CSE 316
Hak Soo Kim (111045936)

1. User Interface Mockup Diagrams

The application will have 13 user interface contexts/views.

View Number	Description
1.1	Welcome Screen Context
1.2	Create Account Screen Context
1.3	Login Screen Context
1.4	Update Account Screen Context
1.5	Map Select Screen Context
1.6	Create New Map Screen Context
1.7	Update Map Name Screen Context
1.8	Map Deletion Verification Screen Context
1.9	Regions Spreadsheet Screen Context
1.10	Regions Deletion Verification Screen Context
1.11	Region Viewer Screen Context
1.12	Update Region Name Screen Context
1.13	Moving a subregion to a different parent region Screen Context

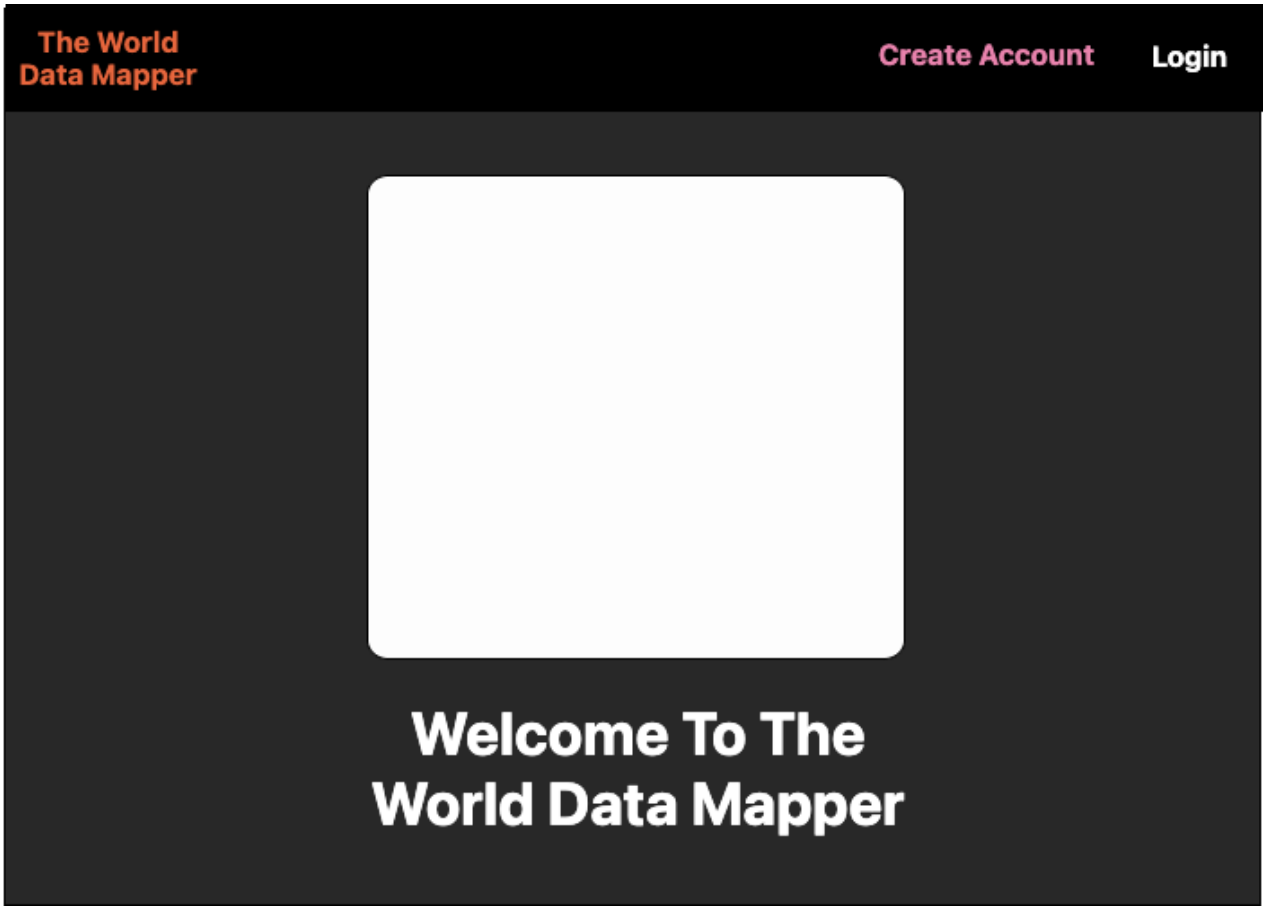


Figure 1.1: Welcome Screen Context

Figure 1.2: Create Account Screen Context

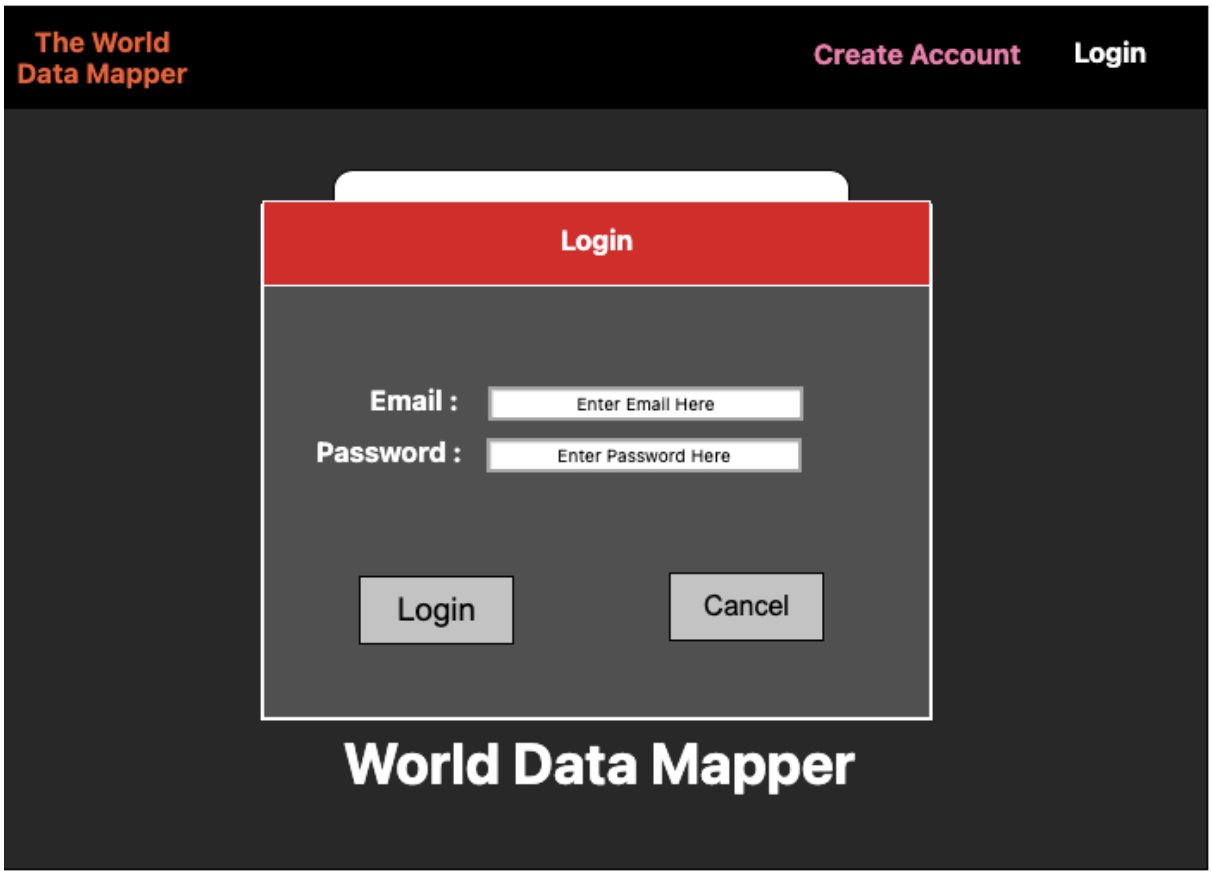


Figure 1.3: Login Screen Context

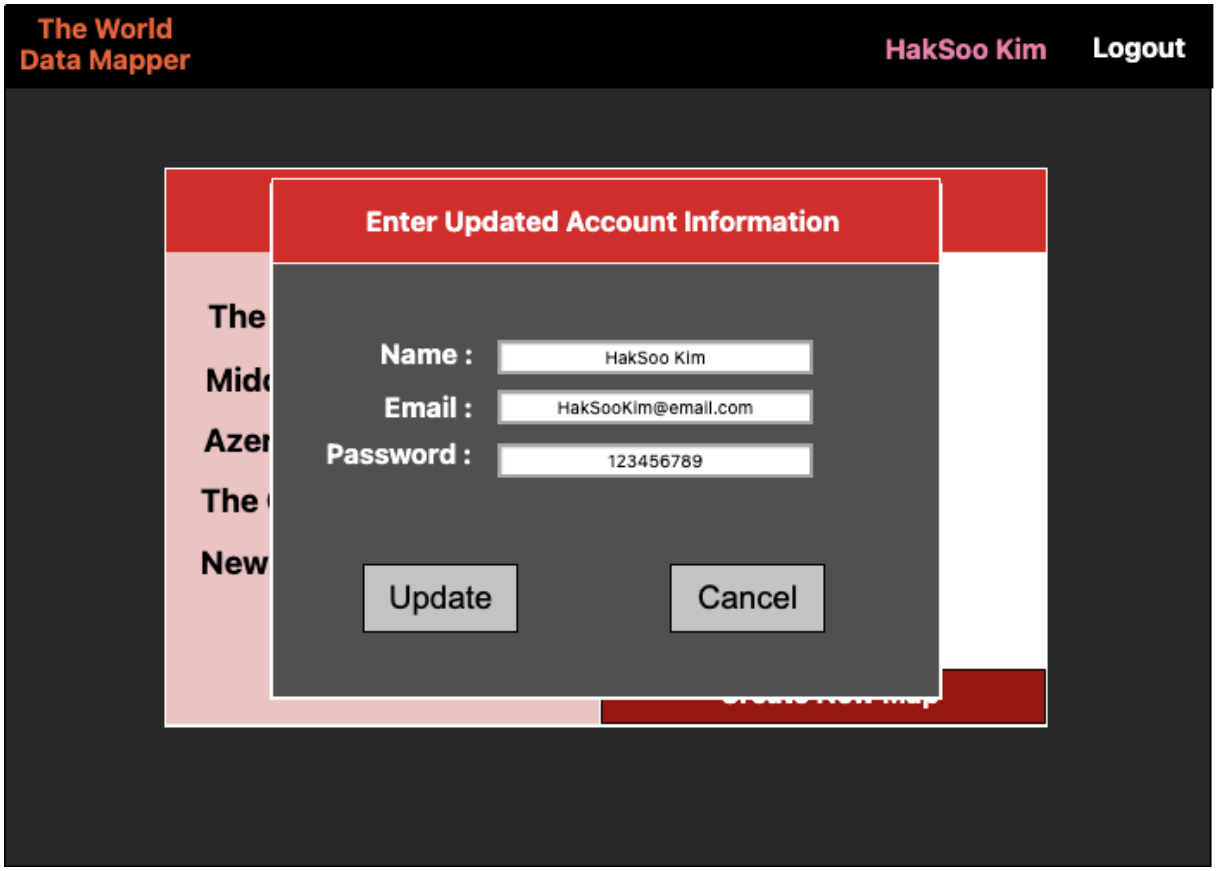


