



# Jupiter Weather

Alex Mazur, Aileen Ma, Brianna Griffin, Owen Carlson





# What is Jupiter Weather?



 City Scanner

 Scan

 My Location

 Set

 Locations














 Weather Lists ▾

 Clear List

 Save

Jupiter Weather is a website that allows users to easily compare the weather at different locations. It also allows to users to save a group of locations in a list. These lists are able to be loaded by the user at a later time on any device.

# Tools

- Framework: Node.js with Express 
- Database: MongoDB with Mongoose  
- VCS: GitHub 
- IDE: Visual Studio Code 
- Deployment Env: Google Cloud Platform / Heroku / MemCachier   
- NPM (Node Package Manager) 
- Dependencies: Bcrypt / express-session / body-parser / requests / connect-memjs
- Front-end Styling: Bootstrap 
- Front-end Engine: EJS 
- Project Tracker: Trello 
- Management Method: Agile 

# Node.js with Express



Usefulness Rating: 5/5

Node.js and express are the backbone of the entire project. The websites backend is a node server that uses the express framework.

When an HTTP request is made where the server is listening, it will give a response. The response could be to load a frontend view, log some data in the db, fetch some data from the db, or anything else that we can program.

# MongoDB with Mongoose



Usefulness Rating: 5/5

MongoDB is a NoSQL database where data is stored in JSON like object.

Mongoose is a framework for accessing MongoDB.

We used MongoDB Atlas hosted on Google Cloud Platform to store user information. This user information was used to verify logins and to store user created lists. Lists can be used to compare the weather and travel times to different locations.

```

//Mongoose query for a single user based off the username passed to us by the login-modal form
userModel.findOne({
  username: req.body.username,
}),
(err,user) =>
{
  if (err)
  {
    console.log(err);
    res.redirect('/');
  }

  else if (!user)
  {
    console.log("No user found");
    response.messages.push('invusername');
    console.log(response.messages);
    res.render('pages/index', response);
  }
  //If a user is found, use bcrypt to compare the user entered password with the encrypted password in mongo
  else
  {
    bcrypt.compare(req.body.password, user.password, (err, match) =>

```

>

```

  _id: ObjectId("5e9a5a2eb5e97b01d7ed93a5")
  username: "thisisatest"
  password: "$2b$10$cGcJXRkk5iMlec84E4adeX2kxZuc7BBfxm.aQp7sPtFpybbmn6KC"
  ✓ lists: Array
    ✓ 0: Object
      ✓ locations: Array
        0: "Boulder, CO, USA"
        1: "Chicago, IL, USA"
        2: "San Francisco, CA, USA"
      _id: ObjectId("5e9a5e5fca793c02d3236fde")
      name: "example"
    > 1: Object
    > 2: Object
    > 3: Object
    > 4: Object
    > 5: Object
  __v: 0


```

# GitHub



Usefulness Rating: 5/5

We used GitHub as a version control system. Each group member pushed their changes to a new branch. Pull requests were required before merging any branch into our master branch.

 97 commits 2 branches 0 packages 0 releases 1 environment 2 contributors

Branch: master ▾

New pull request

Create new file

Upload files









Find file


Clone or download ▾



boulderocoder9 Adjusting readme

Latest commit 5d5f9b0 23 hours ago

 <a href="#">models</a>	Adjusted the user model to support lists, added ability to saves list...	8 days ago
 <a href="#">resources</a>	Delete .DS_Store	yesterday
 <a href="#">routes</a>	Adjusted Changelog, removed a comment, and a useless parameter to cal...	23 hours ago
 <a href="#">views</a>	CL 4/22 cont5.	yesterday
 <a href="#">.gitignore</a>	Adjusted Changelog, removed a comment, and a useless parameter to cal...	23 hours ago
 <a href="#">README.md</a>	Adjusting readme	23 hours ago
 <a href="#">package.json</a>	CL 4/23	yesterday
 <a href="#">server.js</a>	4/23 cont.	yesterday

 [README.md](#)

## Project Jupiter

A live version can be found [here](#).

Jupiter Weather is an express / node website that is hosted on Heroku. The website allows users to quickly and easily compare the weather at multiple locations at the same time. Each card contains the current conditions (rain, sun, snow, etc.), and icon for every condition, current temperature, max / min temperature, and a field for displaying travel times. Each location is stored in a list of cards that a user can save to a MongoDB database and reload at a later date. Users can also set their location and view travel times on each card. Clicking on cards when the user has set their location will open up an embedded directions map.

### Getting up and running *locally*:



# Visual Studio Code

Usefulness Rating: 5/5

Some project members used VSC as their IDE.

