Travel Buddy

Our project is a travel/activities helper that allows users to search for a specific location and will return the real time temperature at this location as well as a list of the top-rated attractions. This aids users in creating future travel itineraries or finding fun nearby activities! Users will be required to sign in via 3rd party authentication to ensure security of personal information and aid in ease of use (eliminated need for new/unnecessary username & passwords)

**Backend Implementation:** Express.js + Passport.js

* For our backend implementation we opted to use Express.js which is a free open-source web application framework for Node.js. After consideration of goals and further research into backend frameworks we chose Express.js over its competitors like Django and Flask. While both Django and Flask cater towards Python developers, Express is key component of the extensive JavaScript ecosystem and Node.js framework which provides full stack development capabilities with extensive libraries and documentation, thus our team felt most comfortable coding in JavaScript. Additionally, Express.js is extremely flexible and customizable while Django and Flask are more opinioned on coding format. Given the freedom provided in this project the potentially of additional flexibility was appealing. Most notable however are Express’s effective routing model, middleware support for 3rd party libraries, and easy handling of request and responses from servers using RESTful API’s as discussed in class. Flask lacks a user authentication/authorization feature, and overall the middleware and routing handling of Express versus Django seemed more suited to our needs. Specifically, we used Passport.js to handle authentication middleware which was extremely compatible with Express.js and was easy to use and integrate into our code.

**Frontend Implementation:** React.js

* In conjunction with our Express JavaScript backend implementation, we opted to use React.js to create an interactive UI. Once we decided to use Node.js to build our application, React was the obvious choice as JavaScript’s own open-source library for frontend web development. Major benefits of React include fast rendering, component-based architecture which is extremely helpful for reusable components, and simple scripting making HTML markups fairly easy. Again, as with other components in the JavaScript ecosystem, it is accompanied by extensive documentation, tutorials, and helpful resources.

**Database:** MongoDB

* Our decision on what database to use was the most controversial, and indeed we almost completely switched halfway through the project. We were between MongoDB and PostgreSQL. Our first attempt at 3rd party authentication using a database utilized MongoDB, however we ran into many problems with Node.js version compatibility with MongoDB. Ultimately, MongoDB’s on demand flexible schema practices, easy horizontal scaling, and simple us prevailed over PostreSQL which required table building in the command line.