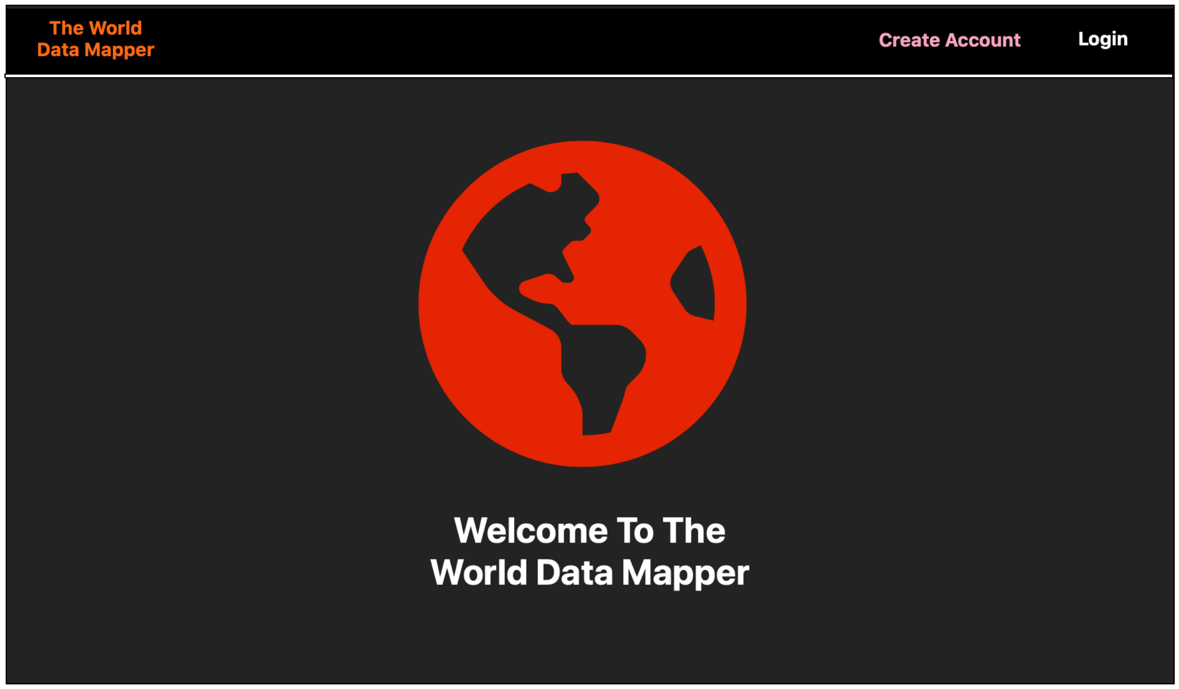
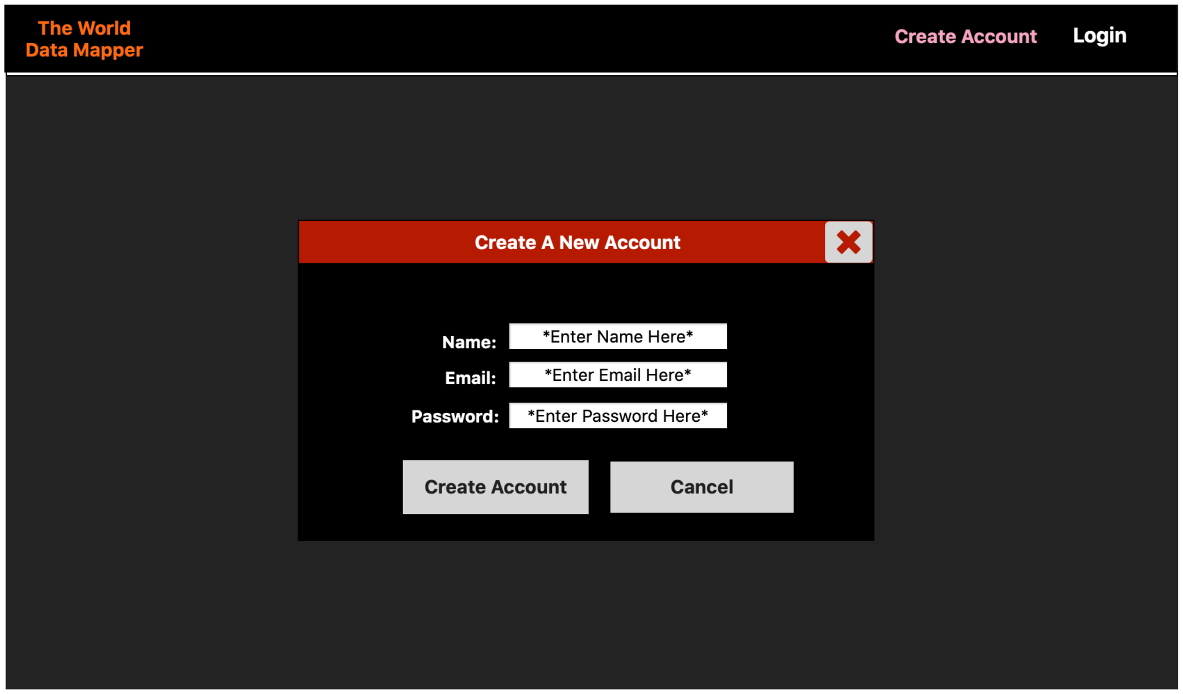
World Data Mapper Design

