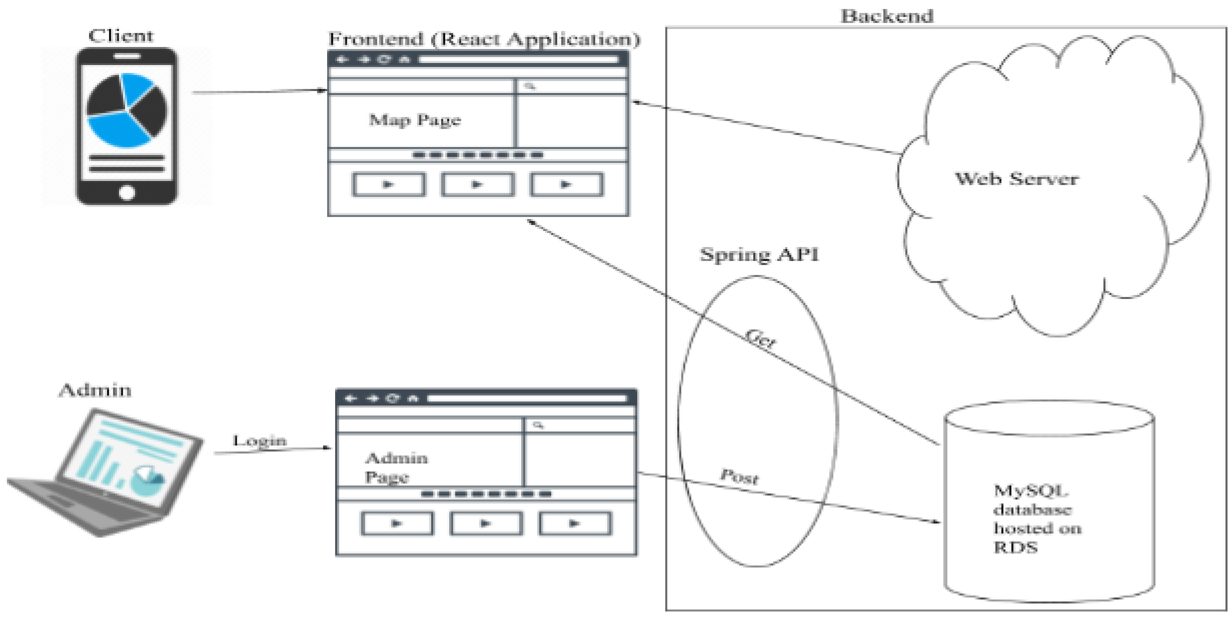
OO Design & UML

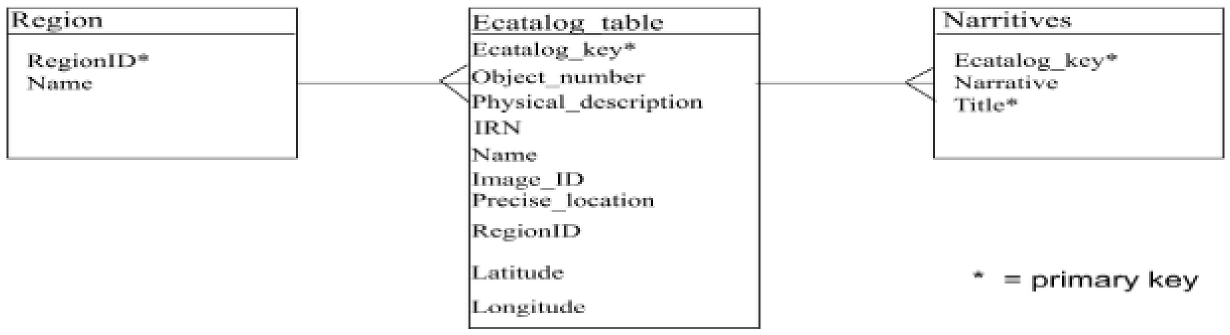
Firstly, we store our data on a MySQL database hosted on Amazon Web Services' (AWS) RDS. Additionally, we'll be utilising SpringBoot as our backend API for its ease of use and the underlying language, Java (a programming language everyone on the team knows proficiently). The API is implemented using the data-access-object (DAO) design pattern, which fits our specifications. Lastly, for frontend, React will be used for its reputation of performance, speed and its built-in templates. Though we all had to pick up JavaScript.