Figure 1.4: Update Account Screen Context

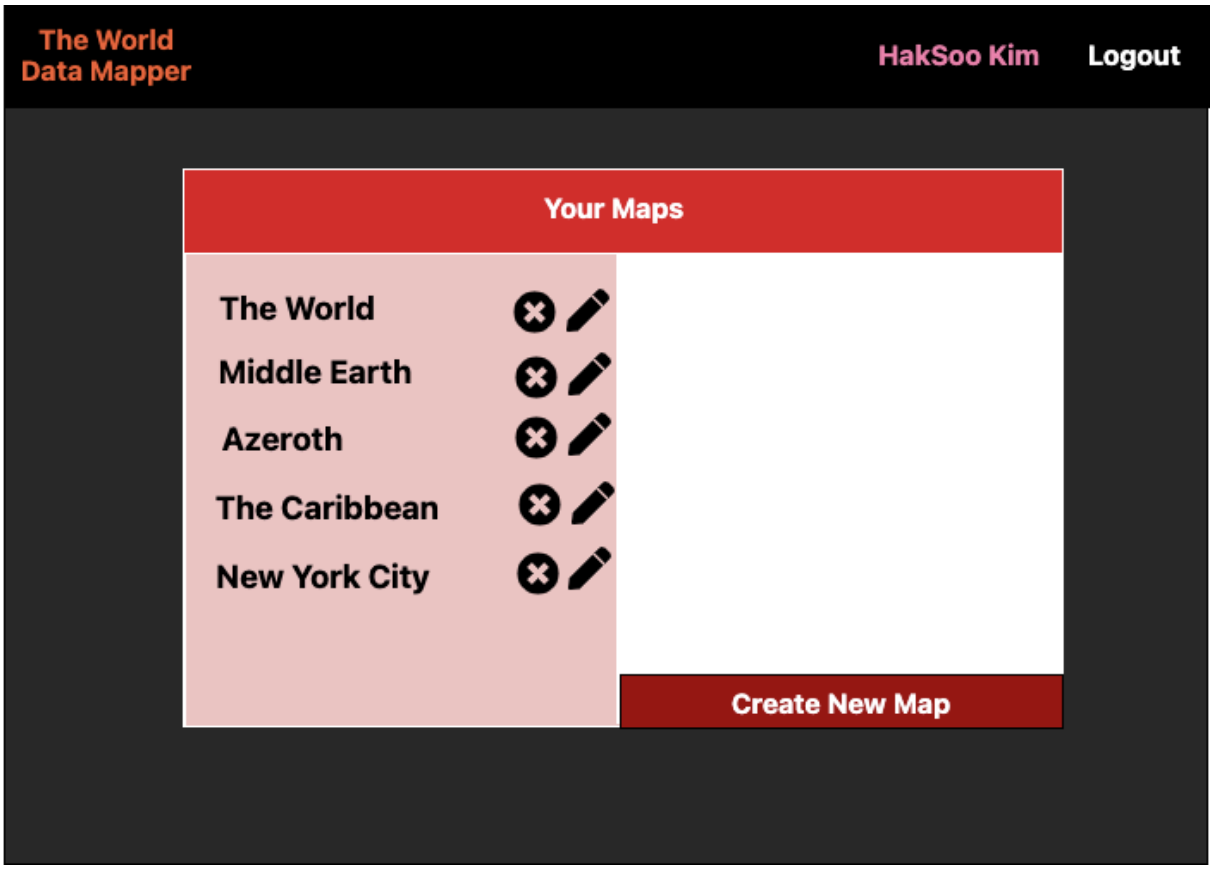


Figure 1.5: Map Select Screen Context

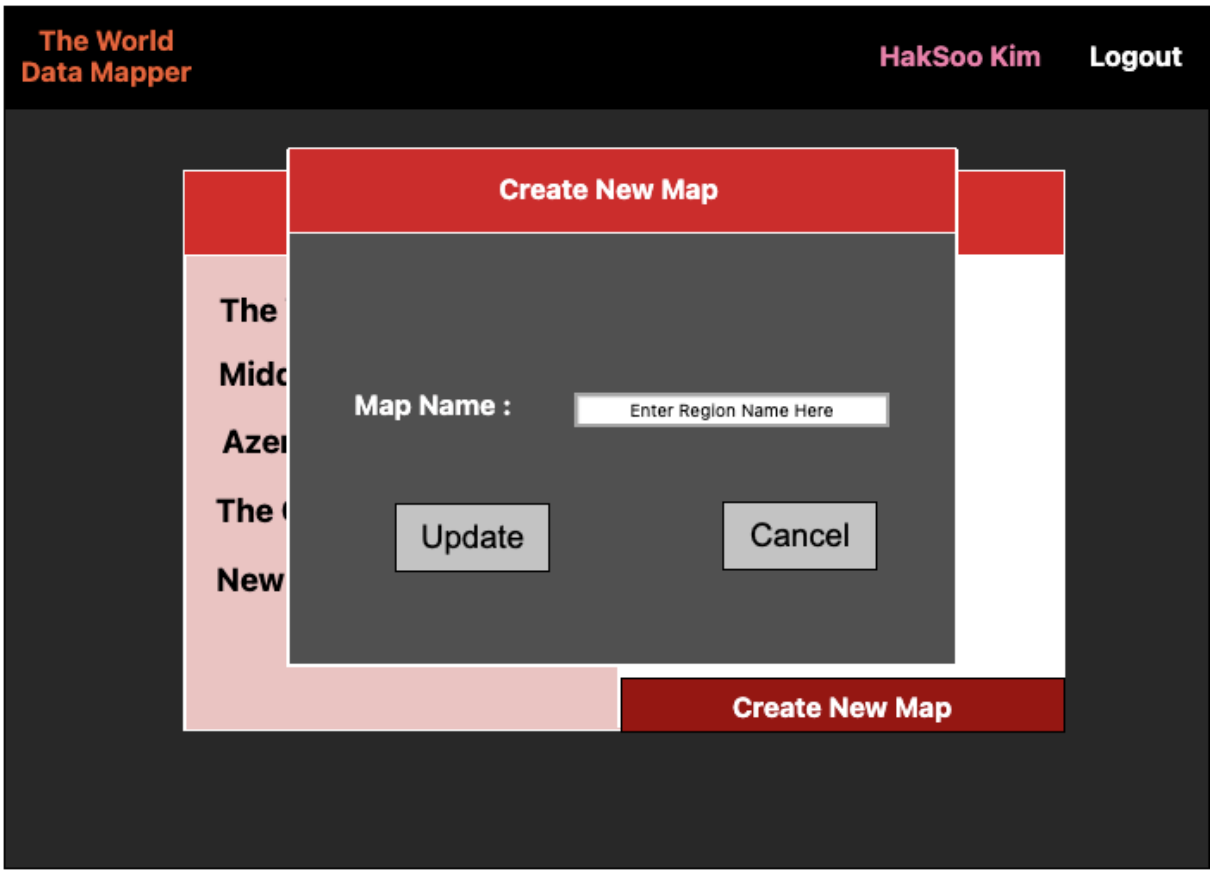


Figure 1.6: Create New Map Screen Context

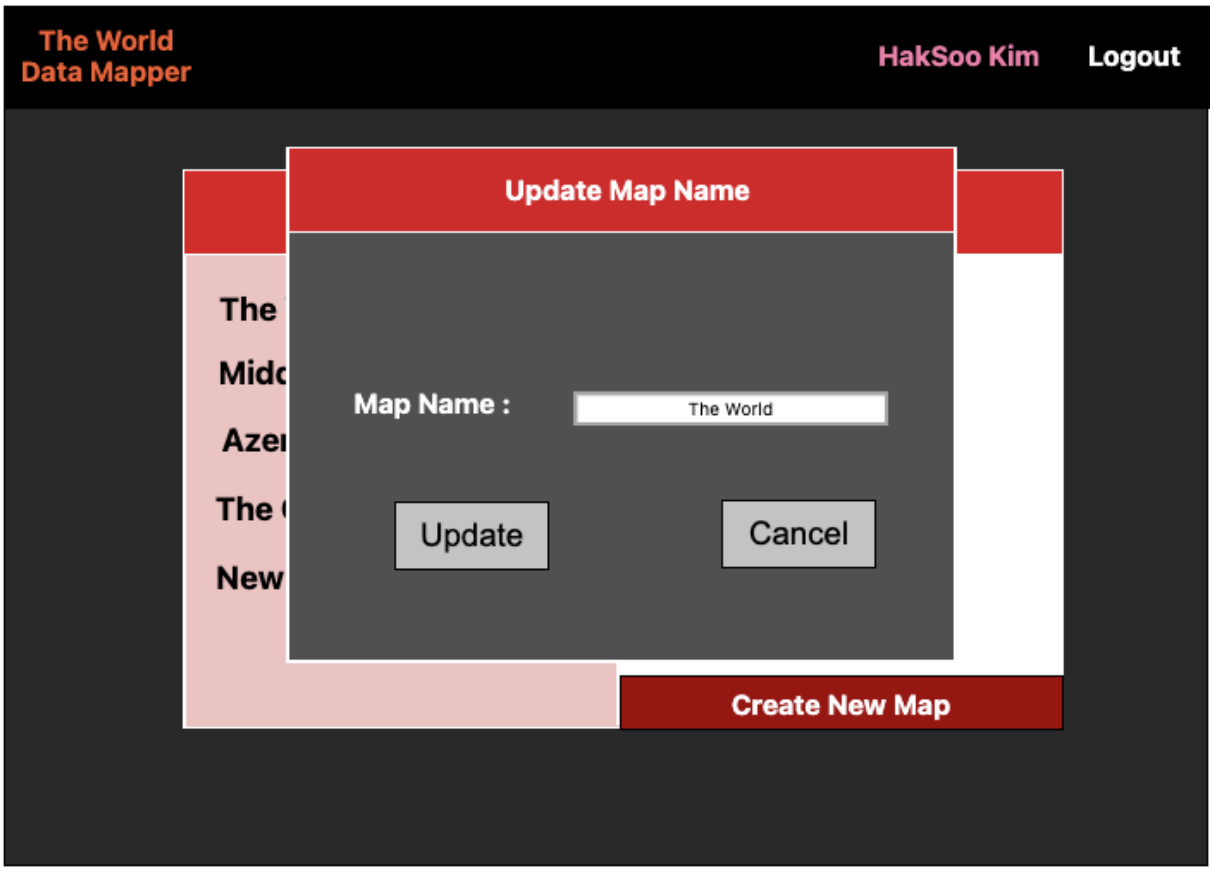


Figure 1.7: Update Map Name Screen Context