1. UI Mockup diagram

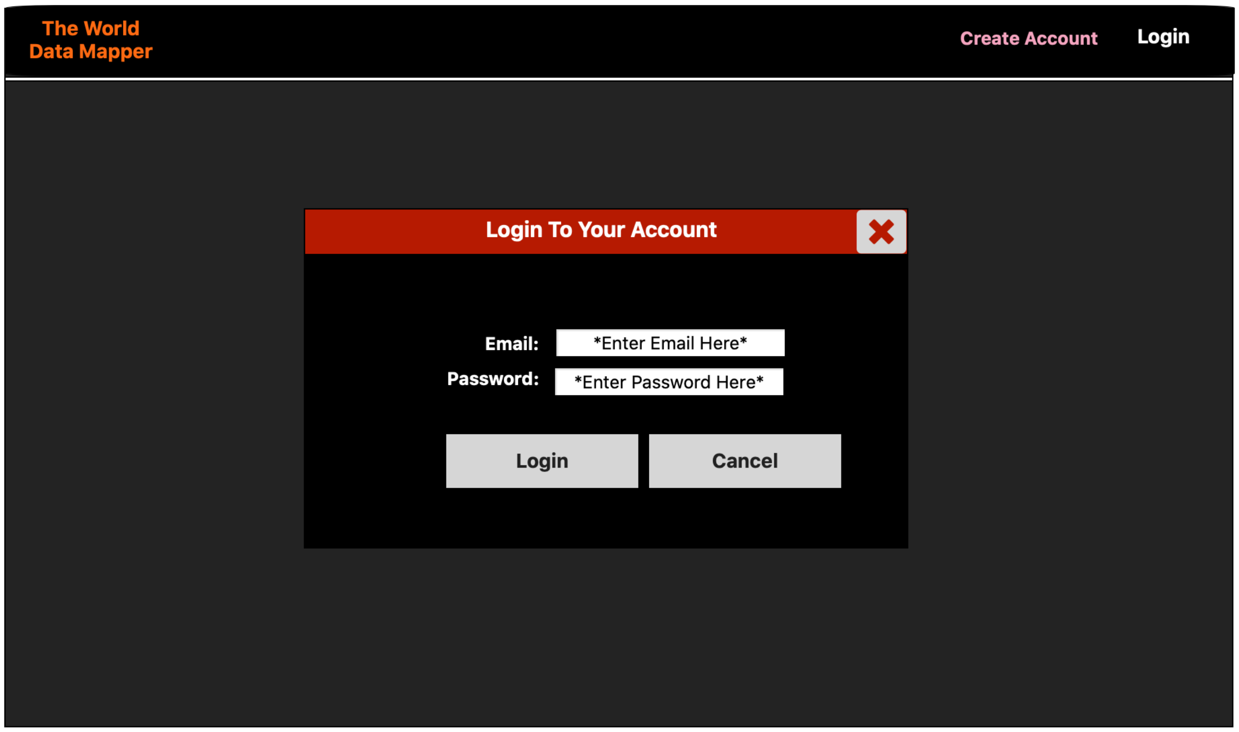
Welcome Screen



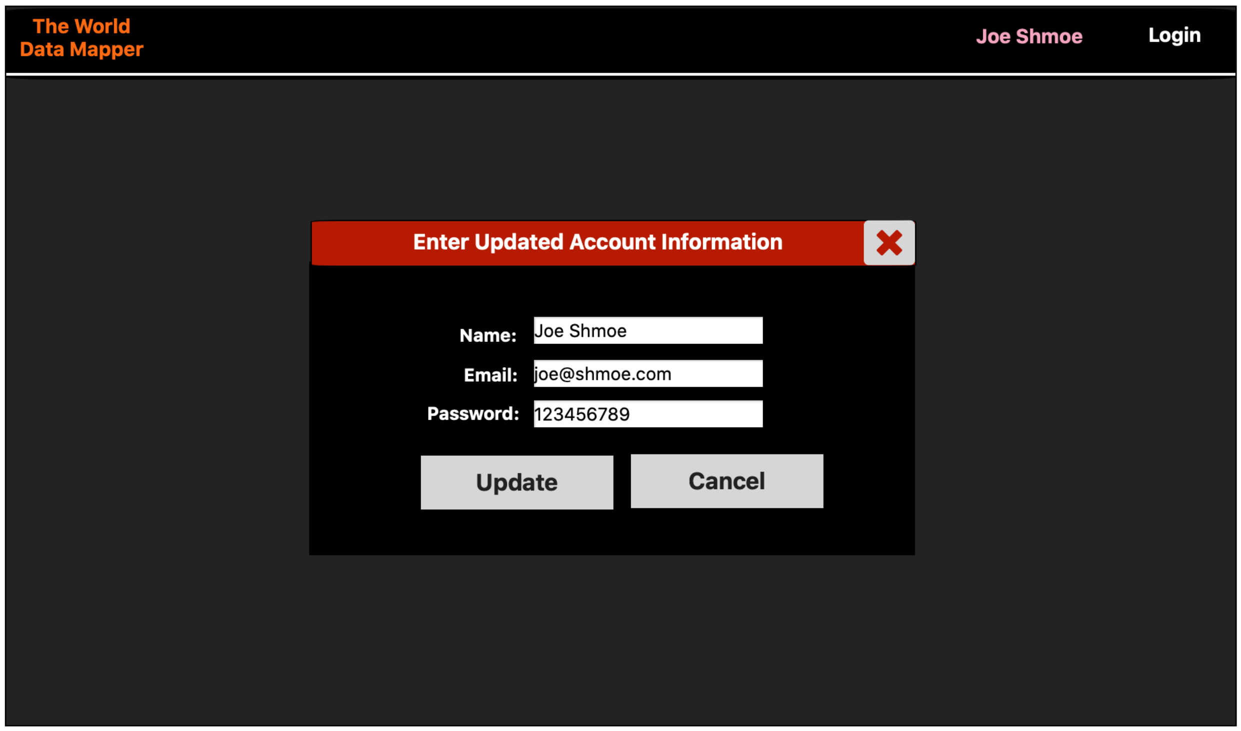
Create Account



Login Screen

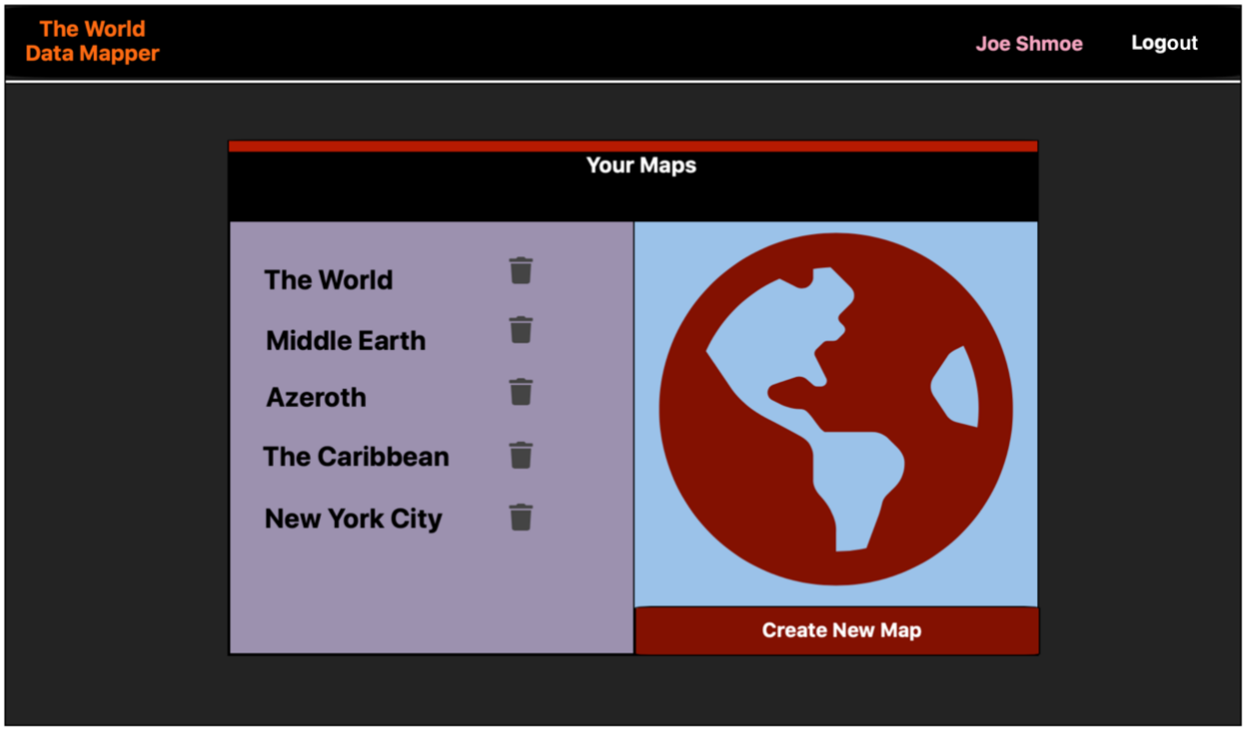


Update Account

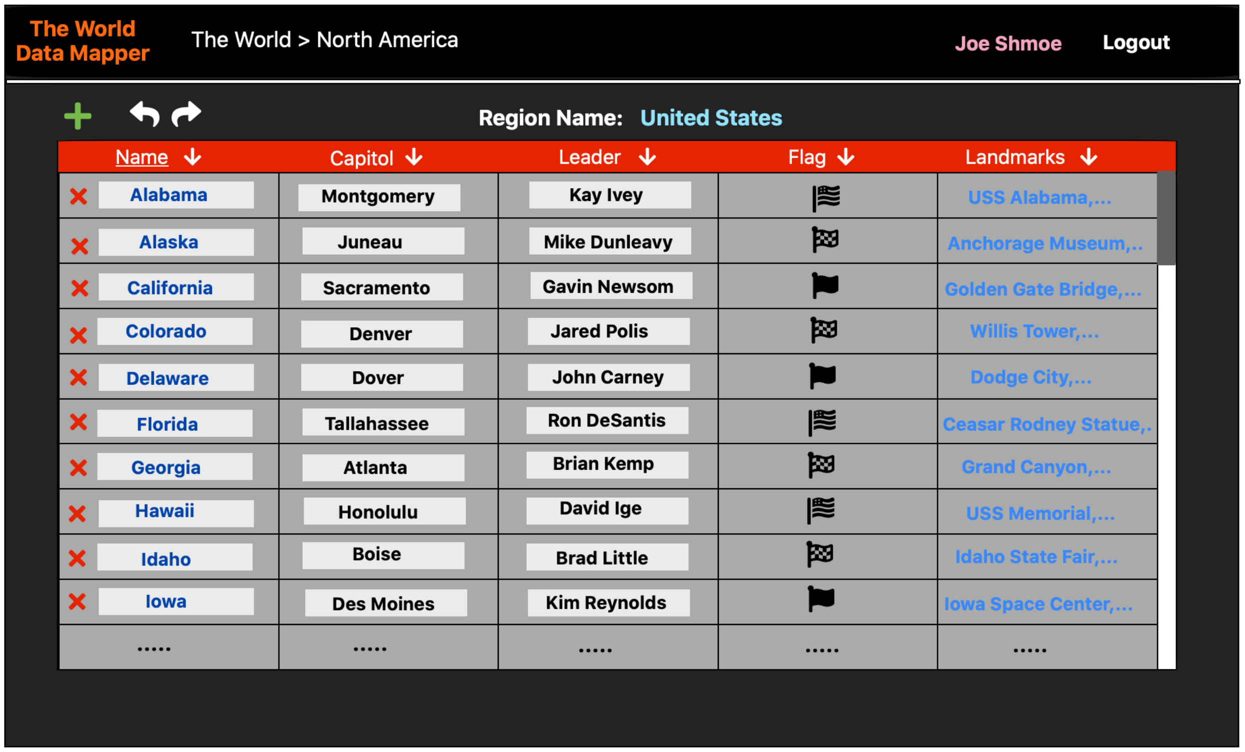


Name: Joe Shmoe
Email: joe@shmoe.com
Password: 123456789

Map Select Screen



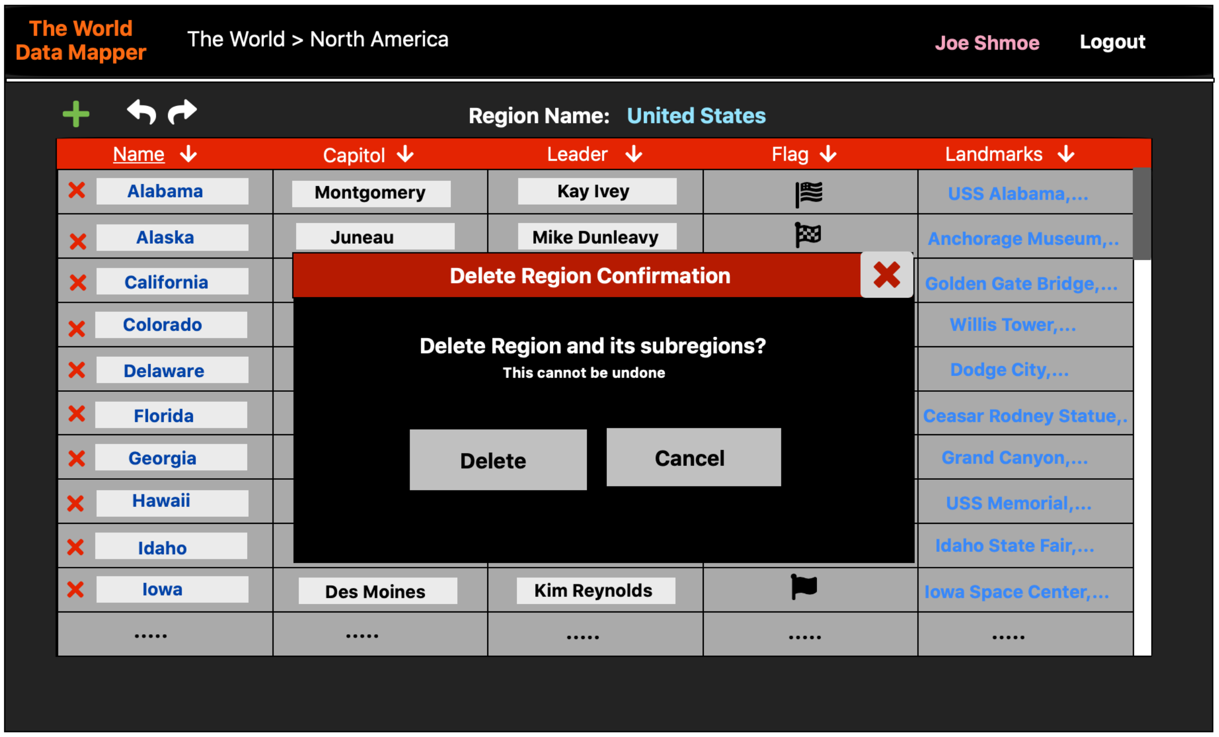
Region Spreadsheet



Region Name: United States

Name ↓	Capitol ↓	Leader ↓	Flag ↓	Landmarks ↓
✗ Alabama	Montgomery	Kay Ivey	▤	USS Alabama,...
✗ Alaska	Juneau	Mike Dunleavy	🏁	Anchorage Museum,...
✗ California	Sacramento	Gavin Newsom	⚑	Golden Gate Bridge,...
✗ Colorado	Denver	Jared Polis	🏁	Willis Tower,...
✗ Delaware	Dover	John Carney	⚑	Dodge City,...
✗ Florida	Tallahassee	Ron DeSantis	▤	Ceasar Rodney Statue,.
✗ Georgia	Atlanta	Brian Kemp	🏁	Grand Canyon,...