# Google Cloud Platform / Heroku / Memcachier

Usefulness Rating: 5/5



Google Cloud Platform: MongoDB Atlas is hosted on GCP. GCP also handles our API calls to Google Directions and Google Maps Embed API.

Heroku: Our app is hosted on Heroku using the automatic deploy from Github branch tool.

MemCachier: This is a Heroku add on that allows us to store session information in memory cache rather than on disk (which is not really possible on Heroku).



NPM (Node Package Manager) is used for managing our varied dependencies.

# bcrypt / express-session / body-parser / requests

Usefulness Rating: 5/5

Bcrypt: Used for hashing and salting plain-text passwords to store in the database. It is also used when a user is logging in to check for password validity.

Express-session: Used to keeping track of whether a user is logged in and what cards are loaded.

body-parser: Allows express to easily see the information passed to the backend during an HTTP request.

Requests: Allows us to make API calls (OpenWeather, Google Maps Embed, Google Directions)

```
{
  "name": "jupiter-weather",
  "main": "server.js",
  "scripts": {
    "start": "npx nodemon server.js"
  },
  "dependencies": {
    "bcrypt": "^4.0.0",
    "body-parser": "^1.19.0",
    "connect-memjs": "^0.2.1",
    "dotenv": "^8.2.0",
    "ejs": "^3.0.1",
    "express": "^4.17.1",
    "express-session": "^1.17.0",
    "mongoose": "^5.9.2",
    "request": "^2.88.2"
  },
  "devDependencies": {
    "nodemon": "^1.19.0"
  }
}
```



Usefulness Rating: 5/5

Bootstrap was used for easy styling of HTML elements.

# EJS



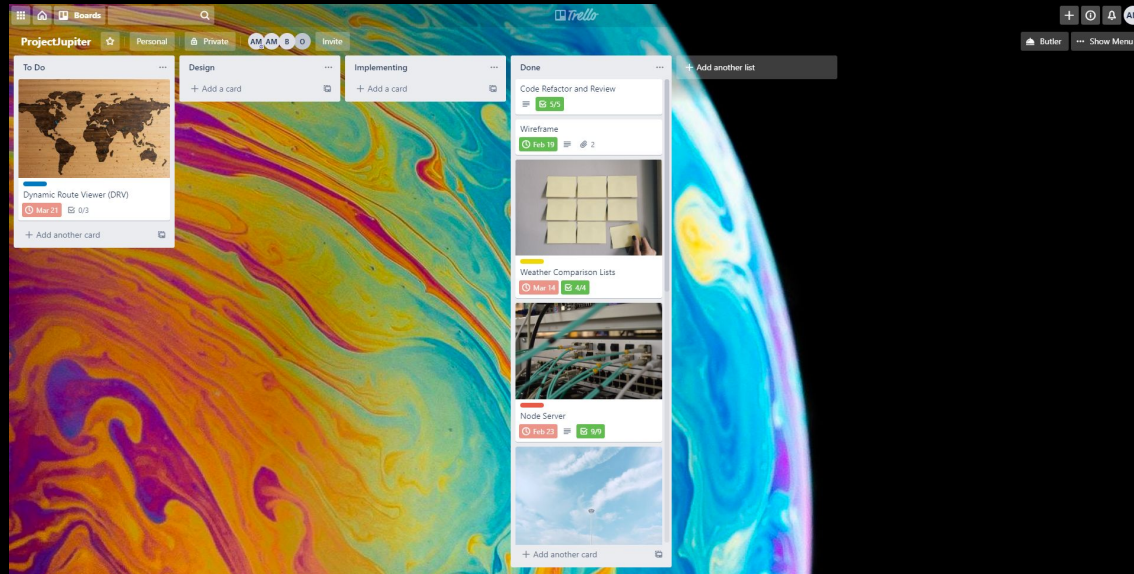
Usefulness Rating: 5/5

EJS is a view engine that allows us to embed javascript directly in our html. It also allows us to pass data from the backend to the frontend in the form of a JSON object.

# Trello

Usefulness Rating: 4/5

We used Trello to keep track of project objectives and timelines





# Agile



Usefulness Rating: 4/5

We used the Agile methodology for managing our project. We used Trello to keep track of our progress during sprints and create user stories.

# Challenges

Honestly, and unsurprisingly, the biggest challenge was keeping everyone on the same page, with the global pandemic and all.

A recent challenge was figuring out how to host the app in some way that any user with internet access could reach. The reason this was challenging was because we utilized express-session for passing some core pieces of data around.

Express-session by default will attempt to save session information on disk. This isn't really possible on most (or any) cloud services. Our solution was Heroku + MemCachier.

# Time for the Project Demo!

<https://jupiter-weather-app.herokuapp.com/>