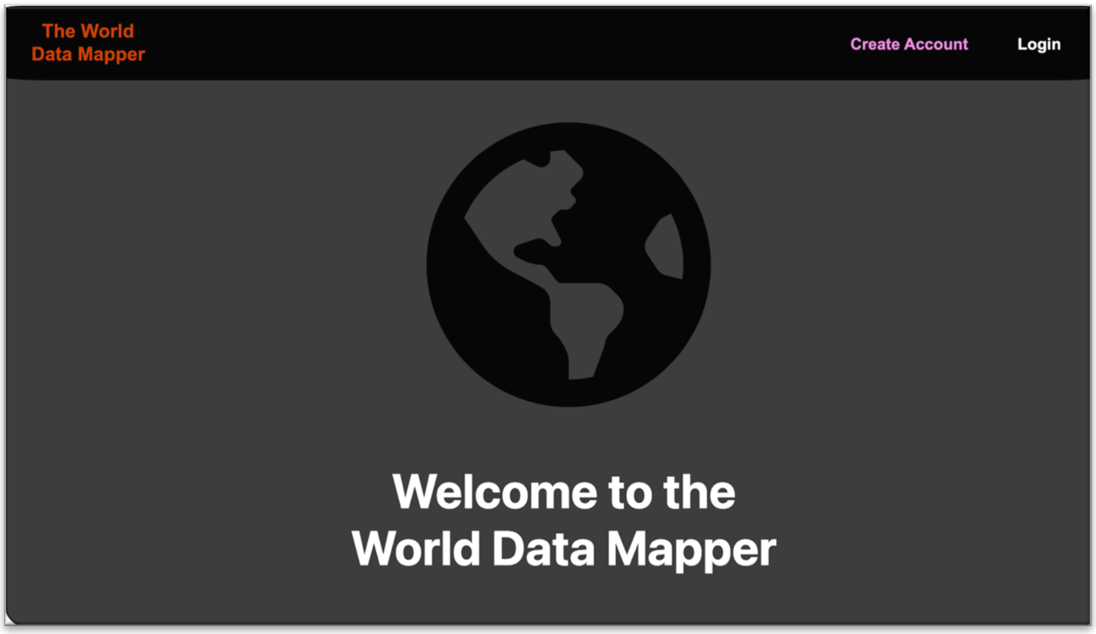
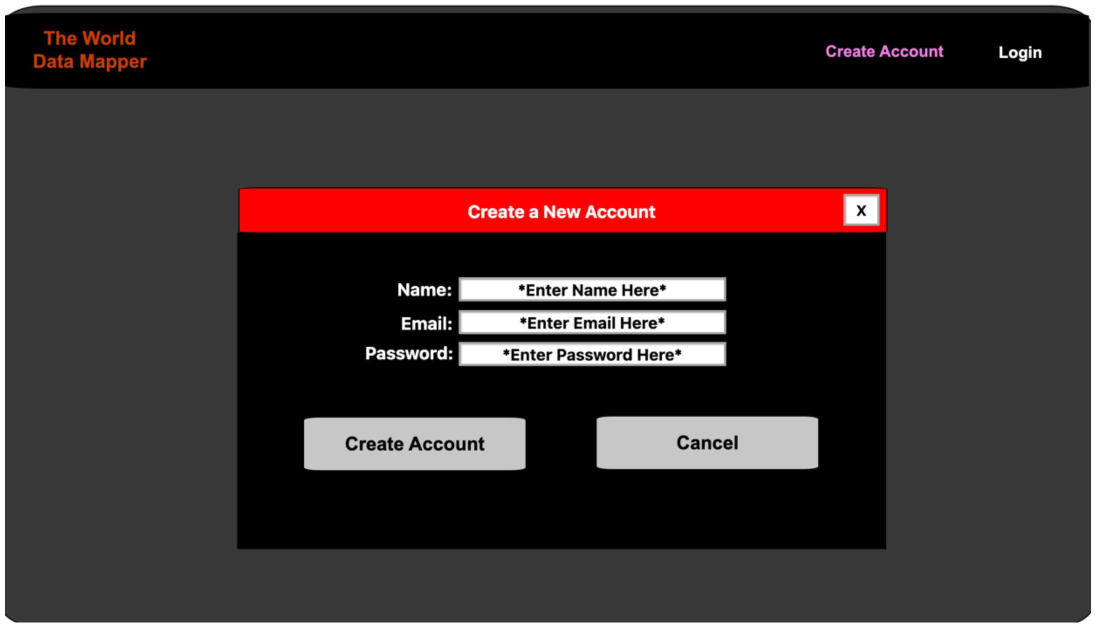
Sidharth Shamshabad

