**Futsal Field Reservation SYstem**

**Backend:**

**Constant:**

**#constant.js**

export const CONNECTION\_URL = "mongodb+srv://hammadahsan07:qhNkH6SGY7F65mWU@futsal-cluster.y8gj5r6.mongodb.net/?retryWrites=true&w=majority&appName=Futsal-Cluster";

export const PORT = process.env.PORT || 5000;

**Models:**

**#academy.js:**

import mongoose from 'mongoose';

const academySchema =  mongoose.Schema({

    studentname: String,

    fathername: String,

    email: String,

    age: Number,

    contact: String,

    address: String,

    myacademy: String

})

export default new mongoose.model("Academy", academySchema)

**#ground.js:**

import mongoose from "mongoose";

import bcrypt from "bcryptjs";

const groundSchema = mongoose.Schema({

  name: String,

  ownername: String,

  email: String,

  password: String,

  phone: String,

  address: String,

  imgURL: String,

  slot: [

    {

      date: String,

      time: [

        {

          bookedby: { type: String, default: "" },

          date: { type: String, default: "" },

          in: { type: String, default: "" },

        },

      ],

    },

  ],

});

groundSchema.pre("save", async function (next) {

  if (this.isModified("password")) {

    this.password = await bcrypt.hash(this.password, 12);

  }

  next();

});

export default new mongoose.model("Ground", groundSchema);

**#team.js:**

import mongoose from "mongoose";

import bcrypt from "bcryptjs";

const teamSchema = mongoose.Schema({

  name: String,

  captainName: String,

  email: String,

  password: String,

  phone: String,

  address: String,

  // aboutTeam: {

  //     type: String,

  //     default: []

  // },

  statusArray: {

    type: Array,

    default: [],

  },

  imgURL: String,

});

teamSchema.pre("save", async function (next) {

  if (this.isModified("password")) {

    this.password = await bcrypt.hash(this.password, 12);

  }

  next();

});

export default new mongoose.model("Team", teamSchema);

**Routes:**

**#academy.js:**

import express from "express";

const router = express.Router();

//Model

import Academy from "../models/academy.js";

router.get("/users/:name", async (req, res) => {

  const name = req.params.name;

  const academy = await Academy.find({ myacademy: name });

  if (!academy) {

    res.send({});

  } else {

    res.send(academy);

  }

});

// /academy/test

router.get("/test3", async (req, res, next) => {

  const academies = await Academy.find();

  res.send(academies);

});

router.post("/academyformregister", async (req, res) => {

  //console.log(req.body)

  const { studentname, fathername, email, age, contact, address, myacademy } =

    req.body;

  Academy.findOne({ email: email }, (err, mongoacademy) => {

    if (mongoacademy) {

      res.send({

        message:

          "Student already Registered with this Email. Try something new. ",

      });

    }

    //Creating. Enter data in database

    else {

      const mongoacademy = new Academy({

        studentname,

        fathername,

        email,

        age,

        contact,

        address,

        myacademy,

      });

      mongoacademy.save((err) => {

        if (err) {

          res.send(err);

        } else {

          res.send({ message: "Student Successfull Registered to Academy" });

        }

      });

    }

  });

});

export default router;

**#Grounds.js:**

import express from "express";

import bcrypt from "bcryptjs";

const router = express.Router();

//Model

import Ground from "../models/ground.js";

// /ground/test

router.get("/test", async (req, res, next) => {

  const grounds = await Ground.find();

  res.send(grounds);

});

//GET PROFILE DETAILS

router.get("/profile/:\_id", async (req, res) => {

  const \_id = req.params.\_id;

  const ground = await Ground.findOne({ \_id });

  if (!ground) {

    res.send({ message: "Not found" });

  } else {

    res.send(ground);

  }

});

router.post("/dp", async (req, res) => {

  try {

    const { \_id, imgURL } = req.body;

    const ground = await Ground.findOneAndUpdate(

      { \_id },

      {

        $set: {

          imgURL,

        },

      }

    );

    res.send({ message: "Updated" });

  } catch (error) {}

});

router.post("/login2", async (req, res, next) => {

  try {

    const { email, password } = req.body;

    if (!email || !password) {

      return res.send({ message: "Fill all the Fields" });