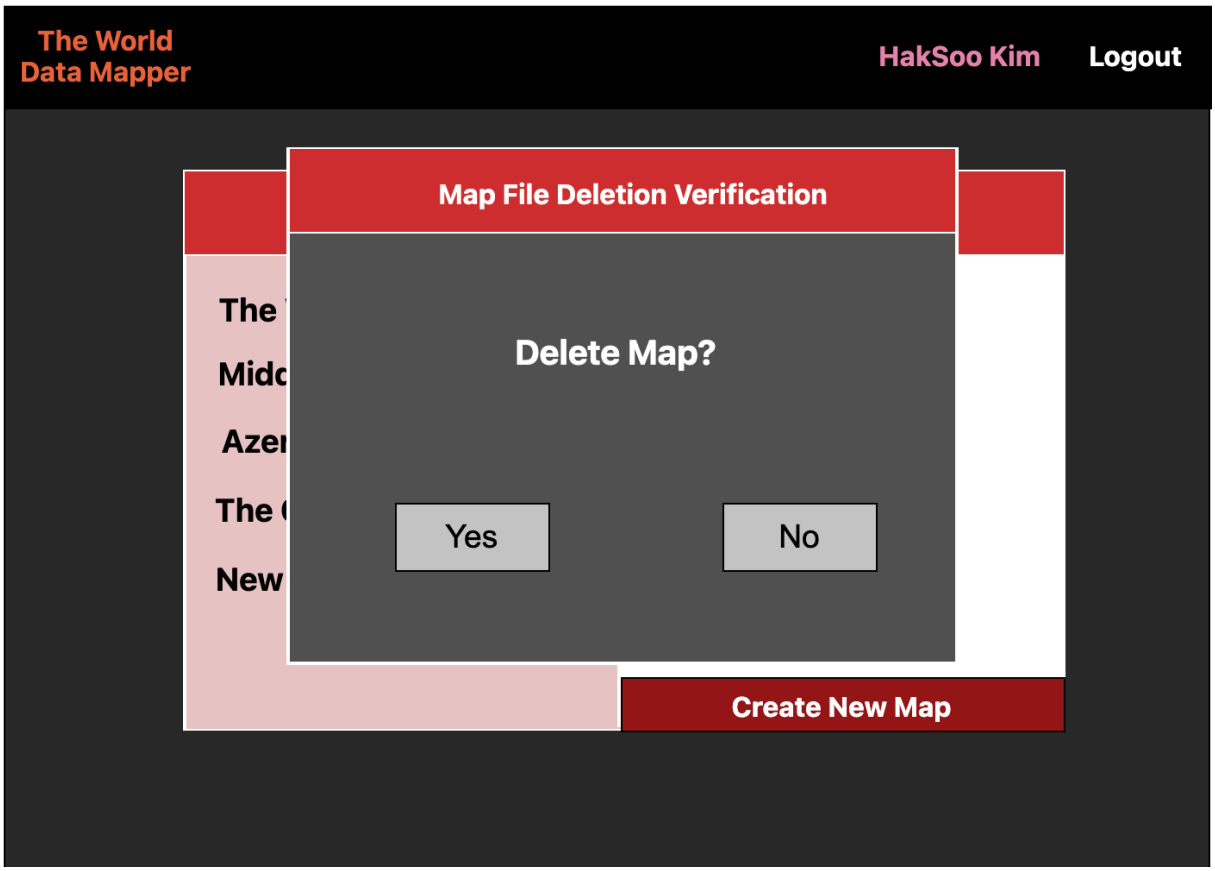


Figure 1.8: Map Deletion Verification Screen Context

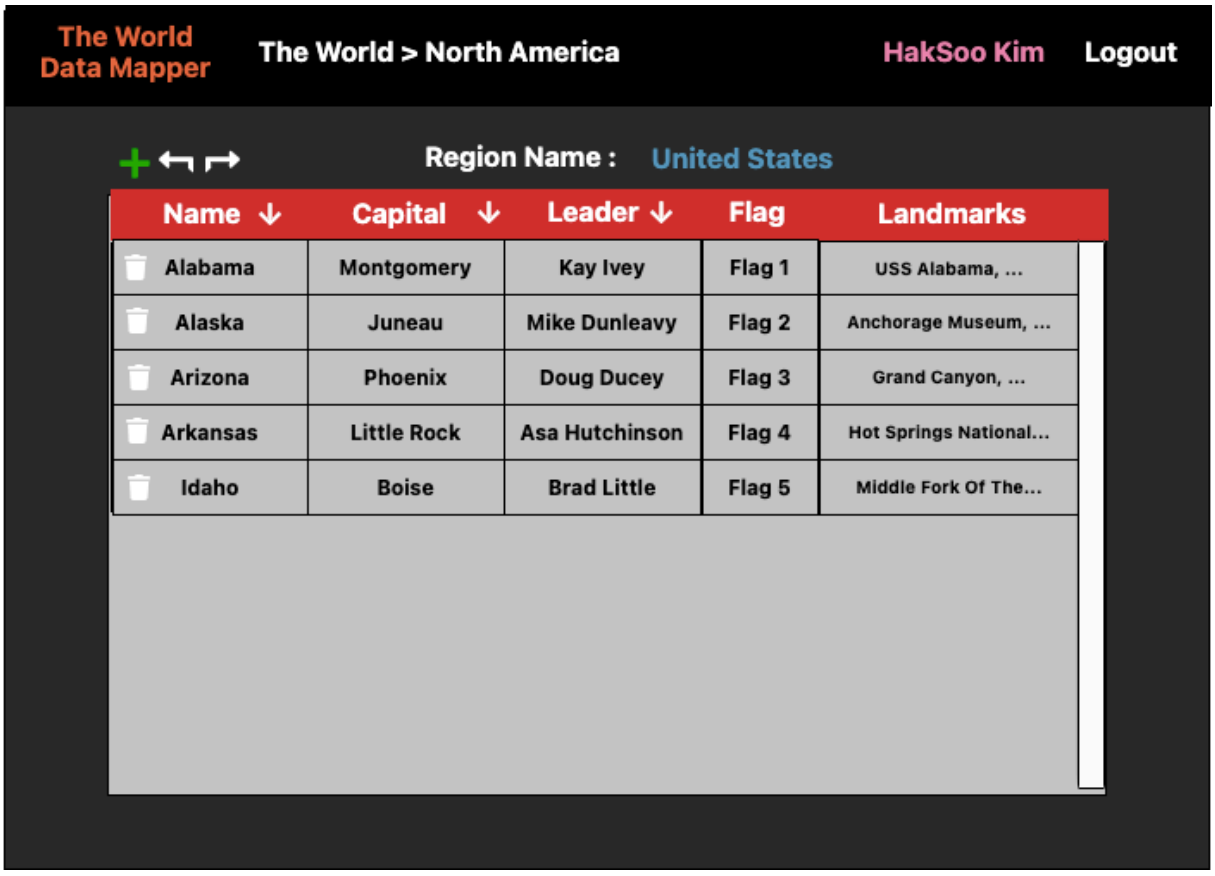


Figure 1.9: Regions Spreadsheet Screen Context

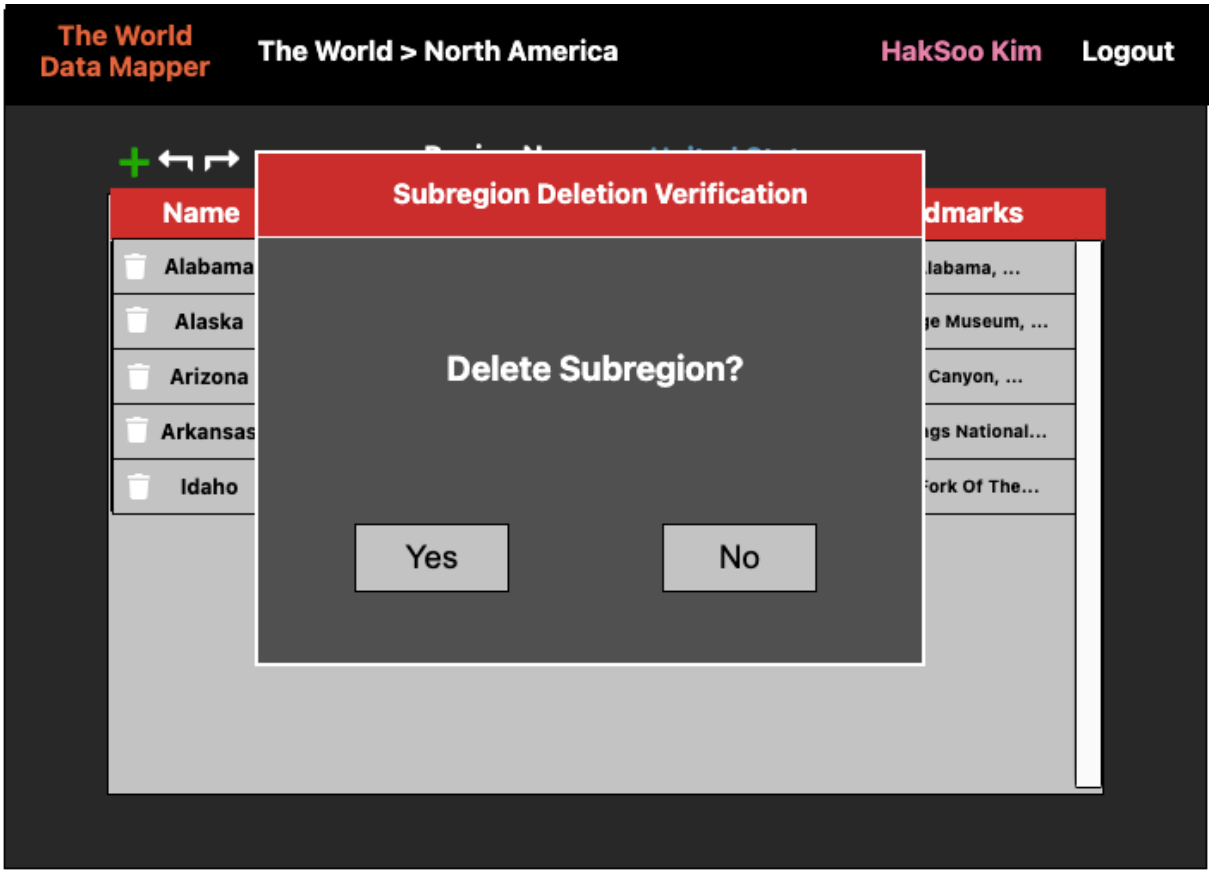


Figure 1.10: Regions Deletion Verification Screen Context

