <http://localhost:3000> [3000-port number]. First we should configure and run mongo.exe in cmd.

Then we should type node app.js in the command prompt. After it gets compiled and then we connect to the server. The main page has signup and login page links. When we sign up, the values are stored into the database. Then check mongo-shell to see the data of registered users.

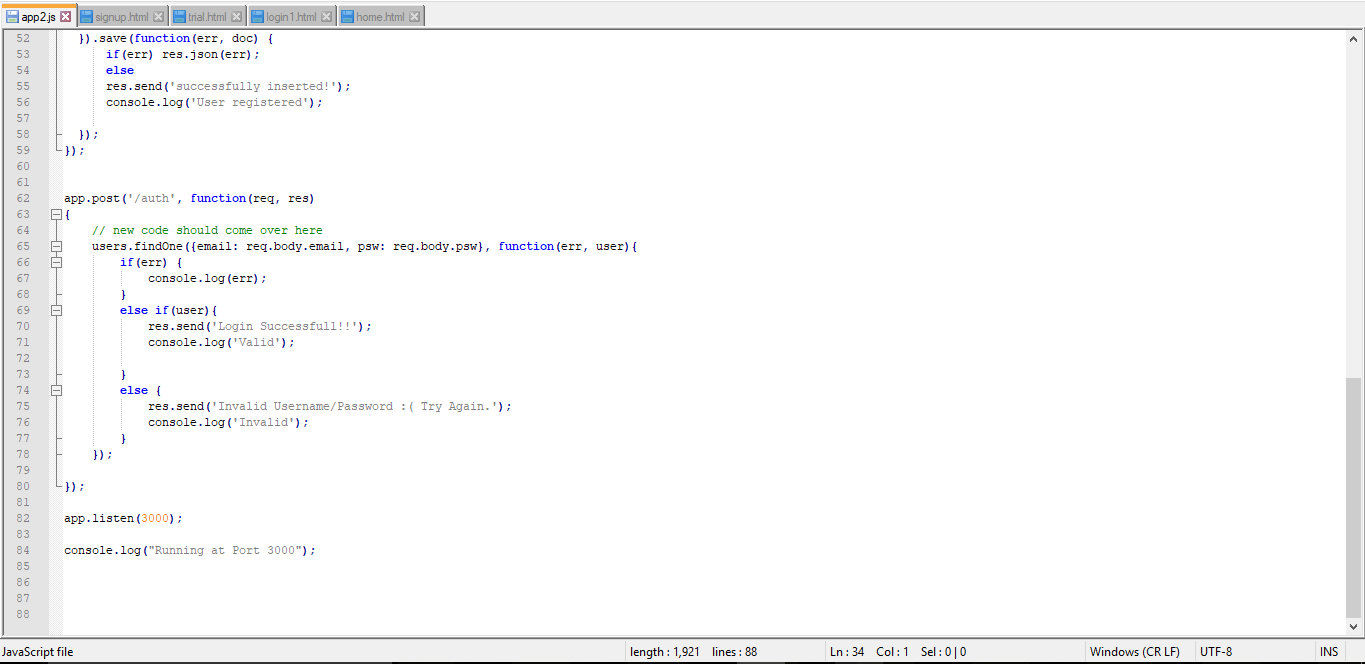
For the login page if the mail id and password doesn’t match with the data stored during registration, then it would display error message else prints “login successful".