✗ Hawaii	Honolulu	David Ige	▤	USS Memorial,...
✗ Idaho	Boise	Brad Little	🏁	Idaho State Fair,...
✗ Iowa	Des Moines	Kim Reynolds	⚑	Iowa Space Center,...
.....	.....	.....	.....	.....

Deletion Modal (similar for delete Map, Region or Landmark)



The World Data Mapper

The World > North America

Joe Shmoe

Logout

Region Name: United States

Name	Capitol	Leader	Flag	Landmarks
Alabama	Montgomery	Kay Ivey	⚑	USS Alabama,...
Alaska	Juneau	Mike Dunleavy	⚑	Anchorage Museum,...
California				Golden Gate Bridge,...
Colorado				Willis Tower,...
Delaware				Dodge City,...
Florida				Ceasar Rodney Statue,.
Georgia				Grand Canyon,...
Hawaii				USS Memorial,...
Idaho				Idaho State Fair,...
Iowa	Des Moines	Kim Reynolds	⚑	Iowa Space Center,...
.....	.....	.....	.....	.....

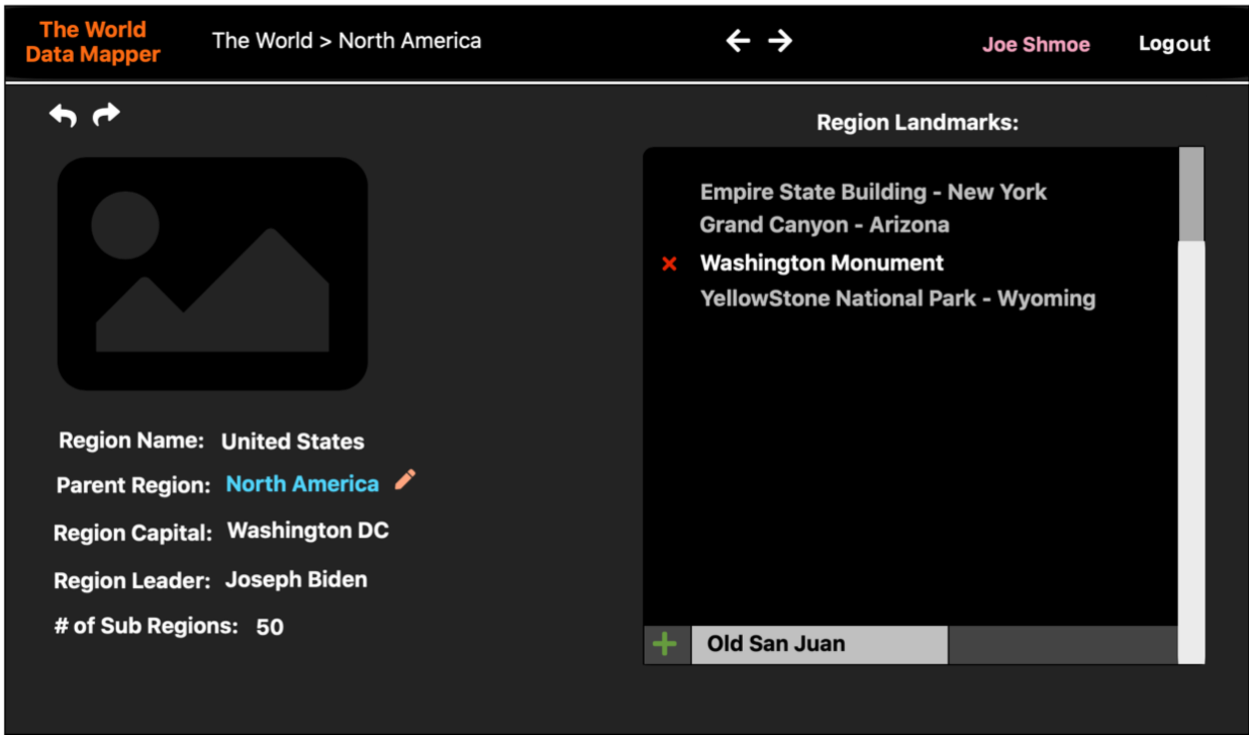
Delete Region Confirmation

Delete Region and its subregions?