    }

    const ground = await Ground.findOne({ email: email });

    if (ground) {

      const isMatch = await bcrypt.compare(password, ground.password);

      if (!isMatch) {

        res.send({ message: "Password didn't match" });

      } else {

        res.send({ message: "Login Successsfully", ground });

      }

    } else {

      return res.send({

        message: "Ground not Registered, Please make an account first",

      });

    }

  } catch (err) {

    console.log(err);

    res.send(err);

  }

});

router.post("/register", async (req, res, next) => {

  try {

    const { name, ownername, email, password, phone, address } = req.body;

    const ground = await Ground.findOne({ email });

    if (ground) {

      res.send({ message: "Ground already Registered" });

    }

    //Creating. Enter data in database

    //teamname:teamname;

    else {

      const newGround = new Ground({

        name,

        ownername,

        email,

        password,

        phone,

        address,

      });

      await newGround.save();

      res.send({ message: "Successfully Registered, Please Login Now" });

    }

  } catch (err) {

    console.log(err);

    res.send(err);

  }

});

router.post("/getSlot", async (req, res) => {

  try {

    const { \_id, date } = req.body;

    const ground = await Ground.findOne({ \_id });

    console.log('ground',ground)

    if (!ground) {

      return res.status(404).json({ error: "Ground not found" }); // Use return

    }

    const slots = ground.slot;

    const matchedSlot = slots.find(item => item.date === date);

    if (matchedSlot) {

      return res.json(matchedSlot.time); // Return time if date matches

    } else {

      return res.json([]); // Return empty array if no match

    }

  } catch (error) {

    console.error('error',error);

    return res.status(500).json({ error: "Server error" });

  }

});

router.post("/bookSlot", async (req, res) => {

  try {

    const { \_id, date, name, index } = req.body;

    console.log(\_id, date, name, index);

    const ground = await Ground.findOne({ \_id });

    const slot = ground.slot;

    let times = [];

    let flag = false;

    const data = {

      bookedby: name,

      date: date,

      in: index,

    };

    slot.map((item, index) => {

      if (item.date == date) {

        times = item.time;

        flag = true;

      }

    });

    if (flag == true) {

      // if (times[index] != null || times[index] != undefined) {

      //   return res.send({ message: "Already Booked this slot" });

      // }

      times[index] = data;

      await Ground.findOneAndUpdate(

        { \_id, "slot.date": date },

        {

          $push: {

            "slot.$[date].time": {

              $each: [data],

              $position: index,

            },

          },

        },

        {

          arrayFilters: [{ "date.date": date }],

        }

      );

      res.send({ message: "Successfully Booked " });

    } else {

      times.push(data);

      let dat = { date, time: times };

      await Ground.findOneAndUpdate(

        { \_id },

        {

          $push: {

            slot: dat,

          },

        }

      );

      res.send({ message: "Successfully Booked " });

    }

  } catch (error) {

    res.send({ error: error });

  }

});

router.post("/undoSlot", async (req, res) => {

  try {

    const { index, \_id, date } = req.body;

    const ground = await Ground.findOne({ \_id });

    const slot = ground.slot;

    await Ground.findOneAndUpdate(

      { \_id },

      {

        $pull: {

          "slot.$[date].time": { in: index },

        },

      },

      {

        arrayFilters: [{ "date.date": date }],

      }

    );

    res.send({ message: "Sucessfully Removed " });

  } catch (error) {

    res.send({ error: error });

  }

});

export default router;

**#Teams.js:**

import express from "express";

import bcrypt from "bcryptjs";

const router = express.Router();

//Model

import Team from "../models/team.js";

import Ground from "../models/ground.js";

//ALL TEAMS AND GROUND

router.get("/all/:\_id", async (req, res) => {

  const \_id = req.params.\_id;

  let k = "new ObjectId(";

  const team = await Team.find({});

  const ground = await Ground.find({});

  const all = team.concat(ground);

  let complete = [];

  all.map((item, index) => {

    if (item.\_id.toString() != \_id) {

      complete.push(item);