**Snapshots of the code and steps:**

app.js



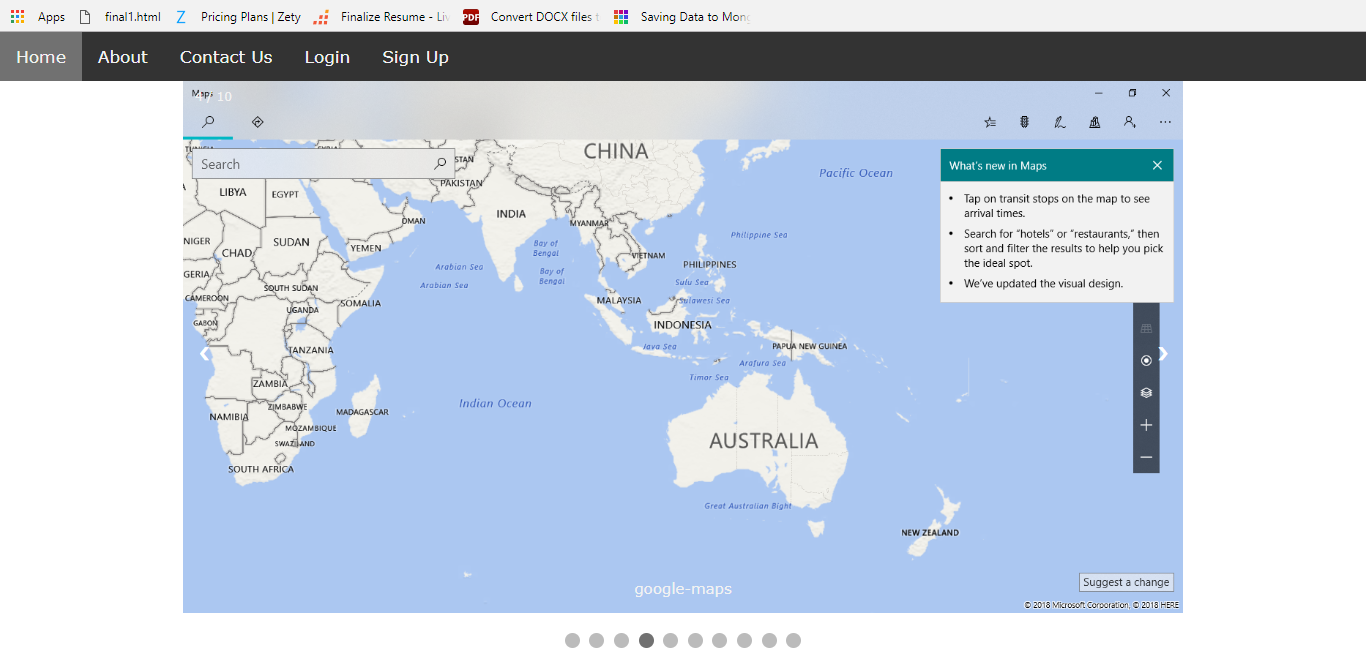




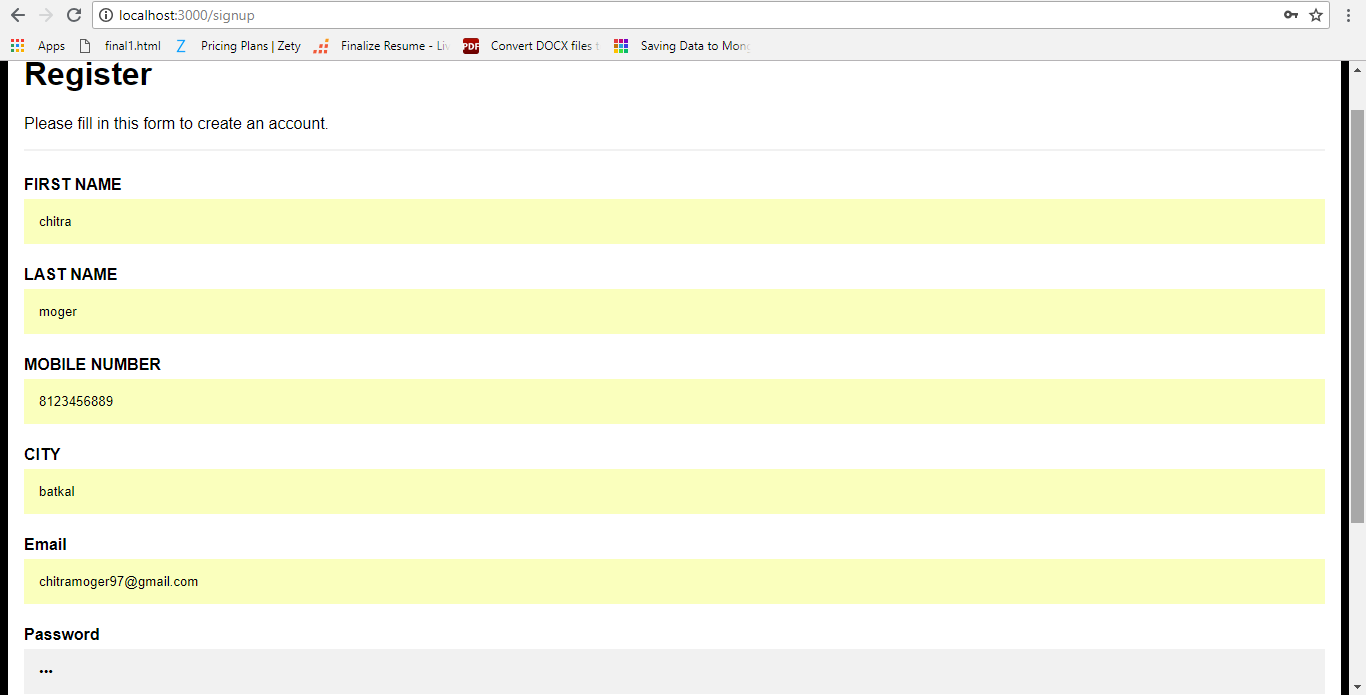