CSE 316: Final Project Mockups & UML Design

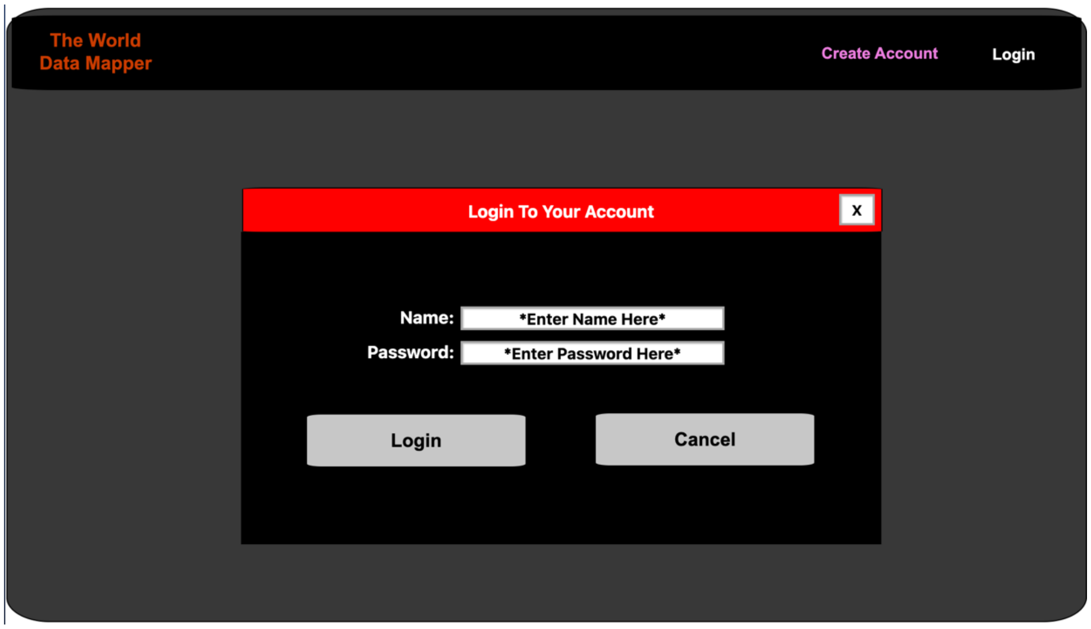
1. UI Mockup Diagram
    a. Welcome Screen Context