    }

  });

  // console.log(complete);

  // console.log(\_id, `new ObjectId(\"${\_id}\")`);

  res.send(complete);

});

//GET PROFILE DETAILS

router.get("/profile/:\_id", async (req, res) => {

  const \_id = req.params.\_id;

  const team = await Team.findOne({ \_id });

  if (!team) {

    res.send({ message: "Not found" });

  } else {

    res.send(team);

  }

});

// /team/test

router.get("/test", async (req, res, next) => {

  const teams = await Team.find({});

  res.send(teams);

});

router.post("/dp", async (req, res) => {

  try {

    const { \_id, imgURL } = req.body;

    const team = await Team.findOneAndUpdate(

      { \_id },

      {

        $set: {

          imgURL,

        },

      }

    );

    res.send({ message: "Updated" });

  } catch (error) {}

});

router.post("/login", async (req, res, next) => {

  try {

    const { email, password } = req.body;

    if (!email || !password) {

      return res.send({ message: "Fill all the Fields" });

    }

    const team = await Team.findOne({ email: email });

    console.log("team", team);

    if (team) {

      const isMatch = await bcrypt.compare(password, team.password);

      if (!isMatch) {

        res.send({ message: "Password didn't match" });

      } else {

        res.send({ message: "Login Successsfully", team });

      }

    } else {

      return res.send({

        message: "Team not Registered, Please make an account first",

      });

    }

  } catch (err) {

    console.log('error',err);

    res.send(err);

  }

});

router.post("/register", async (req, res, next) => {

  try {

    const { name, captainName, email, password, phone, address } = req.body;

    const team = await Team.findOne({ email });

    if (team) {

      res.send({ message: "Team already Registered" });

    }

    //Creating. Enter data in database

    //teamname:teamname;

    else {

      const newTeam = new Team({

        name,

        captainName,

        email,

        password,

        phone,

        address,

      });

      await newTeam.save();

      res.send({ message: "Successfully Registered, Please Login Now" });

    }

  } catch (err) {

    console.log(err);

    res.send(err);

  }

});

router.post("/setStatus/:teamId", async (req, res, next) => {

  console.log("ROUTE ACTIVATED");

  const { status } = req.body;

  try {

    const { teamId } = req.params;

    const team = await Team.findById(teamId);

    const statusArray = team.statusArray;

    console.log(statusArray);

    statusArray.push(status);

    team.statusArray = statusArray;

    await team.save();

    res.send({ message: "Success", statusArray });

  } catch (err) {

    res.send(err);

  }

});

router.get("/getStatus/:teamId", async (req, res, next) => {

  try {

    const { teamId } = req.params;

    const team = await Team.findById(teamId);

    const statusArray = team.statusArray;

    res.send({ message: "Success", statusArray });

    console.log(statusArray);

  } catch (err) {

    res.send(err);

  }

});

// router.post("/setAbout/:teamId", async (req, res, next) => {

//     console.log("ROUTE ACTIVATED")

//     const {about} = req.body;

//     try{

//         const {teamId} = req.params;

//         const team = await Team.findById(teamId);

//         const aboutTeam = team.aboutTeam;

//         console.log(about)

//         aboutTeam.push(about);

//         team.aboutTeam = aboutTeam;

//         await team.save();

//         res.send({message: "Success", aboutTeam})

//     }

//     catch(err){

//         res.send(err);

//     }

// });

// router.get("/getAbout/:teamId", async (req, res, next) => {

//     try{

//         const {teamId} = req.params;

//         const team = await Team.findById(teamId);

//         const aboutTeam = team.aboutTeam;

//         res.send({message: "Success", aboutTeam})

//     }

//     catch(err){

//         res.send(err);

//     }

// })

export default router;

**Outside the routes folder**

**#academybackend.js**

import express from "express";

import cors from "cors";

import mongoose from "mongoose";

import \* as CONSTANT from "./Constants/constants.js";

import bcrypt from "bcryptjs";

// Configuration

const app = express();

app.use(express.json());

app.use(express.urlencoded());

app.use(cors());

mongoose.connect(

  CONSTANT.CONNECTION\_URL,

  {

    useNewUrlParser: true,

    useUnifiedTopology: true,

  },

  () => {

    console.log("Database Connected");