Running app.js

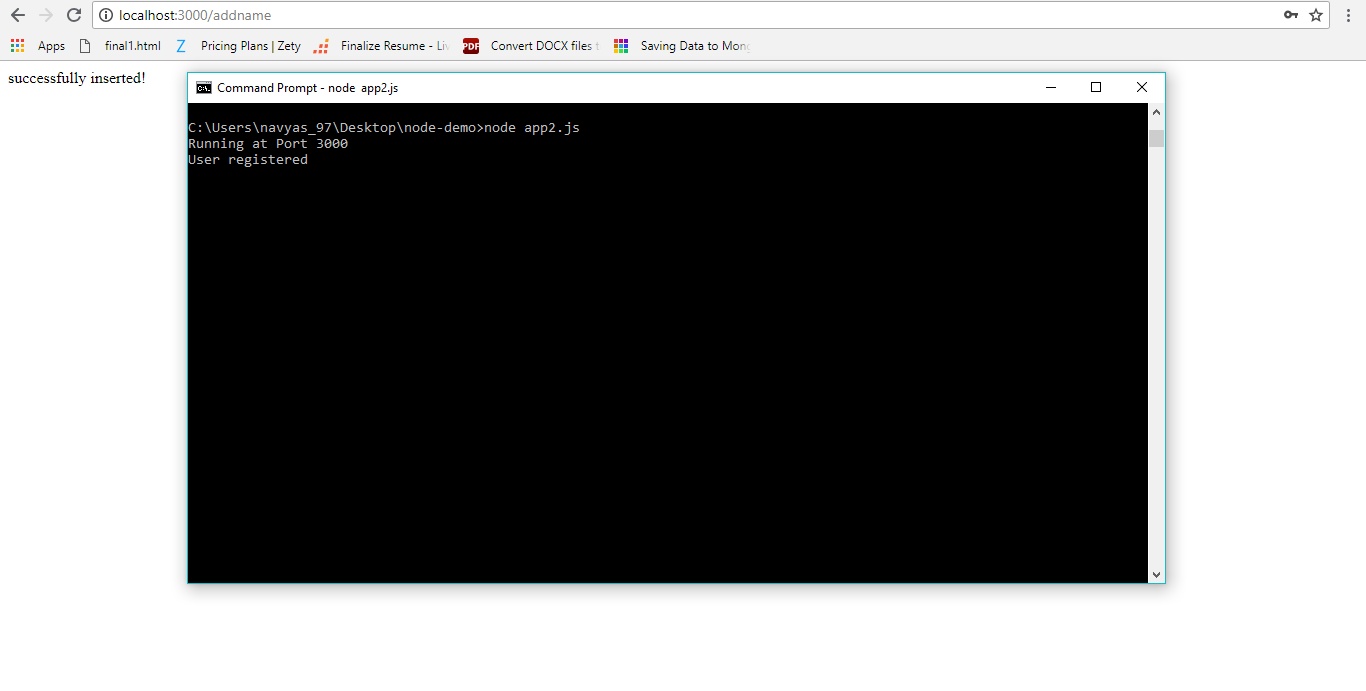


Home page

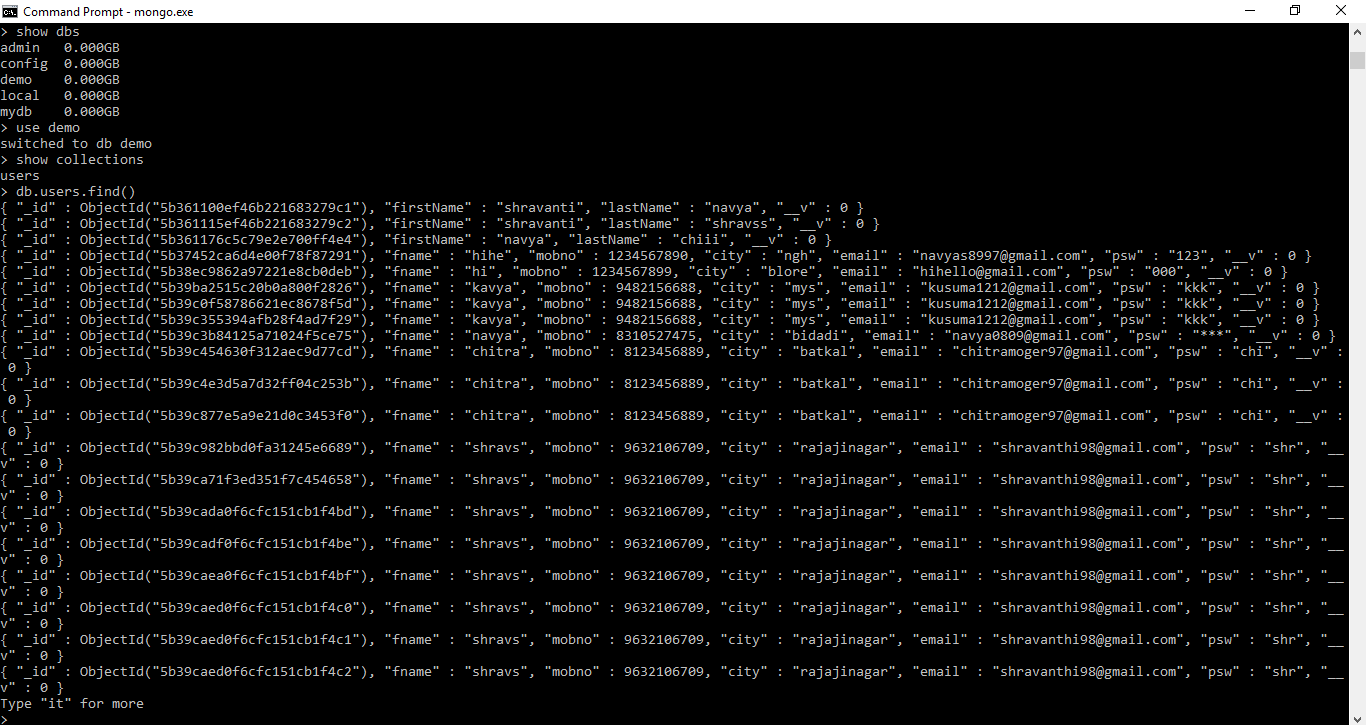


