Sidharth Shamshabad

CSE 316: Final Project Mockups & UML Design

1. A picture containing logo

   Description automatically generatedUI Mockup Diagram
   1. Welcome Screen Context
   2. Graphical user interface, website

      Description automatically generatedCreate Account Screen Context
   3. Graphical user interface, website

      Description automatically generatedLogin Screen Context
   4. Update Screen Context

Graphical user interface, website

Description automatically generated

* 1. Graphical user interface, application

     Description automatically generatedMap Select Screen Context

Graphical user interface, application

Description automatically generated

* 1. Regions Spreadsheet Screen Context

Table

Description automatically generated with medium confidence

* 1. Region Viewer Screen Context

Graphical user interface, website

Description automatically generated

1. Your Routes
   1. “/” – root directory where the user is directed to the Welcome Screen and the user is **not** logged in.
   2. “/home” – home directory where the user is redirected to after a successful login.
   3. “/regions” – a page that consists a list of subregions where the user access a desired subregion
   4. “/subregion” - user redirected here after they select their desired region which displays a list of the subregions, capital, leader, flag, and landmarks.
   5. ‘/subregion/id” – has basic information that describes a summary of the selected subregion.
2. Schemas
   1. User:
      1. 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 })
   2. Region
      1. 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 })
3. Resolvers
   1. root-resolver

const userResolvers = require(‘./user-resolvers’);

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

module.exports = [userResolvers, regionsResolvers];

* 1. user-resolvers

Query: {

getCurrentUser: async (\_, \_\_, { req }) => {}

}

Mutation: {

login: async (\_ , args, { res }) => {}

register: async (\_, args, { res }) => {}

update: async (\_, args, { res }) => {}

logout:(\_, \_\_, { res }) => {}

}

* 1. 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 }) => {}

}

1. Typedefs
   1. 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 }

* 1. 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]!

}

1. UML Diagrams
   1. Welcome Screen

**Diagram

Description automatically generated**

* 1. Map Select Screen

Diagram

Description automatically generated

* 1. Regions Spreadsheet Screen Context

**Diagram, schematic

Description automatically generated**

* 1. Region Viewer Screen Context

Diagram

Description automatically generated