This cannot be undone

Delete Cancel

Region Viewer Screen



The World Data Mapper

The World > North America

Joe Shmoe

Logout

Region Landmarks:

Empire State Building - New York

Grand Canyon - Arizona

Washington Monument

YellowStone National Park - Wyoming

Old San Juan

Region Name: United States

Parent Region: North America

Region Capital: Washington DC

Region Leader: Joseph Biden

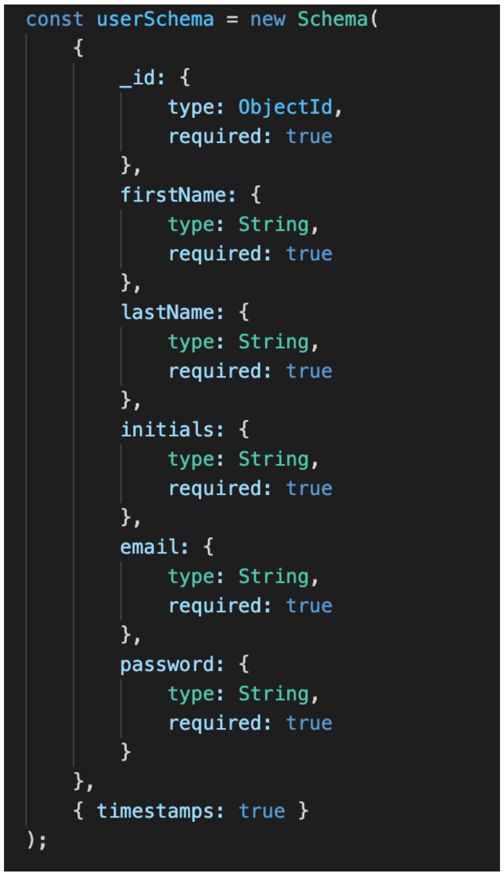
# of Sub Regions: 50

2. Routes

/welcome	
/login	
/create-account	
/update-account	
/maps	(map select page)
/maps:id /regions:id	(region chosen which presents a region spreadsheet)
/regions:id /region-viewer	(region viewer page)

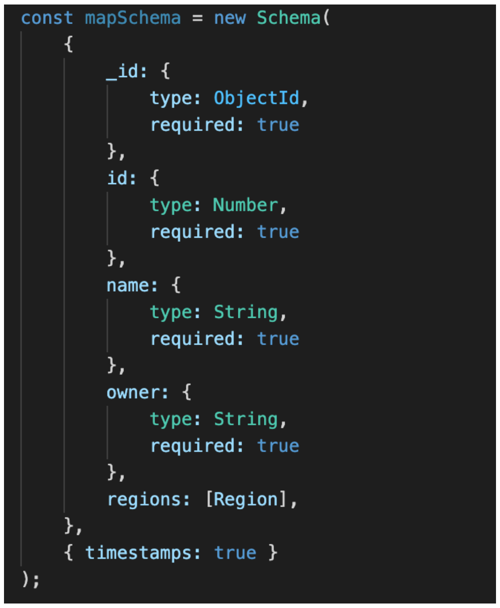
3. Schemas

User



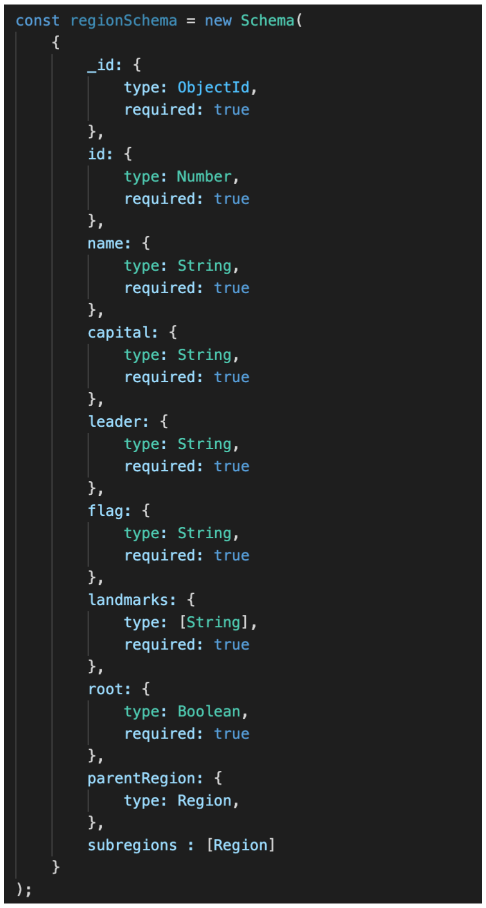
```
const userSchema = new Schema(
    {
        _id: {
            type: ObjectId,
            required: true
        },
        firstName: {
            type: String,
            required: true
        },
        lastName: {
            type: String,
            required: true
        },
        initials: {
            type: String,
            required: true
        },
        email: {
            type: String,
            required: true
        },
        password: {
            type: String,
            required: true
        }
    },
    { timestamps: true }
);
```

Map



```
const mapSchema = new Schema(
    {
        _id: {
            type: ObjectId,
            required: true
        },
        id: {
            type: Number,
            required: true
        },
        name: {
            type: String,
            required: true
        },
        owner: {
            type: String,
            required: true
        },
        regions: [Region],
    },
    { timestamps: true }
);
```

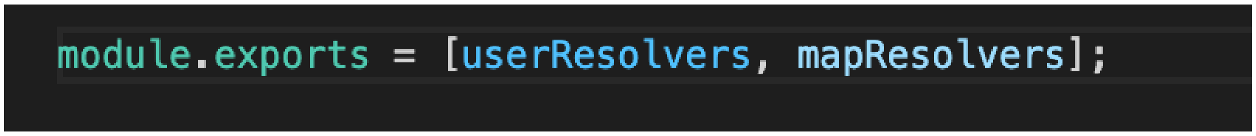
Region



```
const regionSchema = new Schema(
    {
        _id: {
            type: ObjectId,
            required: true
        },
        id: {
            type: Number,
            required: true
        },
        name: {
            type: String,
            required: true
        },
        capital: {
            type: String,
            required: true
        },
        leader: {
            type: String,
            required: true
        },
        flag: {
            type: String,
            required: true
        },
        landmarks: {
            type: [String],
            required: true
        },
        root: {
            type: Boolean,
            required: true
        },
        parentRegion: {
            type: Region,
        },
        subregions : [Region]
    }
);
```