  }

);

const academySchema = new mongoose.Schema({

  studentname: String,

  fathername: String,

  email: String,

  age: Number,

  contact: Number,

  address: String,

  myacademy: String,

});

const Academy = new mongoose.model("Academy", academySchema);

app.post("/academyformregister", async (req, res) => {

  //console.log(req.body)

  const { studentname, fathername, email, age, contact, address, myacademy } =

    req.body;

  Academy.findOne({ email: email }, (err, mongoacademy) => {

    if (mongoacademy) {

      res.send({

        message:

          "Student already Registered with this Email. Try something new. ",

      });

    }

    //Creating. Enter data in database

    else {

      const mongoacademy = new Academy({

        studentname,

        fathername,

        email,

        age,

        contact,

        address,

        myacademy,

      });

      mongoacademy.save((err) => {

        if (err) {

          res.send(err);

        } else {

          res.send({ message: "Student Successfull Registered to Academy" });

        }

      });

    }

  });

});

MongoClient.connect(url, (err, db) => {

  if (err) throw error;

  const dbo = db.db("bookmyslot");

  dbo

    .collection("teams")

    .aggregate([

      {

        $lookup: {

          from: "academies",

          localField: "myacademy",

          foreignField: "\_id",

          as: "new\_data",

        },

      },

    ])

    .toArray((err, res) => {

      if (err) throw err;

      console.log(JSON.stringify(res));

      db.close();

    });

});

app.listen(CONSTANT.PORT, () => {

  console.log("Academyformbackend started at port 9006");

});

**#app.js:**

import express from "express";

import mongoose from "mongoose";

import cors from "cors";

//Routes

import teamRoute from "./routes/team.js";

import groundRoute from "./routes/ground.js";

import academyRoute from "./routes/academy.js";

import \* as CONSTANT from "./Constants/constants.js";

const app = express();

// CORS Configuration (Yeh pehle hi add karein)

app.use(cors({

  origin: ['http://localhost:3000', 'http://localhost:3001'], // Specific allowed origins

  methods: ['GET', 'POST', 'PUT', 'DELETE'], // Allowed methods

  credentials: true // If using cookies/auth

}));

// Body Parsers

app.use(express.json());

app.use(express.urlencoded({ extended: true })); // <-- extended option add karein

// Home Route (Add this before other routes)

app.get("/", (req, res) => {

  res.send("Futsal API Working! Try /team, /ground or /academy");

});

// Routes

app.use("/team", teamRoute);

app.use("/ground", groundRoute);

app.use("/academy", academyRoute);

// 404 Handler

app.use("\*", (req, res) => {

  res.status(404).send({ message: "Wrong Endpoint" });

});

// DB Connection

mongoose.connect(CONSTANT.CONNECTION\_URL, {

  useNewUrlParser: true,

  useUnifiedTopology: true

})

.then(() => {

  console.log("Database Connected");

  app.listen(CONSTANT.PORT, () =>

    console.log(`Server running on port ${CONSTANT.PORT}`)

  );

})

.catch(err => console.log("Connection Failed:", err.message));

**Frontend**

**SRC:**

**#user.js:**

import { useState, useEffect } from 'react';

import Button from '@material-ui/core/Button';

import Box from '@material-ui/core/Box';

const User = () => {

  const [selectedImage, setSelectedImage] = useState(null);

  const [imageUrl, setImageUrl] = useState(null);

  useEffect(() => {

    if (selectedImage) {

      setImageUrl(URL.createObjectURL(selectedImage));

    }

  }, [selectedImage]);

  return (

    <>

      <input

        accept="image/\*"

        type="file"

        id="select-image"

        style={{ display: 'none' }}

        onChange={e => setSelectedImage(e.target.files[0])}

      />

      <label htmlFor="select-image">

        <Button variant="contained" color="primary" component="span">

          Upload photo

        </Button>

      </label>

      {imageUrl && selectedImage && (

        <Box mt={2} textAlign="center">

          <img src={imageUrl} alt={selectedImage.name} height="200px" />

        </Box>

      )}

    </>

  );