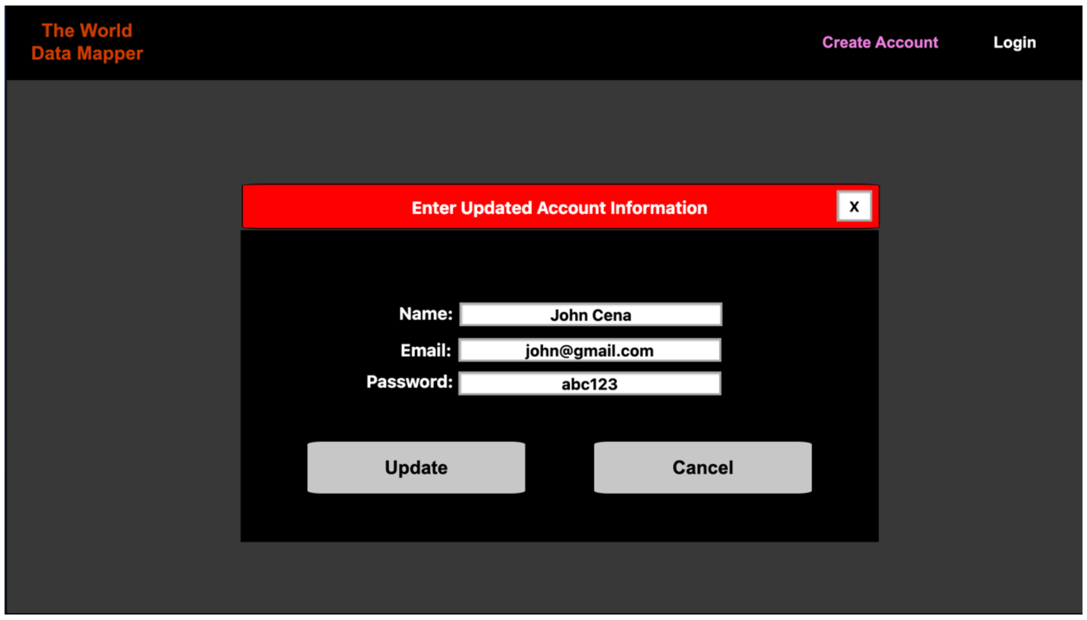
    b. Create Account Screen Context



c. Login Screen Context

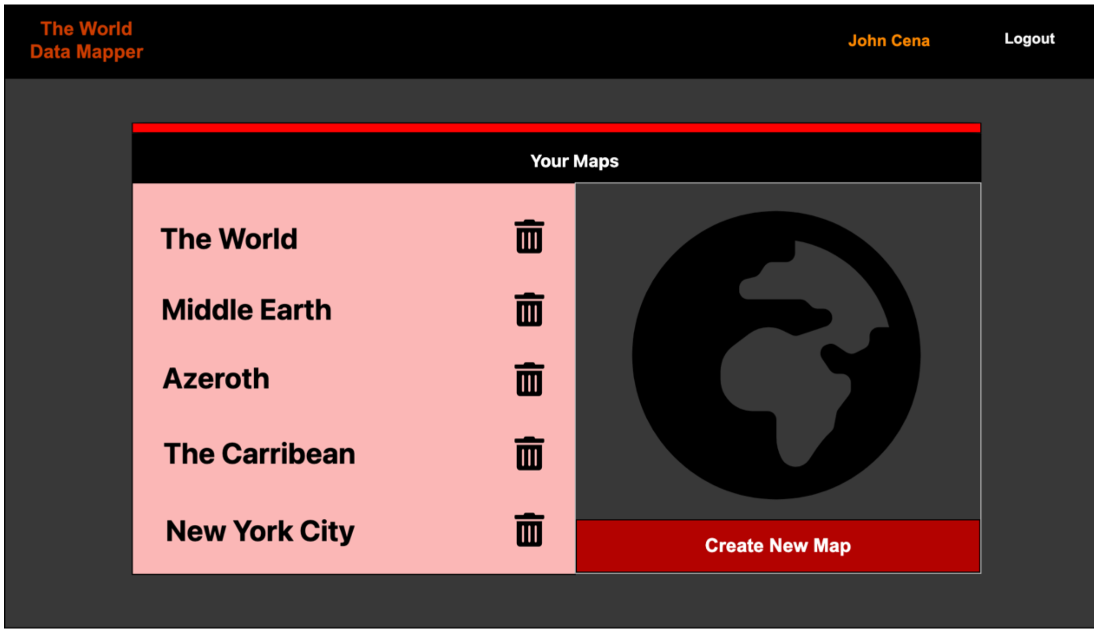


d. Update Screen Context



Name: John Cena
Email: john@gmail.com
Password: abc123

e. Map Select Screen Context



The World Data Mapper

John Cena

Logout

Your Maps

The World

Middle Earth

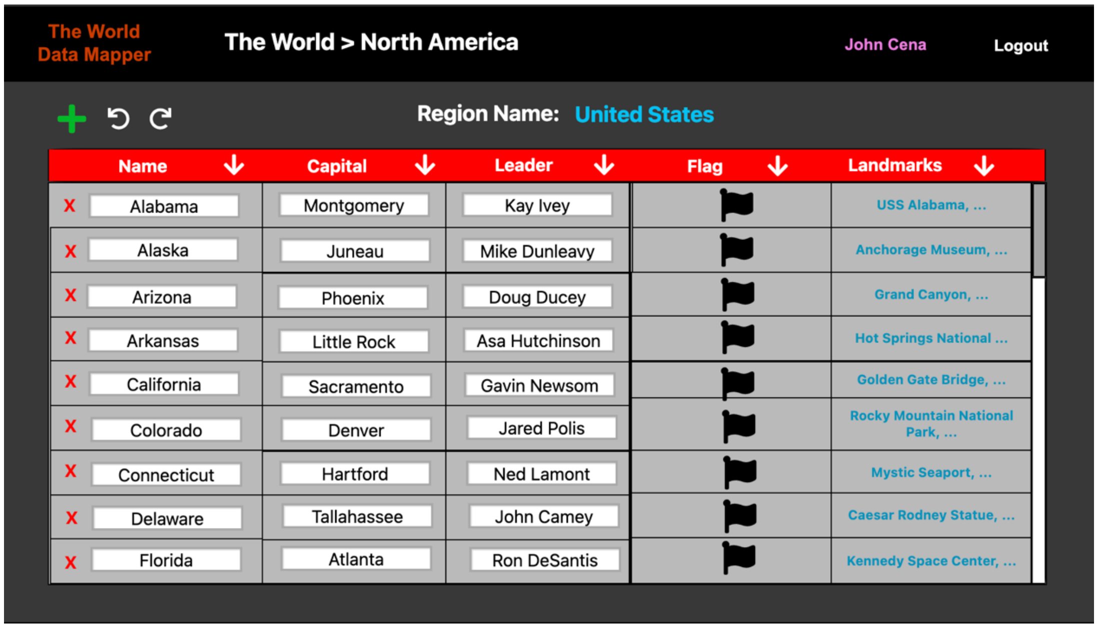
Azeroth

The Carribean

New York City

Create New Map

f. Regions Spreadsheet Screen Context



The World Data Mapper

The World > North America

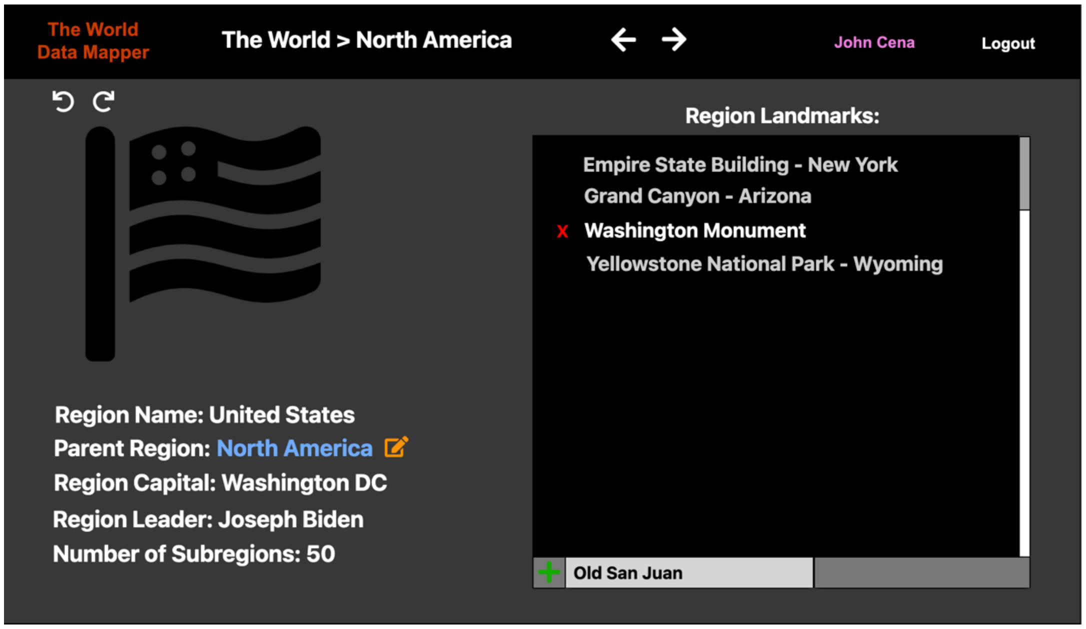
John Cena

Logout

Region Name: United States

	Name	Capital	Leader	Flag	Landmarks
X	Alabama	Montgomery	Kay Ivey	⚑	USS Alabama, ...
X	Alaska	Juneau	Mike Dunleavy	⚑	Anchorage Museum, ...
X	Arizona	Phoenix	Doug Ducey	⚑	Grand Canyon, ...
X	Arkansas	Little Rock	Asa Hutchinson	⚑	Hot Springs National ...
X	California	Sacramento	Gavin Newsom	⚑	Golden Gate Bridge, ...