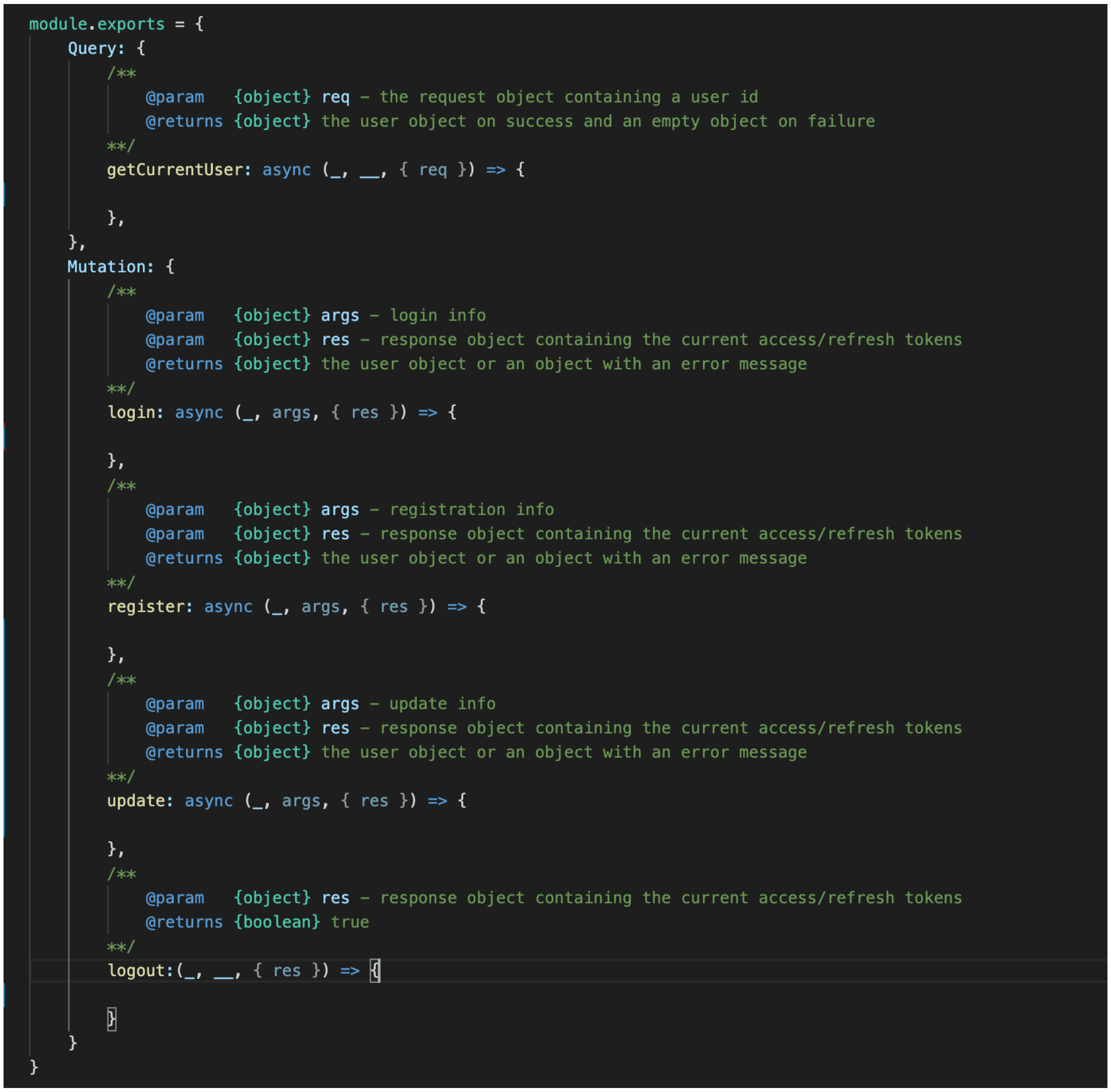
4. Resolvers

Root Resolver



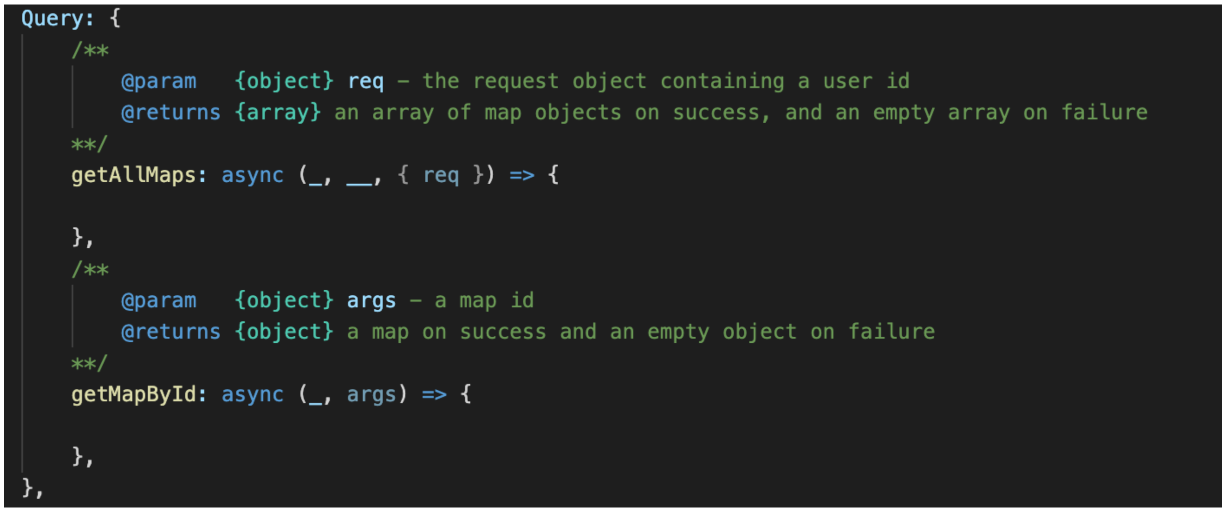
```
module.exports = [userResolvers, mapResolvers];
```

User Resolver

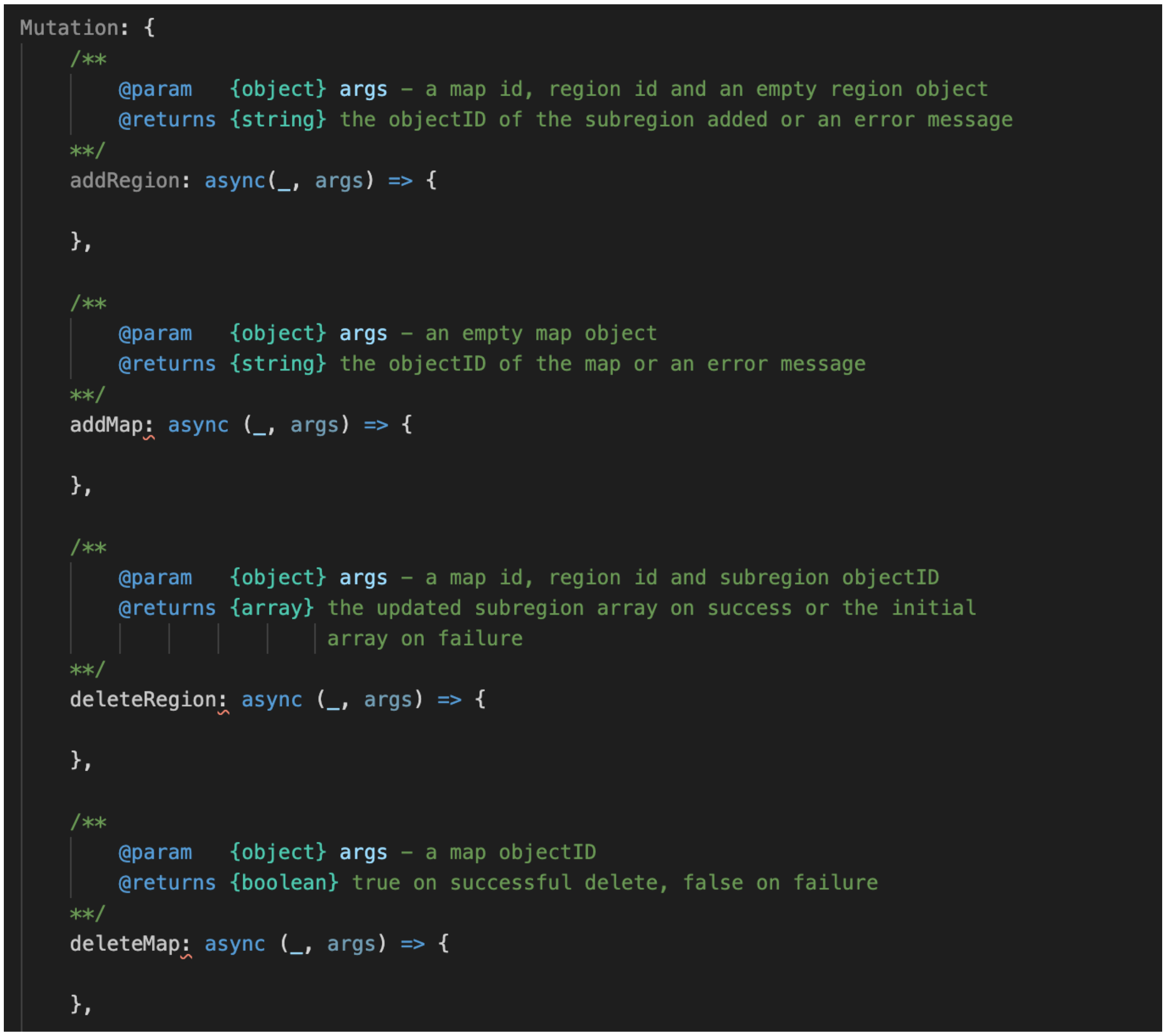


```
module.exports = {
    Query: {
        /**
            @param   {object} req - the request object containing a user id
            @returns {object} the user object on success and an empty object on failure
        **/
        getCurrentUser: async (_, __, { req }) => {

        },
    },
    Mutation: {
        /**
            @param   {object} args - login info
            @param   {object} res - response object containing the current access/refresh tokens
            @returns {object} the user object or an object with an error message
        **/
        login: async (_, args, { res }) => {

        },
        /**
            @param   {object} args - registration info
            @param   {object} res - response object containing the current access/refresh tokens
            @returns {object} the user object or an object with an error message
        **/
        register: async (_, args, { res }) => {

        },
        /**
            @param   {object} args - update info
            @param   {object} res - response object containing the current access/refresh tokens
            @returns {object} the user object or an object with an error message
        **/
        update: async (_, args, { res }) => {

        },
        /**
            @param   {object} res - response object containing the current access/refresh tokens
            @returns {boolean} true
        **/
        logout:(_, __, { res }) => {

        }
    }
}
```