Register page

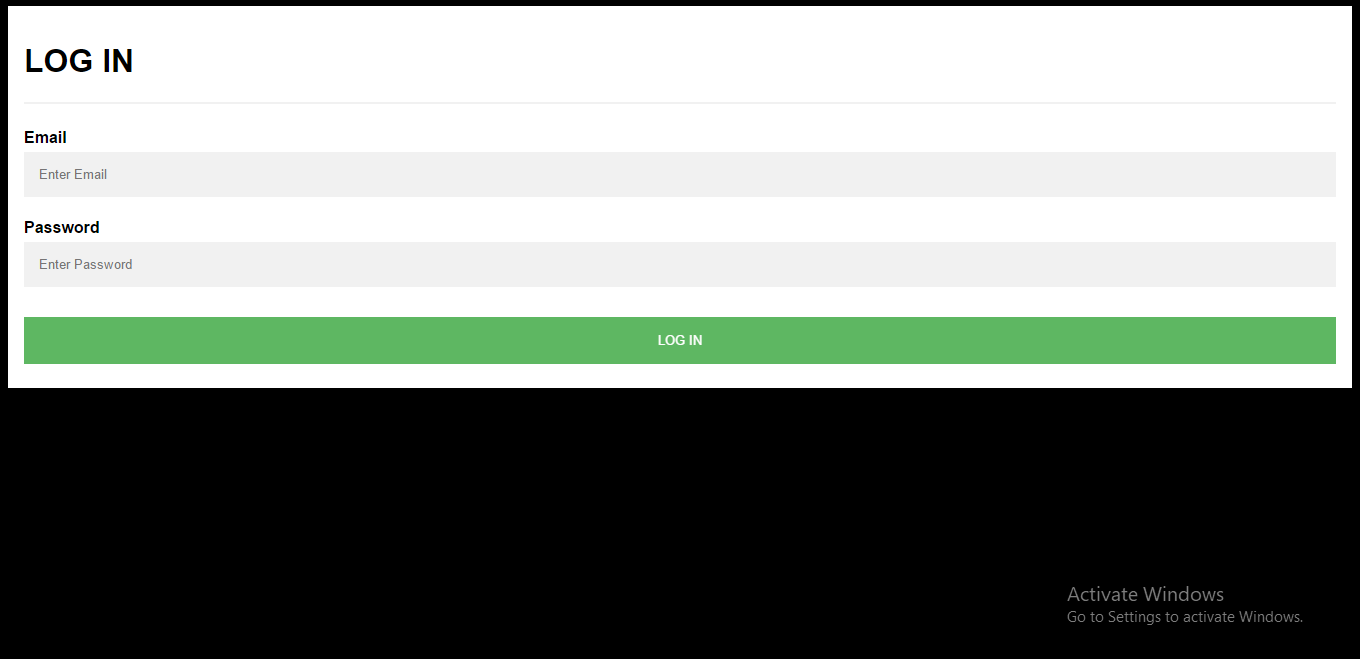




Database



Login page



Main page