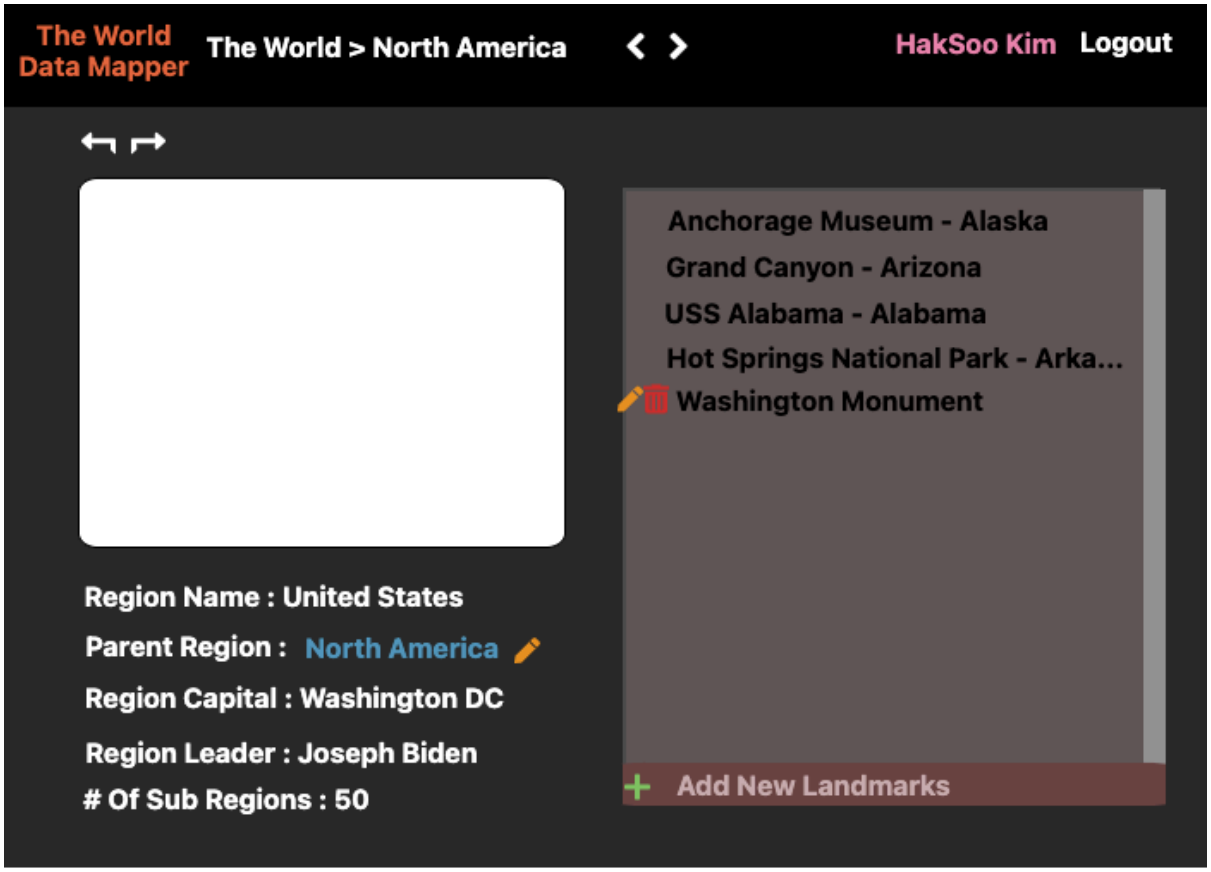


Figure 1.11: Region Viewer Screen Context

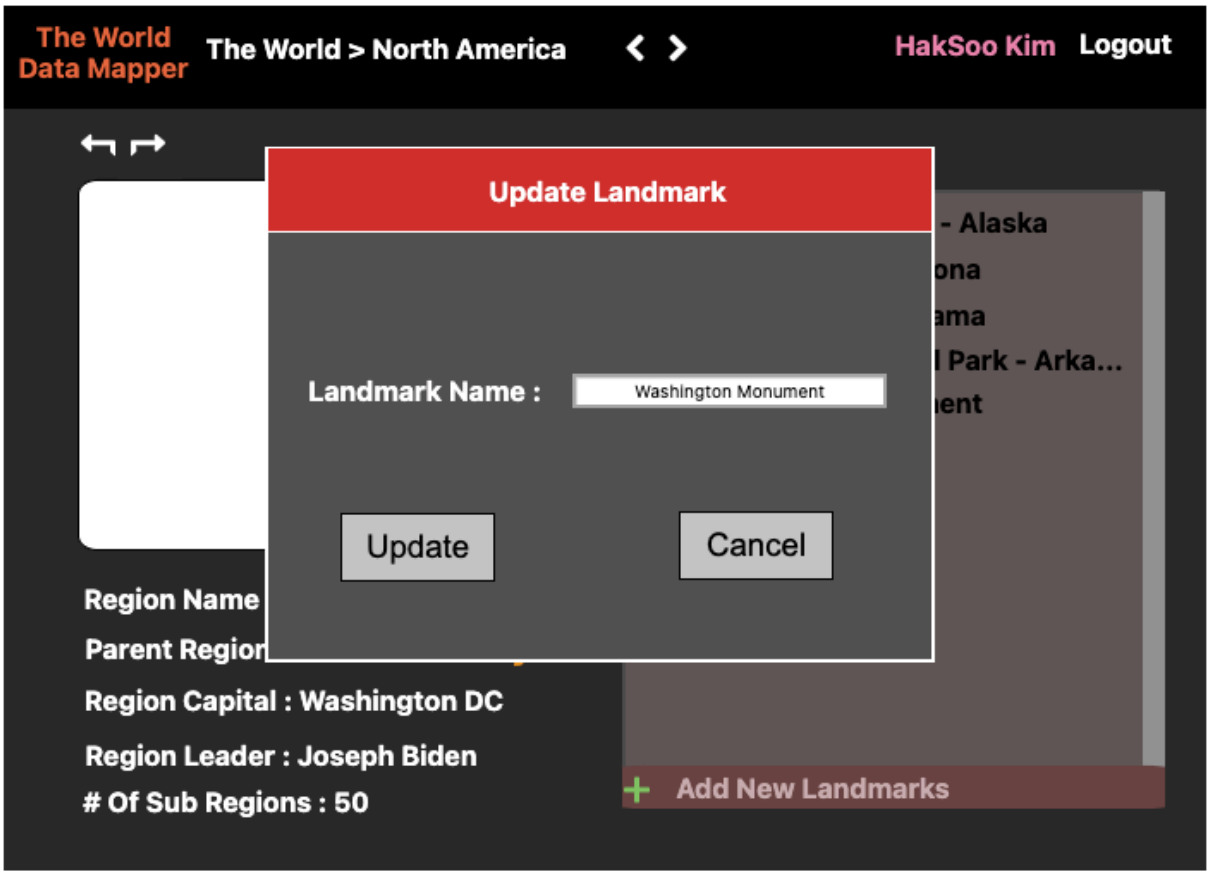


Figure 1.12: Update Region Name Screen Context

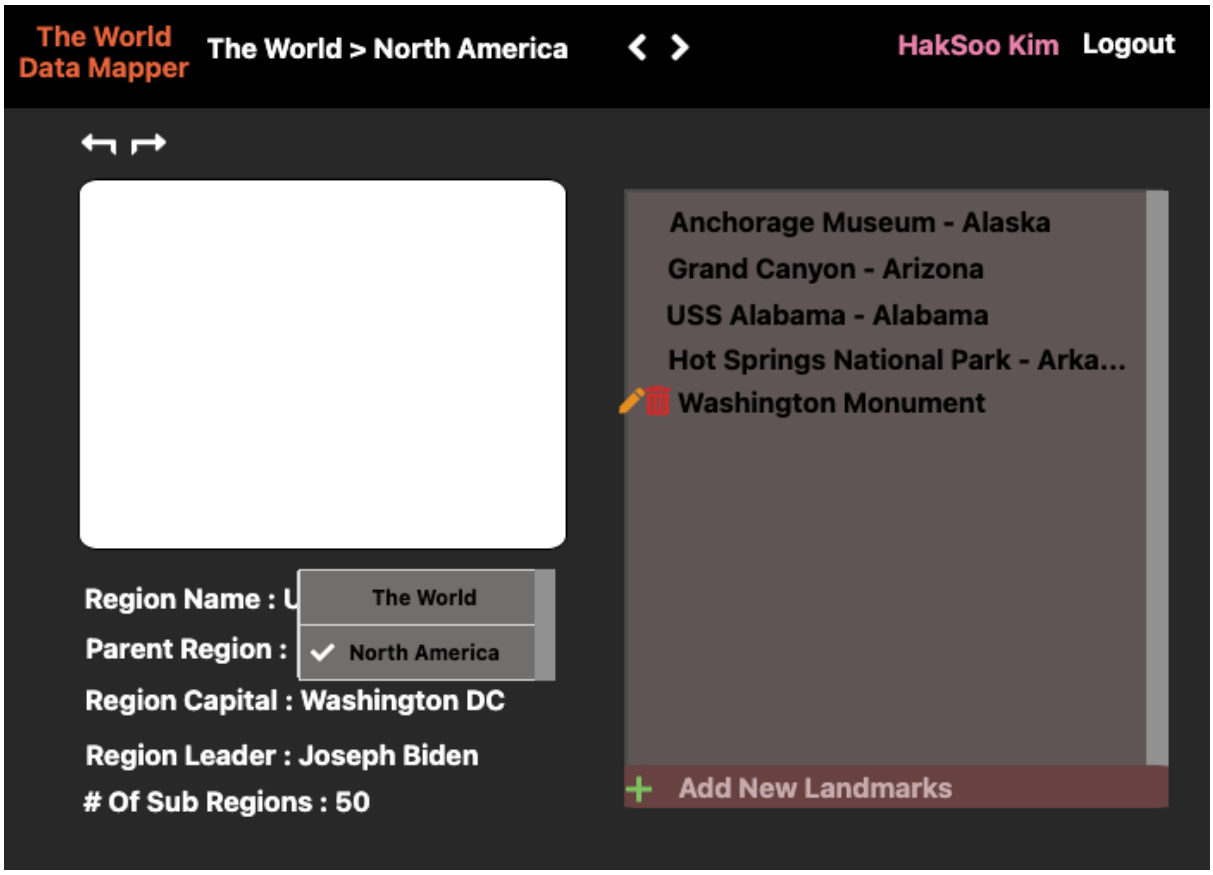


Figure 1.13: Moving a subregion to a different parent region

2.