};

export default User;

**#profile.jsx:**

import React from "react";

import AddIcon from "@mui/icons-material/Add";

import { useState, useEffect } from "react";

import Button from "@material-ui/core/Button";

import { Grid } from "@material-ui/core";

import axios from "axios";

import \* as CONTANT from "../Constant/constant.js";

import { useHistory } from "react-router-dom";

import { fontSize } from "@mui/system";

import { useSelector } from "react-redux";

import { ToastContainer, toast } from "react-toastify";

import "react-toastify/dist/ReactToastify.css";

import { storage } from ".//firebase";

import {

  ref,

  uploadBytes,

  getDownloadURL,

  listAll,

  list,

} from "firebase/storage";

import { v4 } from "uuid";

import { useDispatch } from "react-redux";

import {

  setTeamId,

  setAddress,

  setCaptainName,

  setEmail,

  setPhone,

  setTeamName,

} from "../redux/slices/teamSlice";

import { getStorage, deleteObject } from "firebase/storage";

// import firebase from "firebase/compat/app";

// import "firebase/compat/auth";

// import "firebase/compat/firestore";

function Profile() {

  const storagee = getStorage();

  const dispatch = useDispatch();

  const { localStorage } = window;

  const userEmail = useSelector((state) => state?.utilitySlice?.userEmail);

    const storedTeamId = localStorage.getItem("teamId");

    const reduxTeamId = useSelector((state) => state?.teamSlice?.teamId);

    const teamId = storedTeamId || reduxTeamId;

    console.log("teamId", teamId);

    const storedTeamName = localStorage.getItem("teamname");

    const reduxTeamName = useSelector((state) => state?.teamSlice?.teamname);

    const team = storedTeamName || reduxTeamName;

    console.log("team", team);

  //Dp

  const [dp\_Url, setDpUrl] = useState(localStorage.getItem("dpUrl"));

  const [dpUpload, setDpUpload] = useState(null);

  const [url3, setUrl3] = useState(null);

  useEffect(() => {

    setTimeout(() => {

      setDpUrl(localStorage.getItem("dpUrl"));

      getAllStatus();

    }, 1000);

  }, [dp\_Url]);

  const uploadDp = () => {

    if (dpUpload == null) return;

    // const dpRef = ref(storage, `teamdp/${dpUpload.name + v4()}`)

    const dpRef = ref(storage, `teamdp/${teamId}`);

    // const dpRef = ref(storage, `teamdp/${}`)

    uploadBytes(dpRef, dpUpload)

      .then(() => {

        getDownloadURL(dpRef)

          .then((url3) => {

            setDpUrl(url3);

            window.localStorage.setItem("dpUrl", url3);

            CONTANT.API.post("/team/dp", { \_id: teamId, imgURL: url3 });

          })

          .catch((error) => {

            console.log(error.message, "error getting while uploading");

          });

        setDpUpload(null);

      })

      .catch((error) => {

        console.log(error.message);

      });

  };

  //Cover

  const [cover\_Url, setCoverUrl] = useState(localStorage.getItem("coverUrl"));

  const [coverUpload, setCoverUpload] = useState(null);

  const [url2, setUrl2] = useState(null);

  useEffect(() => {

    getAllStatus();

  });

  useEffect(() => {

    setTimeout(() => {

      setCoverUrl(localStorage.getItem("coverUrl"));

      getAllStatus();

    }, 1000);

  }, [cover\_Url]);

  const uploadCover = () => {

    if (coverUpload == null) return;

    const coverRef = ref(storage, `teamcover/${teamId}`);

    uploadBytes(coverRef, coverUpload)

      .then(() => {

        getDownloadURL(coverRef)

          .then((url2) => {

            setCoverUrl(url2);

            window.localStorage.setItem("coverUrl", url2);

          })

          .catch((error) => {

            console.log(error.message, "error getting while uploading");

          });

        setCoverUpload(null);

      })

      .catch((error) => {

        console.log(error.message);

      });

  };

  //Delete Cover

  const Deletecover = () => {

    const storage = getStorage();

    // Create a reference to the file to delete

    const desertRef = ref(storage, `teamcover/${teamId}`);

    // Delete the file

    deleteObject(desertRef)

      .then(() => {

        window.localStorage.removeItem("coverUrl");

        setCoverUrl();

        // File deleted successfully

      })

      .catch((error) => {

        // Uh-oh, an error occurred!