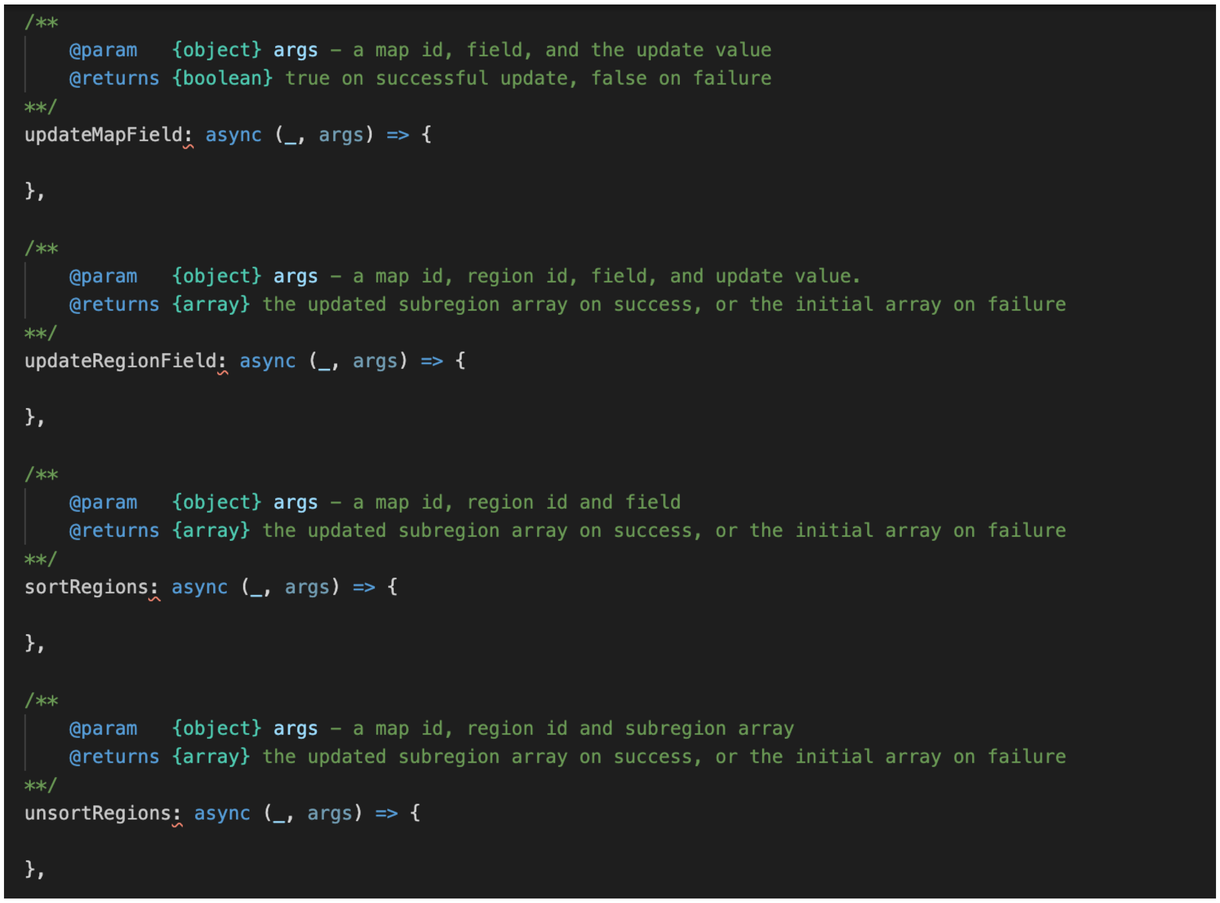
Map Resolver



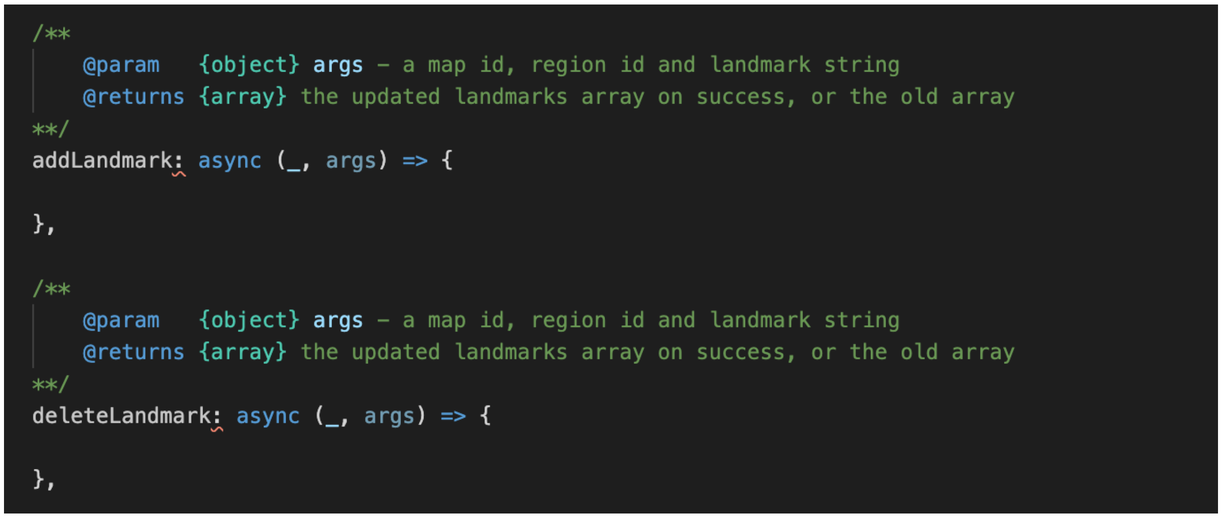
```
Query: {
    /**
        @param   {object} req - the request object containing a user id
        @returns {array} an array of map objects on success, and an empty array on failure
    **/
    getAllMaps: async (_, __, { req }) => {

    },
    /**
        @param   {object} args - a map id
        @returns {object} a map on success and an empty object on failure
    **/
    getMapById: async (_, args) => {

    },
},
```



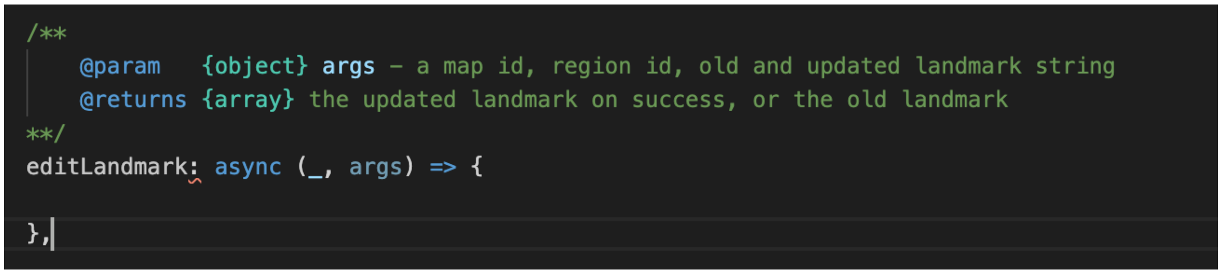
```
Mutation: {
    /**
        @param   {object} args - a map id, region id and an empty region object
        @returns {string} the objectID of the subregion added or an error message
    **/
    addRegion: async(_, args) => {

    },

    /**
        @param   {object} args - an empty map object
        @returns {string} the objectID of the map or an error message
    **/
    addMap: async (_, args) => {

    },

    /**
        @param   {object} args - a map id, region id and subregion objectID
        @returns {array} the updated subregion array on success or the initial
                         array on failure
    **/
    deleteRegion: async (_, args) => {

    },

    /**
        @param   {object} args - a map objectID
        @returns {boolean} true on successful delete, false on failure
    **/
    deleteMap: async (_, args) => {

    },
```



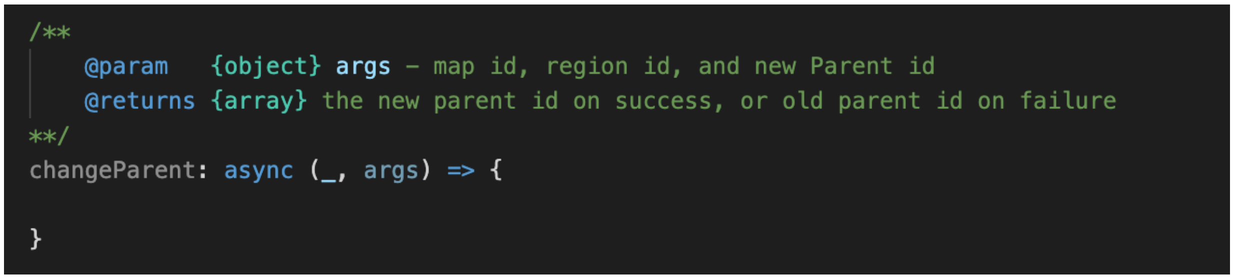
```
/**
    @param   {object} args - a map id, field, and the update value
    @returns {boolean} true on successful update, false on failure
**/
updateMapField: async (_, args) => {

},

/**
    @param   {object} args - a map id, region id, field, and update value.
    @returns {array} the updated subregion array on success, or the initial array on failure
**/
updateRegionField: async (_, args) => {

},

/**
    @param   {object} args - a map id, region id and field
    @returns {array} the updated subregion array on success, or the initial array on failure
**/
sortRegions: async (_, args) => {

},

/**
    @param   {object} args - a map id, region id and subregion array
    @returns {array} the updated subregion array on success, or the initial array on failure
**/
unsortRegions: async (_, args) => {

},
```



```
/**
    @param   {object} args - a map id, region id and landmark string
    @returns {array} the updated landmarks array on success, or the old array
**/
addLandmark: async (_, args) => {

},

/**
    @param   {object} args - a map id, region id and landmark string
    @returns {array} the updated landmarks array on success, or the old array
**/
deleteLandmark: async (_, args) => {

},
```