X	Colorado	Denver	Jared Polis	⚑	Rocky Mountain National Park, ...
X	Connecticut	Hartford	Ned Lamont	⚑	Mystic Seaport, ...
X	Delaware	Tallahassee	John Camey	⚑	Caesar Rodney Statue, ...
X	Florida	Atlanta	Ron DeSantis	⚑	Kennedy Space Center, ...

g. Region Viewer Screen Context



Empire State Building - New York
Grand Canyon - Arizona
x Washington Monument
Yellowstone National Park - Wyoming

Region Name: United States
Parent Region: North America
Region Capital: Washington DC
Region Leader: Joseph Biden
Number of Subregions: 50

2. Your Routes
   a. "/" – root directory where the user is directed to the Welcome Screen and the user is **not** logged in.
   b. "/home" – home directory where the user is redirected to after a successful login.
   c. "/regions" – a page that consists a list of subregions where the user access a desired subregion
   d. "/subregion" - user redirected here after they select their desired region which displays a list of the subregions, capital, leader, flag, and landmarks.
   e. '/subregion/id" – has basic information that describes a summary of the selected subregion.
3. Schemas
   a. User:
      i. 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 })
   b. Region
      i. const regionSchema = new Schema({ _id: { type: ObjectId, required: true }, id: { type: Number, required: true }, name: { type: String, required: true }, owner: { type: String, required: true }, parentRegion: { type: String, required: **false** }, subregions: { type: [Region], required: **false** }, capital:

{.type: String, required: true }, leader: { type: String, required: true }, flag: { type: String, required: true }, landmarks: { type: [String], required: true } { timestamps: true })

4. Resolvers
   a. root-resolver

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

const regionsResolvers = require('./region-resolvers');

