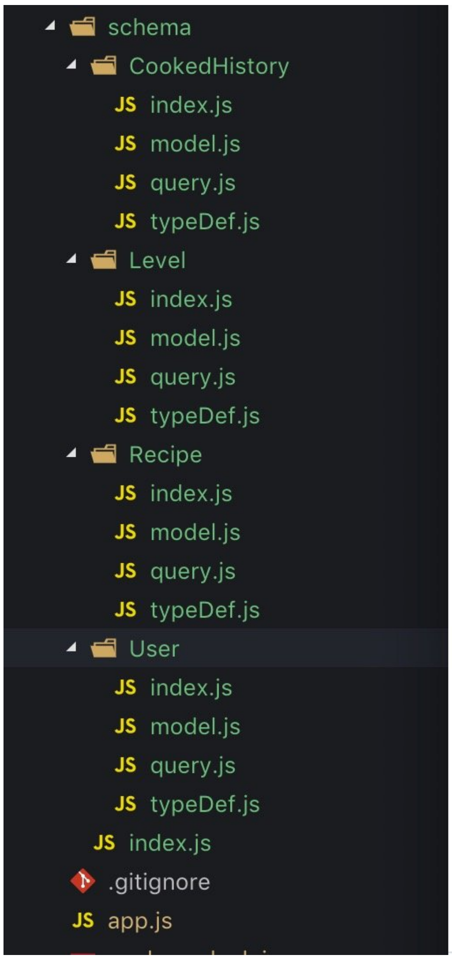
Problem Encountered

Known Issues

1. The application build is around 22Mb, because some modules from 'ExpoKit' is used like Voice function, Icon and Gradient background. Besides, Expo aims to provide support for multiple SDK and OTA update which can allow clients to recieve latest update without reinstalling apk. These factors give rise to the oversize of build bundle.
2. In cuisine learning page with multiple levels, the first time rendering may cause some layout error, multiple views stacking up on each other. Swiping back and re-enter the screen solve the issue. After lots of digging, it still cannot be addressed.

GraphQL folder structure

There is **no** best structure, but to make it scalable, either group by `resolvers` , `typeDef` , `Queries` , `Mutations` (later `Subscriptions`) or by different `models` .



```
schema
  CookedHistory
    index.js
    model.js
    query.js
    typeDef.js
  Level
    index.js
    model.js
    query.js
    typeDef.js
  Recipe
    index.js
    model.js
    query.js
    typeDef.js
  User
    index.js
    model.js
    query.js
    typeDef.js
  index.js
.gitignore
app.js
```

```
types
├── Media
│   ├── index.js
│   ├── mediaQueryResolvers.js
│   ├── mediaSchema.js
│   └── mediaTypes.js
├── User
│   ├── UserModel.js
│   ├── UserSchema.js
│   ├── UserType.js
│   ├── index.js
│   ├── mutationResolvers
│   │   ├── confirmEmail.js
│   │   ├── confirmOAuth.js
│   │   ├── followUser.js
│   │   ├── index.js
│   │   ├── loginViewer.js
│   │   ├── registerViewer.js
│   │   ├── unfollowUser.js
│   │   ├── updateAccountSettings.js
│   │   └── updateViewerProfile.js
│   └── queryResolvers
│       ├── followersConnection.js
│       ├── getUser.js
│       └── index.js
```

Relational Model

Do not have to use Mongoose `populate` for relational subdocument, can just search in that collection.

ID Typing

Defining `_id` as `String` explicitly can seem easy, but it requires `default` value field manually set in Mongoose Schema.

> Searching by `ObjectId` can be achieved by providing `String`

Handling ObjectId in GraphQL

To circumvent the error `ID cannot represent value: { _bsontype: \"ObjectID\"}` caused by having to define `_id` field as `ID` or `String` where it should be `ObjectId` in MongoDB, just define it as `String`, and add the following snippet to `app.js` to **override** the default behavior of Mongoose.

```
const { ObjectId } = mongoose.Types;
ObjectId.prototype.valueOf = function () {
  return this.toString();
};
```

Resolver Coding

When coding the **top-level** Query resolvers, which is **Root Resolvers**, the ***first*** argument should be `undefined`. Can note it as `_` or `root` (whatever).

```
Query: {
    getLevel: async (_, param) => await LevelModel.findOne(param)
},
```

And when querying **subdocuments**, which is nested query, the ***first*** argument will be result obtained from **parent** query.

```
Level: {
    recipe: async (level) => await RecipeModel.findById(level.recipe)
}
```

Wiring Up

`makeExecutableSchema` can combine arrays of `typeDefs` and `resolvers` to create a schema, which can be later used in `ApolloServer` creation.

```
const schema = makeExecutableSchema({
    typeDefs: [Root, LevelType],
    resolvers: [resolvers, LevelResolvers],
})
const server = new ApolloServer({ schema });
```