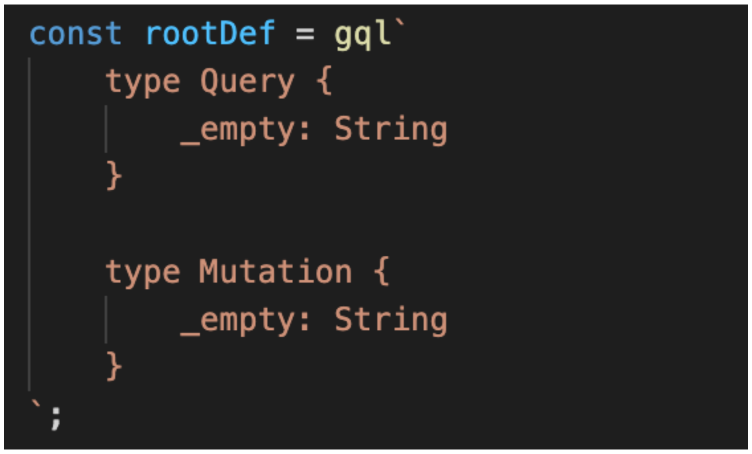
```
/**
    @param   {object} args - a map id, region id, old and updated landmark string
    @returns {array} the updated landmark on success, or the old landmark
**/
editLandmark: async (_, args) => {

},
```



```
/**
    @param   {object} args - map id, region id, and new Parent id
    @returns {array} the new parent id on success, or old parent id on failure
**/
changeParent: async (_, args) => {

}
```

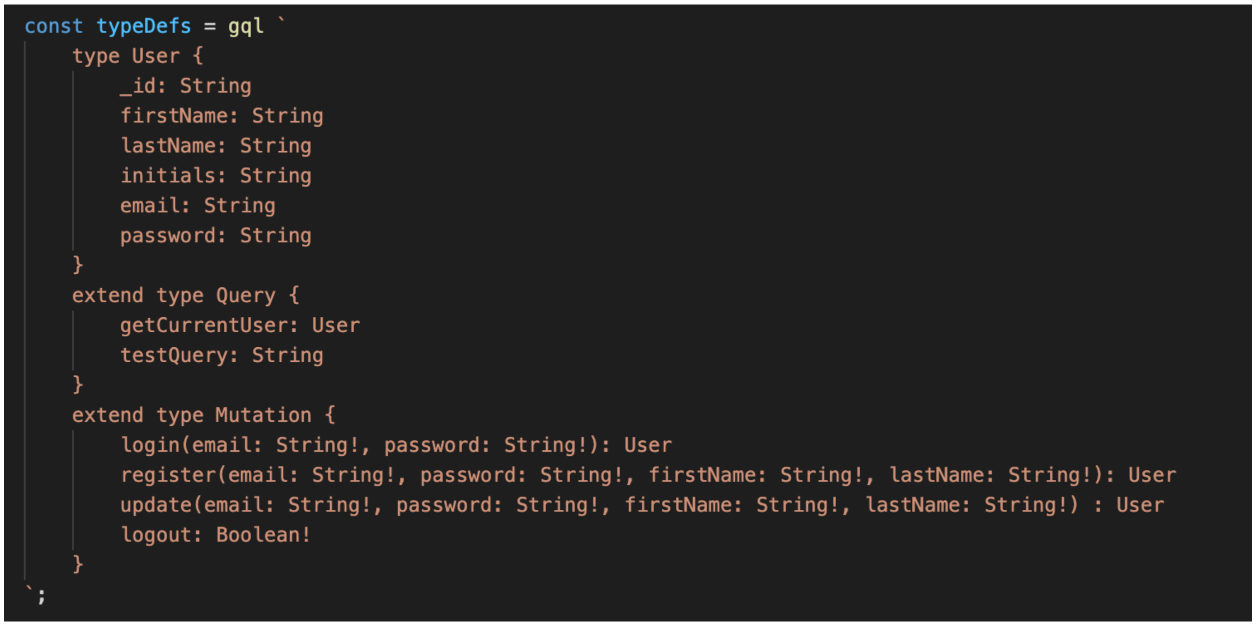
5. Typedefs

Root Def



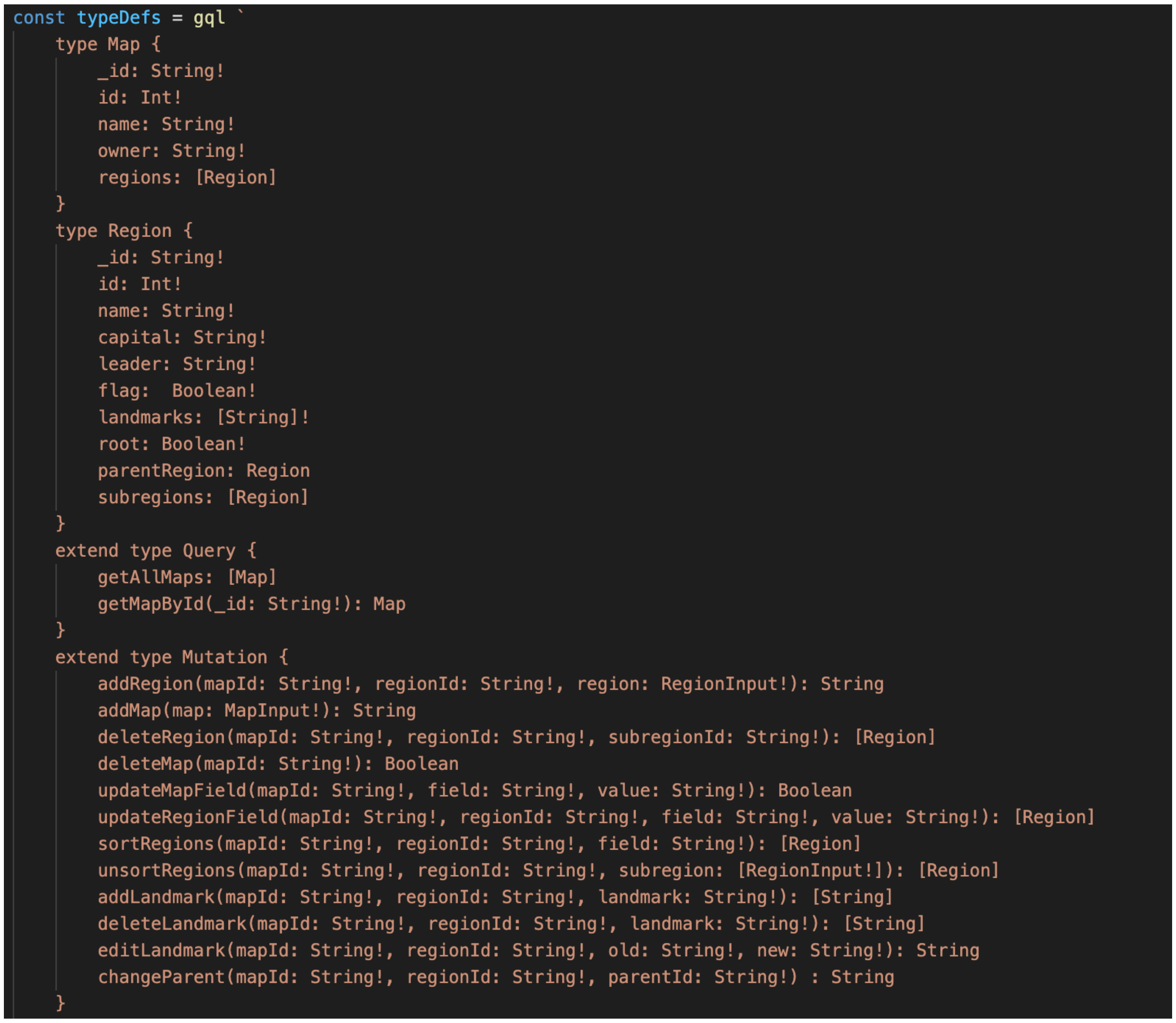
```
const rootDef = gql`
    type Query {
        _empty: String
    }

    type Mutation {
        _empty: String
    }
`;
```