      });

  };

  //Delete Dp

  const Deletedp = () => {

    const storage = getStorage();

    // Create a reference to the file to delete

    const desertRef = ref(storage, `teamdp/${teamId}`);

    // Delete the file

    deleteObject(desertRef)

      .then(() => {

        // File deleted successfully

        window.localStorage.removeItem("dpUrl");

        setDpUrl();

      })

      .catch((error) => {

        // Uh-oh, an error occurred!

      });

    // window.location.reload(false);

  };

  //Status

  const [status, setStatus] = useState("");

  const [statusArray, setStatusArray] = useState([]);

  const getAllStatus = async () => {

    const response = await CONTANT.API.get(`/team/getStatus/${teamId}`);

    if (response?.data) {

      setStatusArray(response.data.statusArray.reverse());

    }

  };

  //Timeline

  const [imageUpload, setImageUpload] = useState(null);

  const [imageList, setImageList] = useState([]);

  const [imagePath, setImagePath] = useState([]);

  const imageListRef = ref(storage, `teamtimeline/${teamId}`);

  const uploadImage = () => {

    if (imageUpload == null) return;

    const imageRef = ref(

      storage,

      `teamtimeline/${teamId}/${imageUpload.name + v4()}`

    );

    uploadBytes(imageRef, imageUpload).then((snapshot) => {

      getDownloadURL(snapshot.ref).then((url) => {

        const list = [...imageList];

        list.unshift(url);

        setImageList(list);

        window.localStorage.setItem("imageList", list);

      });

    });

  };

  const deletePicture = async (e, index) => {

    e.preventDefault();

    const list = [...imageList];

    const paths = [...imagePath];

    const deletePath = paths[index];

    paths.splice(index, 1);

    setImagePath(paths);

    list.splice(index, 1);

    setImageList(list);

    const storage = getStorage();

    // Create a reference to the file to delete

    const desertRef = ref(storage, deletePath);

    deleteObject(desertRef)

      .then(() => {

        // File deleted successfully

        console.log("Deleted");

      })

      .catch((error) => {

        // Uh-oh, an error occurred!

      });

    window.localStorage.setItem("imageList", list);

  };

  const deleteStatus = (e, index) => {

    e.preventDefault();

    const list = [...statusArray];

    list.splice(index, 1);

    setStatusArray(list);

    const url = list[index];

    console.log(url);

    // let imageRef = storagee.refFromURL(url);

    // imageRef.delete();

    window.localStorage.setItem("statusArray", list);

  };

  useEffect(() => {

    listAll(imageListRef).then((response) => {

      response.items.forEach((item) => {

        let path = imagePath;

        path.push(item.fullPath);

        setImagePath([...path]);

        getDownloadURL(item).then((url) => {

          let AllURL = imageList;

          AllURL.push(url);

          setImageList([...AllURL]);

        });

      });

    });

  }, []);

  const logout = () => {

    const { localStorage } = window;

    localStorage.clear();

    dispatch(setTeamId(""));

    dispatch(setTeamName(""));

    dispatch(setCaptainName(""));

    dispatch(setEmail(""));

    dispatch(setPhone(""));

    dispatch(setAddress(""));

  };

  const addStatus = async () => {

    const list = [...statusArray];

    list.unshift(status);

    setStatusArray(list);

    window.localStorage.setItem("statusArray", list);

    const response = await CONTANT.API.post(`/team/setStatus/${teamId}`, {

      status,

    });

    setStatus("");

    console.log(response);

  };

  return (

    <>

      <div className="profilepage">

        <ToastContainer />

        <div className="profilecontainer">

          <button className="logoutt" onClick={() => logout()}>

            Logout

          </button>

          {/\* Cover \*/}

          <input

            type="file"

            className="coverinputt"

            id="select-image2"

            onChange={(event) => {

              setCoverUpload(event.target.files[0]);