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

   b. user-resolvers

```
Query: {
        getCurrentUser: async (_, __, { req }) => {}
}
Mutation: {
        login: async (_ , args, { res }) => {}
        register: async (_, args, { res }) => {}
        update: async (_, args, { res }) => {}
        logout:(_, __, { res }) => {}
}
```

   c. region-resolvers

```
Query: {
        getAllRegions: async (_, { req }) => {}
        getRegionById: async (_, args) => {}

}

Mutation: {
        createMapFile: async (_, { args }) => {}
        deleteMapFile: async (_, { args }) => {}
        selectMapFile: async (_, { args }) => {}
        addSubregion: async (_, { args }) => {}
        editSubregion: async (_, { args }) => {}
        deleteSubregion: async (_, { args }) => {}
        sortNamesTable: async (_, { args }) => {}
        sortCapitalsTable: async (_, { args }) => {}
        sortLeadersTable: async (_, { args }) => {}
        sortFlagsTable: async (_, { args }) => {}
        sortLandmarksTable: async (_, { args }) => {}
```

```
                restoreOriginalList: (_, { args }) => {}
                changeRegionParent: (_, { args }) => {}
                addRegionLandmark: (_, { args }) => {}
                removeRegionLandmark: (_, { args }) => {}
                editRegionLandmark: (_, { args }) => {}
        }
```

5. Typedefs
    a. user-def

```
        const typeDefs = gql `
                type User {
                        _id: String
                        name: String
                        email: String
                        password: String
                }
                extend type Query {
                        getCurrentUser: User
                        testQuery: String
                }
                Extend type Mutation {
                        login(email: String!, password: String!): User
                        register(name: String!, email: String!, password: String!):
                        User
                        update(_id: String, name: String!, email: String!, password:
                        String!): User
                        logout: Boolean!
                }
        `;
        Module.exports = { typeDefs: typeDefs }
```

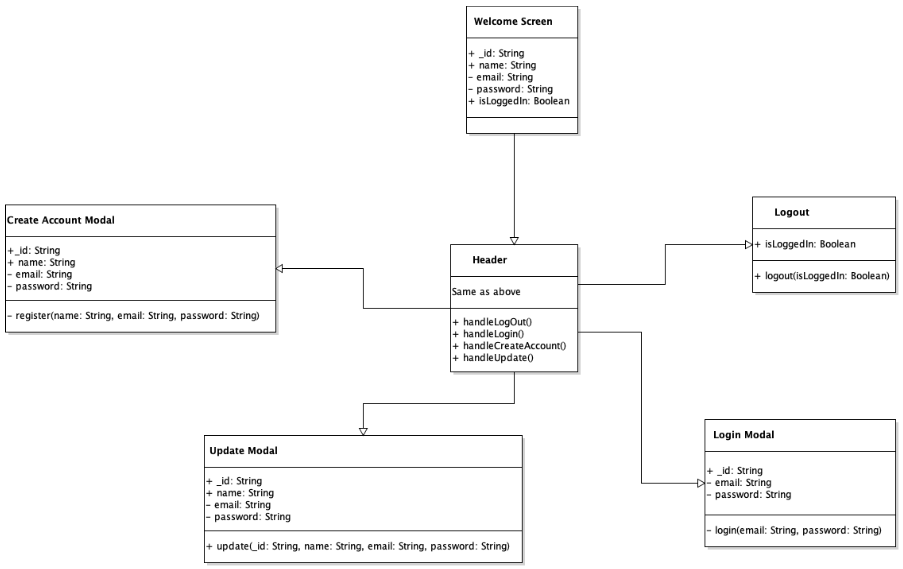
    b. region-defs

```
        const typeDefs = gql`
                type Region {
                        _id: String!
                        id: Int!
                        name: String!
                        owner: String!
                        parentRegion: [Region]
                        subregions: [Region]
```