User Def

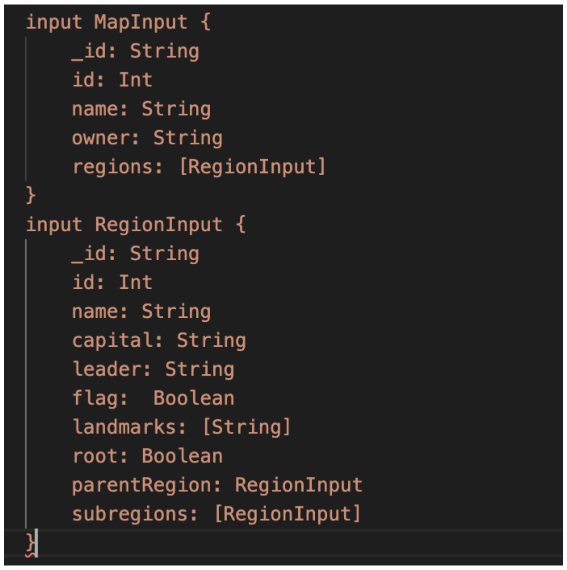


```
const typeDefs = gql `
    type User {
        _id: String
        firstName: String
        lastName: String
        initials: String
        email: String
        password: String
    }
    extend type Query {
        getCurrentUser: User
        testQuery: String
    }
    extend type Mutation {
        login(email: String!, password: String!): User
        register(email: String!, password: String!, firstName: String!, lastName: String!): User
        update(email: String!, password: String!, firstName: String!, lastName: String!) : User
        logout: Boolean!
    }
`;
```

Map Def

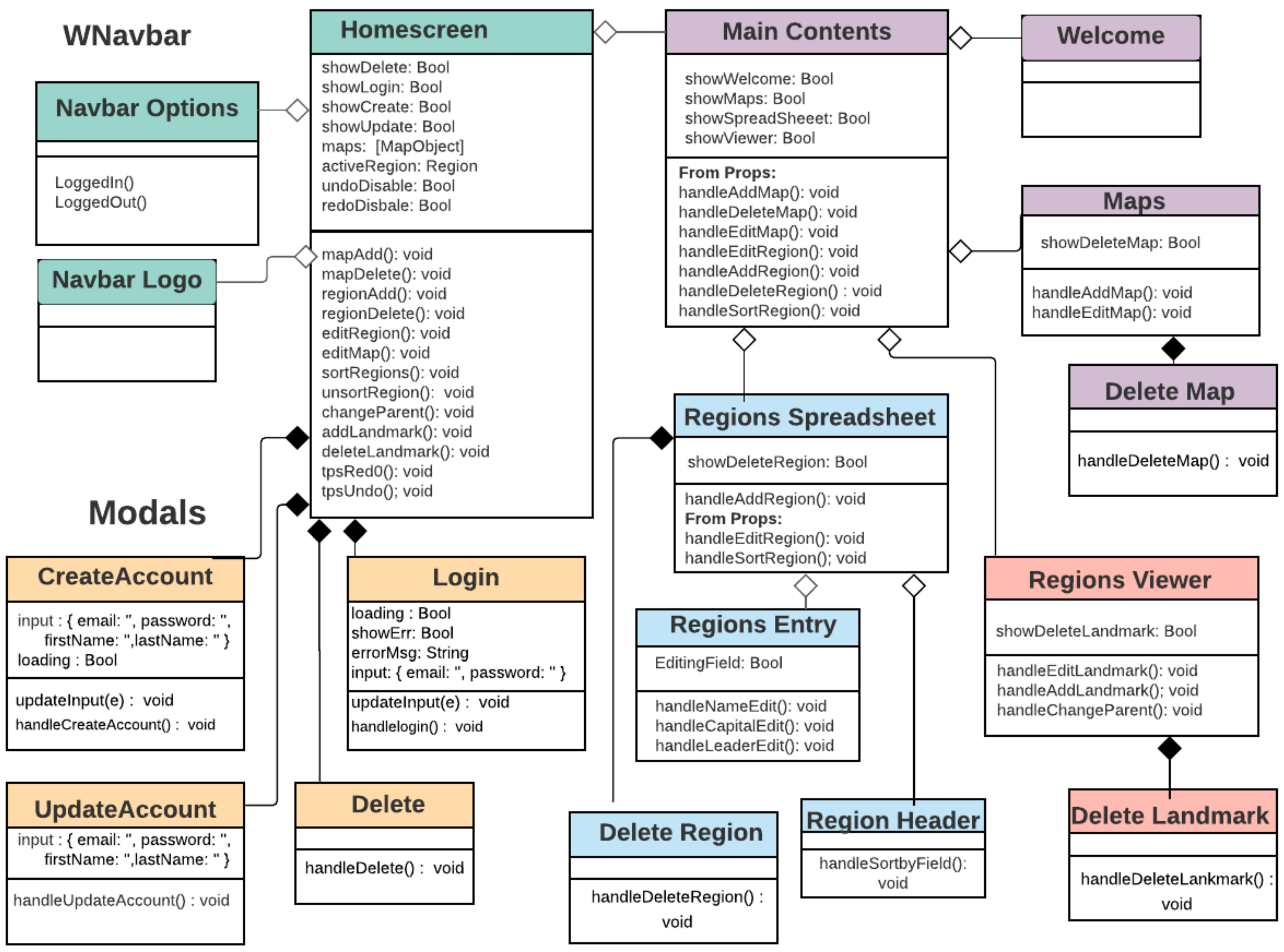


```
const typeDefs = gql `
    type Map {
        _id: String!
        id: Int!
        name: String!
        owner: String!
        regions: [Region]
    }
    type Region {
        _id: String!
        id: Int!
        name: String!
        capital: String!
        leader: String!
        flag:  Boolean!
        landmarks: [String]!
        root: Boolean!
        parentRegion: Region
        subregions: [Region]
    }
    extend type Query {
        getAllMaps: [Map]
        getMapById(_id: String!): Map
    }
    extend type Mutation {
        addRegion(mapId: String!, regionId: String!, region: RegionInput!): String
        addMap(map: MapInput!): String
        deleteRegion(mapId: String!, regionId: String!, subregionId: String!): [Region]
        deleteMap(mapId: String!): Boolean
        updateMapField(mapId: String!, field: String!, value: String!): Boolean
        updateRegionField(mapId: String!, regionId: String!, field: String!, value: String!): [Region]
        sortRegions(mapId: String!, regionId: String!, field: String!): [Region]
        unsortRegions(mapId: String!, regionId: String!, subregion: [RegionInput!]): [Region]
        addLandmark(mapId: String!, regionId: String!, landmark: String!): [String]
        deleteLandmark(mapId: String!, regionId: String!, landmark: String!): [String]
        editLandmark(mapId: String!, regionId: String!, old: String!, new: String!): String
        changeParent(mapId: String!, regionId: String!, parentId: String!) : String
    }
```



```
input MapInput {
    _id: String
    id: Int
    name: String
    owner: String
    regions: [RegionInput]
}
input RegionInput {
    _id: String
    id: Int
    name: String
    capital: String
    leader: String
    flag:  Boolean
    landmarks: [String]
    root: Boolean
    parentRegion: RegionInput
    subregions: [RegionInput]
}
```

6. React Components


WNavbar
Navbar Options
LoggedIn()
LoggedOut()
Navbar Logo
Homescreen
showDelete: Bool
showLogin: Bool
showCreate: Bool
showUpdate: Bool
maps: [MapObject]
activeRegion: Region
undoDisable: Bool
redoDisbale: Bool
mapAdd(): void
mapDelete(): void
regionAdd(): void
regionDelete(): void
editRegion(): void
editMap(): void
sortRegions(): void
unsortRegion(): void
changeParent(): void
addLandmark(): void
deleteLandmark(): void
tpsRed0(): void
tpsUndo(); void
Main Contents
showWelcome: Bool
showMaps: Bool
showSpreadSheeet: Bool
showViewer: Bool
From Props:
handleAddMap(): void
handleDeleteMap(): void
handleEditMap(): void
handleEditRegion(): void
handleAddRegion(): void
handleDeleteRegion() : void
handleSortRegion(): void
Welcome
Maps
showDeleteMap: Bool
handleAddMap(): void
handleEditMap(): void
Delete Map
handleDeleteMap() : void
Regions Spreadsheet
showDeleteRegion: Bool
handleAddRegion(): void
From Props:
handleEditRegion(): void
handleSortRegion(); void
Regions Entry
EditingField: Bool
handleNameEdit(): void
handleCapitalEdit(): void
handleLeaderEdit(): void
Regions Viewer
showDeleteLandmark: Bool
handleEditLandmark(): void
handleAddLandmark(); void
handleChangeParent(): void
Delete Landmark
handleDeleteLankmark() : void
Region Header
handleSortbyField(): void
Delete Region
handleDeleteRegion() : void
Modals
CreateAccount
input : { email: '', password: '', firstName: '',lastName: '' }
loading : Bool
updateInput(e) : void
handleCreateAccount() : void
Login
loading : Bool
showErr: Bool
errorMsg: String
input: { email: '', password: '' }
updateInput(e) : void
handlelogin() : void
UpdateAccount
input : { email: '', password: '', firstName: '',lastName: '' }
handleUpdateAccount() : void
Delete
handleDelete() : void