/home/welcome

/home/:User_id

/home/:User_id/:Region_id

/RegionViewer/:Region_id

3.

1) Region Schema

```
const { model, Schema, ObjectId } = require('mongoose');

const regionSchema = new Schema(
    {
        _id: {
            type: ObjectId,
            required: true
        },
        owner: {
            type: String,
            required: true
        },
        name: {
            type: String,
            required: true
        },
        capital: {
            type: String,
            required: true
        },
        leader: {
            type: String,
            required: true
        },
        flag: {
            type: String,
            required: true
        },
        landmark:{
            type: [String],
            required: true
        },
        parentRegion_id: {
            type: String,
            requried: true
        },
        top:{
            type: Boolean,
            required: true
        }
        // subRegion:{
        //     type: [Region]
        // }
    },
    { timestamps: true }
);
const Region = model('Region', regionSchema);
module.exports = Region;
```

2) User Schema

```
const { model, Schema, ObjectId } = require('mongoose');

const userSchema = new Schema(
    {
        _id: {
            type: ObjectId,
            required: true
        },
        name: {
            type: String,
            required: true
        },
        email: {
            type: String,
            required: true
        },
        password: {
            type: String,
            required: true
        }
    },
    { timestamps: true }
);
const User = model('User', userSchema);
module.exports = User;
```

4.

1) Root resolver

```
const userResolvers = require('./user-resolvers');
const regionResolvers = require('./region-resolvers');

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

2) Region resolver

```
const ObjectId = require('mongoose').Types.ObjectId;
const Region = require('../models/region-model');

module.exports = {
    Query: {

        getRootRegionsByUserId: async (_, __, { req }) => {

        },

        getRegionById: async (_, args) => {

        },
        getSubRegionsById: async (_, args) => {

        },
        getParentRegionById: async (_, args) => {

        },

    },
    Mutation: {

        deleteReigion: async(_, args) => {

        },

        updateReigionField: async (_, args) => {

        },

        addRegion: async (_, args) => {


        },

        selectedRootregionTop: async (_, args) => {

        },

        moveParentregion: async (_, args) => {

        },
    }
}
```

3) User resolver

```
const ObjectId = require('mongoose').Types.ObjectId;
const bcrypt = require('bcryptjs');
const User = require('../models/user-model');
const tokens = require('../utils/tokens');

module.exports = {
    Query: {
        getCurrentUser: async (_, __, { req }) => {
        },
    },
    Mutation: {
        login: async (_, args, { res }) => {
        },
        register: async (_, args, { res }) => {
        },
        updateAccount: async (_, args, { res }) => {
        },
        logout:(_, __, { res }) => {
        }
    }
}

```

5.

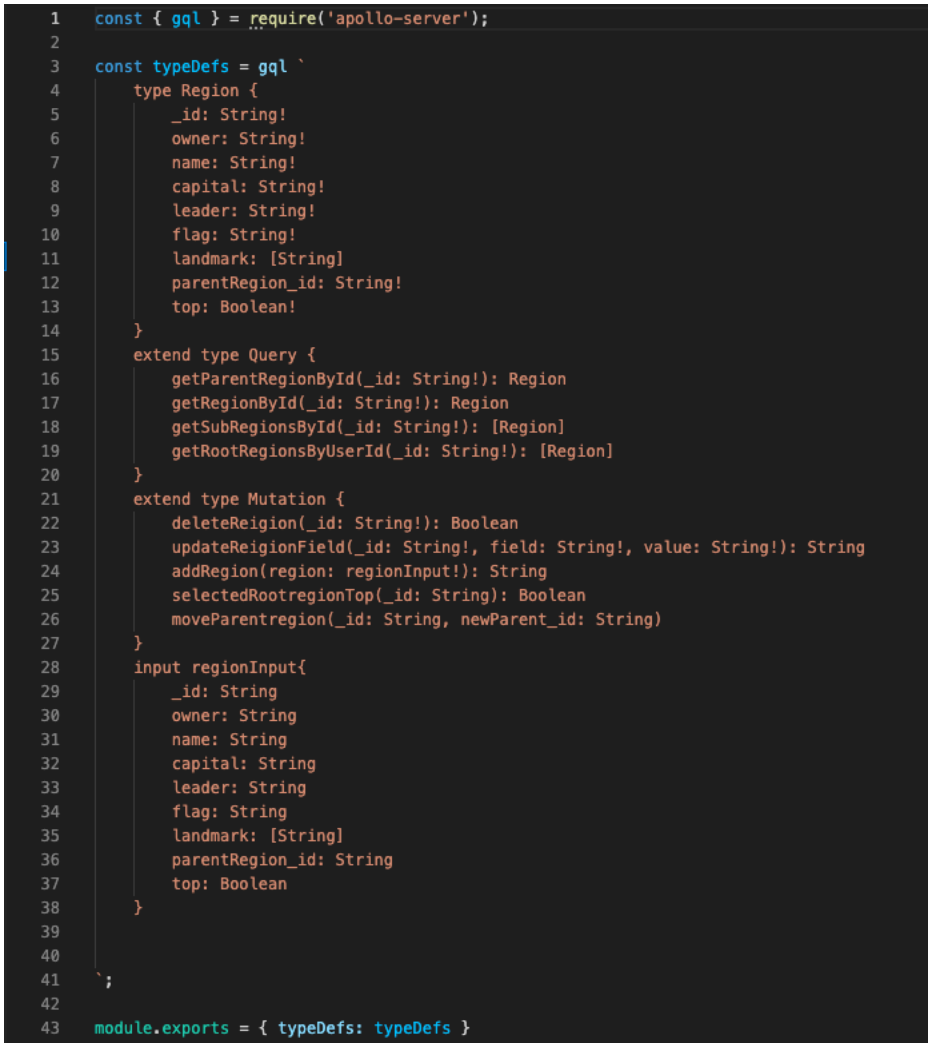
1.) Root-typedef

```
const { gql } = require('apollo-server');
const userDef = require('./user-def').typeDefs;
const regionDef = require('./region-def').typeDefs;