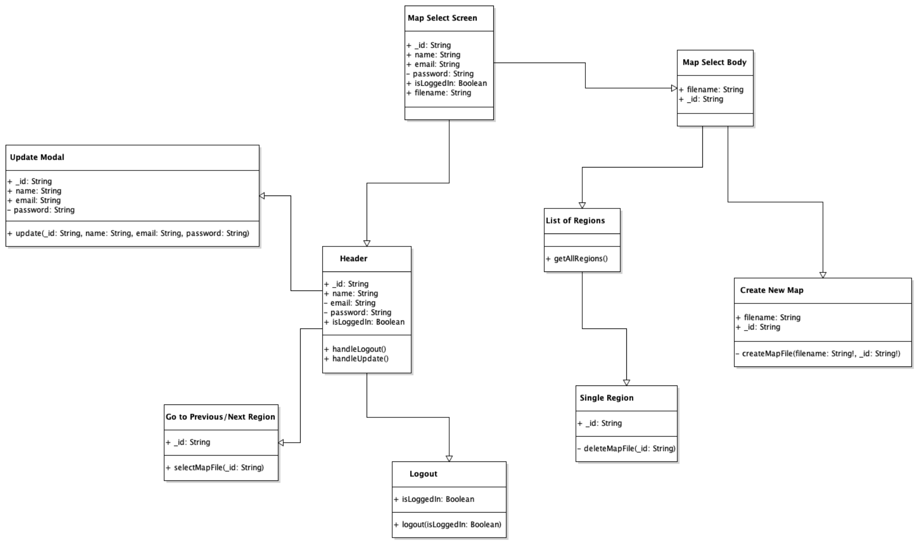
```
	capital: String!
	leader: String!
	flag: String!
	landmarks: [String]!
}
extend type Query {
	getAllRegions: [Region]
	getRegionById(_id: String!): Region
}
extend type Mutation {
	createMapFile(filename: String!, _id: String!): String
	deleteMapFile(_id: String!): Boolean
	selectMapFile(_id: String!): [Region]
	addSubregion(_id: String!, subregion: RegionInput!): String
	editSubregion(_id: String!, fieldToEdit: String!): [Region]
	deleteSubregion(_id: String!): [Region]
	sortNamesTable(_id: String!, sortNamesFlag: Int!): [Region]
	sortCapitalsTable(_id: String!, sortCapitalsFlag: Int!):
	[Region]
	sortLeadersTable(_id: String!, sortLeadersFlag: Int!):
	[Region]
	sortFlagsTable(_id: String!, sortFlagsFlag: Int!): [Region]
	sortLandmarksTable(_id: String!, sortLandmarksFlag:
	Int!):[Region]
	restoreOriginalList(_id: String!, orderItems: [Region]!):
	[Region]
	changeRegionParent(_id: String!, desiredParentId: String!):
	[Region]
	addRegionLandmark(_id: String!): String
	removeRegionLandmark(_id: String!, position: Int!): String
	editRegionLandmark(_id: String!, position: Int!): String
}
input RegionInput {
	_id: String
	id: Int!
	name: String!
	owner: String!
	parentRegion: [Region]
```

subregions: [Region]
capital: String!
leader: String!
flag: String!
landmarks: [String]!
}