As we progressed through the basic webpage, many modules were utilised, for example, we implemented an interactive map of Bristol using Mapbox. Its flexibility allows for many ways to customise how our map functions. Furthermore, all of our forms (admin features) are built using Material UI, giving it a clean look. By the end of the implementation, we should've had the web application hosted on a web server.

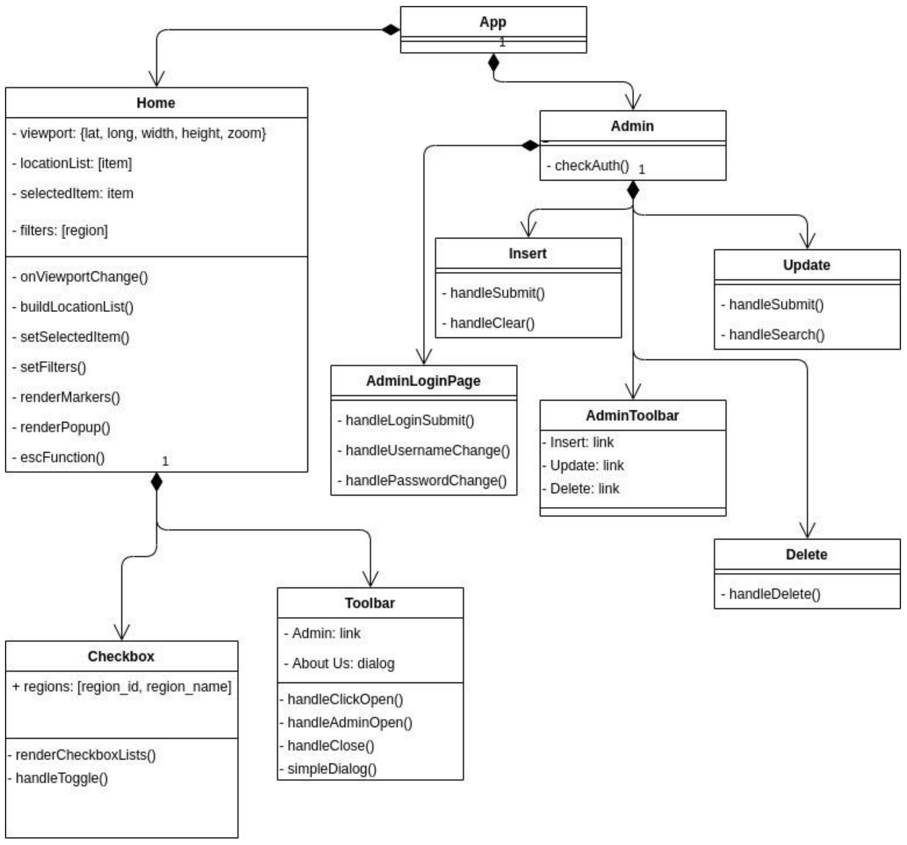
High Level Architecture Diagram



Relational Database Diagram

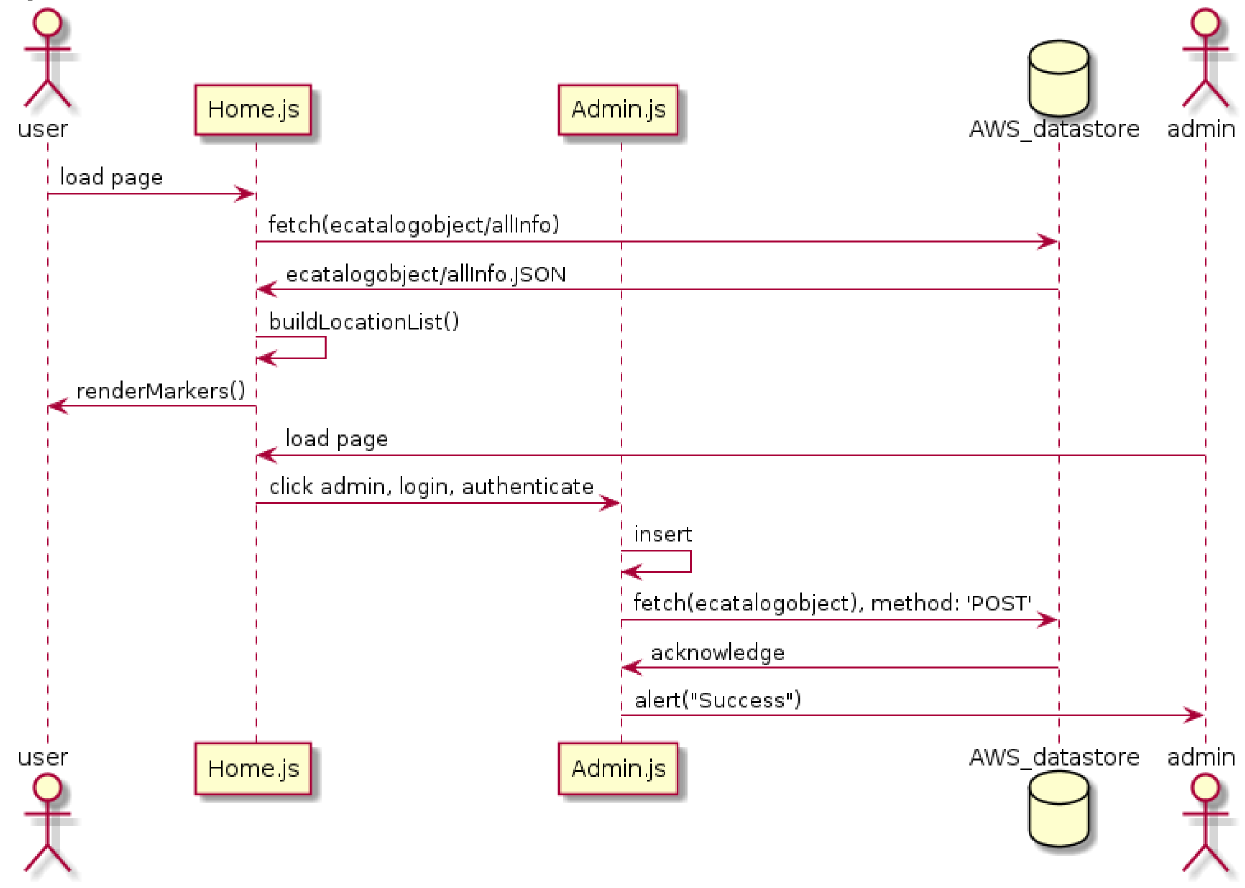


Static Model



In our static model, we've broken our app down into its individual React components. For the attributes we've mostly substituted this for the states of the components, however, for the toolbars we instead just used the different links they contained. Although these aren't strictly attributes, it conveys more within the diagram to get a general overview of the entire app. For simplicity's sake, some of the smaller helper functions have been omitted from this diagram, leaving just the functions which convey some form of core functionality. From this diagram, it is easy to see which components import other components from the composition arrows and gives a good overview of the functionality of our app. In the creation of this diagram, it occurred to me that maybe some inheritance could be used to improve our design as both insert.js and update.js make use of handleSubmit(). However, with respect to the scale of our app, it would not make any huge improvements in time spent coding but would just be slightly cleaner.

Dynamic Model



Our dynamic model gives a step by step example of an example action that both a user and an admin could make. In this case, the user is simply loading the page whilst the admin is inserting a new entry into the database, reflecting two of our key user stories. Admin also have the ability to delete and update but these have much the same sequence as insert. Users are also able to select markers and filter them by region, however, this just manipulates the view not the actual database so this type of diagram wouldn't convey a lot of information for those cases. Also, in reality insert, update, and delete each have their own javascript component that admin.js links to, however, for the sake of a more readable diagram all the admin features are assumed to be condensed into admin.js.