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

    type Mutation {
        _empty: String
    }
`;

module.exports = {
    typeDefs: [rootDef, userDef, regionDef]
};
```

2.) User-typedef

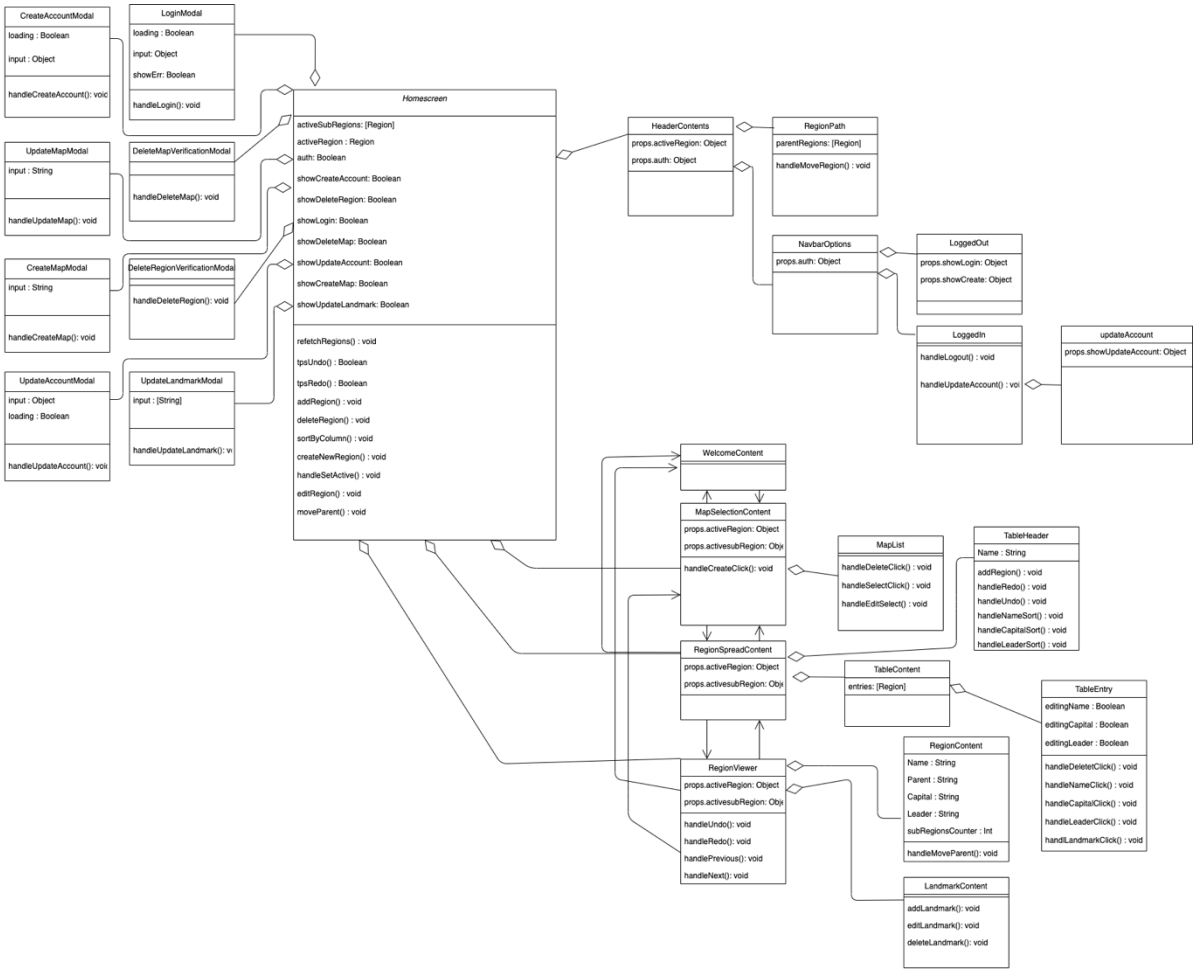
```
const { gql } = require('apollo-server');

const typeDefs = gql `
    type User {
        _id: String
        name: String
        email: String
        password: String
    }
    extend type Query {
        getCurrentUser: User
    }
    extend type Mutation {
        login(email: String!, password: String!): User
        register(email: String!, password: String!, name: String! ): User
        updateAccount(email: String!, password: String!, name: String! ): Boolean!
        logout: Boolean!
    }



`;

module.exports = { typeDefs: typeDefs }
```

3.) Region-typedef



```
const { gql } = require('apollo-server');

const typeDefs = gql `
    type Region {
        _id: String!
        owner: String!
        name: String!
        capital: String!
        leader: String!
        flag: String!
        landmark: [String]
        parentRegion_id: String!
        top: Boolean!
    }
    extend type Query {
        getParentRegionById(_id: String!): Region
        getRegionById(_id: String!): Region
        getSubRegionsById(_id: String!): [Region]
        getRootRegionsByUserId(_id: String!): [Region]
    }
    extend type Mutation {
        deleteReigion(_id: String!): Boolean
        updateReigionField(_id: String!, field: String!, value: String!): String
        addRegion(region: regionInput!): String
        selectedRootregionTop(_id: String): Boolean
        moveParentregion(_id: String, newParent_id: String)
    }
    input regionInput{
        _id: String
        owner: String
        name: String
        capital: String
        leader: String
        flag: String
        landmark: [String]
        parentRegion_id: String
        top: Boolean
    }


`;

module.exports = { typeDefs: typeDefs }
```

6. React Component


CreateAccountModal
loading : Boolean
input : Object
handleCreateAccount(): voic
LoginModal
loading : Boolean
input: Object
showErr: Boolean
handleLogin(): void
UpdateMapModal
input : String
handleUpdateMap(): void
DeleteMapVerificationModal
handleDeleteMap(): void
CreateMapModal
input : String
handleCreateMap(): void
DeleteRegionVerificationModal
handleDeleteRegion(): void
UpdateAccountModal
input : Object
loading : Boolean
handleUpdateAccount(): voi
UpdateLandmarkModal
input : [String]
handleUpdateLandmark(): v
Homescreen
activeSubRegions: [Region]
activeRegion : Region
auth: Boolean
showCreateAccount: Boolean
showDeleteRegion: Boolean
showLogin: Boolean
showDeleteMap: Boolean
showUpdateAccount: Boolean
showCreateMap: Boolean
showUpdateLandmark: Boolean
refetchRegions() : void
tpsUndo() : Boolean
tpsRedo() : Boolean
addRegion() : void
deleteRegion() : void
sortByColumn() : void
createNewRegion() : void
handleSetActive() : void
editRegion() : void
moveParent() : void
HeaderContents
props.activeRegion: Object
props.auth: Object
RegionPath
parentRegions: [Region]
handleMoveRegion() : void
NavbarOptions
props.auth: Object
LoggedOut
props.showLogin: Object
props.showCreate: Object
LoggedIn
handleLogout() : void
handleUpdateAccount() : voi
updateAccount
props.showUpdateAccount: Object
WelcomeContent
MapSelectionContent
props.activeRegion: Object
props.activesubRegion: Obj
handleCreateClick(): void
MapList
handleDeleteClick() : void
handleSelectClick() : void
handleEditSelect() : void
TableHeader
Name : String
addRegion() : void
handleRedo() : void
handleUndo() : void
handleNameSort() : void
handleCapitalSort() : void
handleLeaderSort() : void
RegionSpreadContent
props.activeRegion: Object
props.activesubRegion: Obj
TableContent
entries: [Region]
TableEntry
editingName : Boolean
editingCapital : Boolean
editingLeader : Boolean
handleDeletetClick() : void
handleNameClick() : void
handleCapitalClick() : void
handleLeaderClick() : void
handlLandmarkClick() : void
RegionViewer
props.activeRegion: Object
props.activesubRegion: Obj
handleUndo(): void
handleRedo(): void
handlePrevious(): void
handleNext(): void
RegionContent
Name : String
Parent : String
Capital : String
Leader : String
subRegionsCounter : Int
handleMoveParent(): void
LandmarkContent
addLandmark(): void
editLandmark(): void
deleteLandmark(): void