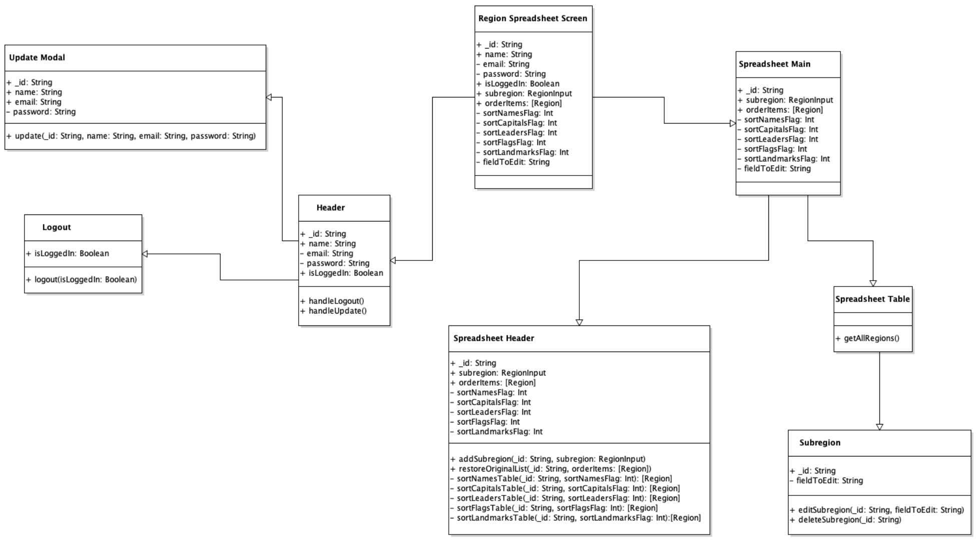
6. UML Diagrams
   a. Welcome Screen



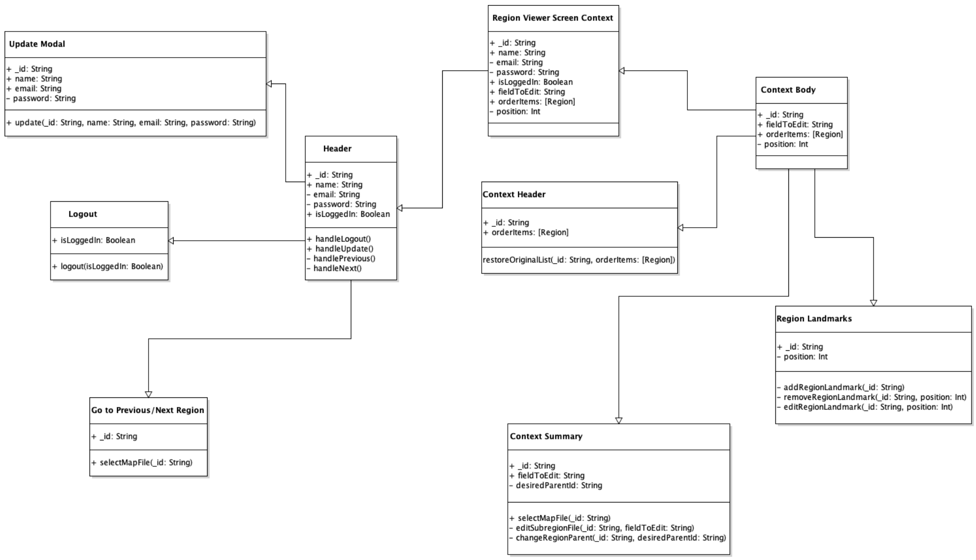
   b. Map Select Screen



c. Regions Spreadsheet Screen Context



d. Region Viewer Screen Context



Update Modal
+ _id: String
+ name: String
+ email: String
- password: String
+ update(_id: String, name: String, email: String, password: String)
Logout
+ isLoggedIn: Boolean
+ logout(isLoggedIn: Boolean)
Header
+ _id: String
+ name: String
- email: String
- password: String
+ isLoggedIn: Boolean
+ handleLogout()
+ handleUpdate()
- handlePrevious()
- handleNext()
Go to Previous/Next Region
+ _id: String
+ selectMapFile(_id: String)
Region Viewer Screen Context
+ _id: String
+ name: String
- email: String
- password: String
+ isLoggedIn: Boolean
+ fieldToEdit: String
+ orderItems: [Region]
- position: Int
Context Body
+ _id: String
+ fieldToEdit: String
+ orderItems: [Region]
- position: Int
Context Header
+ _id: String
+ orderItems: [Region]
restoreOriginalList(_id: String, orderItems: [Region])
Region Landmarks
+ _id: String
- position: Int
- addRegionLandmark(_id: String)
- removeRegionLandmark(_id: String, position: Int)
- editRegionLandmark(_id: String, position: Int)
Context Summary
+ _id: String
+ fieldToEdit: String
- desiredParentId: String
+ selectMapFile(_id: String)
- editSubregionFile(_id: String, fieldToEdit: String)
- changeRegionParent(_id: String, desiredParentId: String)