            }}

          />

          <label htmlFor="select-image2">

            <Button

              onClick={uploadCover}

              variant="contained"

              color="primary"

              id="coverimginput"

            >

              Upload Cover

            </Button>

          </label>

          <div className="coverborder">

            {cover\_Url && <img src={cover\_Url} className="coverimg" alt="" />}

          </div>

          {/\* Dp \*/}

          <div className="profiledetails">

            <div className="profileimg">

              <input

                type="file"

                className="dpinputtt"

                id="select-image"

                onChange={(event) => {

                  setDpUpload(event.target.files[0]);

                }}

              />

              <label htmlFor="select-image">

                <Button

                  onClick={uploadDp}

                  variant="contained"

                  color="primary"

                  id="profilephotoinputt"

                >

                  Upload Dp

                </Button>

              </label>

              <div id="dpborder">

                {dp\_Url && <img src={dp\_Url} className="profilephoto" alt="" />}

              </div>

            </div>

            <button

              className="message"

              onClick={() => {

                window.open(

                  "http://localhost:3001",

                  "\_blank"

                );

              }}

            >

              Message

            </button>

            <button className="matchup2">Match Up</button>

          </div>

          <Button onClick={Deletecover}> Delete Cover </Button>

          <Button onClick={Deletedp}> Delete Dp </Button>

          <br></br>

          <br></br>

          <div className="details">

            <h1 className="Name">{team}</h1>

          </div>

          {/\* About \*/}

          <form method="POST">

            {/\* Timeline \*/}

            <div className="teamgallery">

              <h2>Timeline</h2>

              <input

                type="file"

                onChange={(event) => {

                  setImageUpload(event.target.files[0]);

                }}

              />

              <br></br>

              <br></br>

              <div></div>

              <Button onClick={uploadImage} variant="contained" color="primary">

                Upload Image

              </Button>

              <br></br>

              <br></br>

              {imageList &&

                imageList?.map((url, index) => {

                  return (

                    <Grid key={index} container>

                      <Grid item xs={12}>

                        <img src={url} className="timelinepics" alt="" />

                      </Grid>

                      <Grid item xs={12}>

                        <Button

                          style={{}}

                          onClick={(e) => deletePicture(e, index)}

                          variant="contained"

                          color="secondary"

                        >

                          Delete Picture

                        </Button>

                      </Grid>

                    </Grid>

                  );

                })}

            </div>

            {/\*Status\*/}

            <div className="teamgallery">

              <h2>Status</h2>

              <div>

                <textarea

                  type="text"

                  name="status"

                  autoComplete="off"

                  placeholder="What's on Your Mind ......"

                  onChange={(event) => setStatus(event.target.value)}

                  value={status}

                  id="status"

                ></textarea>

              </div>

              <button id="statusbutton" onClick={addStatus} type="button">

                <p

                  style={{

                    marginLeft: "-11px",

                    marginTop: "-3px",

                    fontSize: "14px",

                  }}

                >

                  {" "}

                  Share{" "}

                </p>

              </button>

            </div>

          </form>

          <br></br>

          <br></br>

          {statusArray &&

            statusArray?.map((status, index) => {

              return (

                <div key={index} className="statusresult">

                  <div style={{ marginTop: "2vh", float: "right" }}>

                    <Button

                      style={{}}

                      onClick={(e) => deleteStatus(e, index)}

                      variant="contained"

                      color="secondary"

                    >

                      Delete Status

                    </Button>

                  </div>

                  <div className="statusposted">

                    <h1>{status}</h1>

                  </div>

                </div>

              );

            })}

        </div>

      </div>

    </>

  );

}

export default Profile;

**#index.js:**

import React from "react";

import ReactDOM from "react-dom";

import App from "./App.jsx";

import { BrowserRouter } from "react-router-dom";

import { Provider } from "react-redux";

import store from "./redux/store";

import "bootstrap/dist/css/bootstrap.min.css";

import 'bootstrap/dist/js/bootstrap.bundle.min.js';

ReactDOM.render(

  <Provider store={store}>

    <App></App>

  </Provider>,

  document.getElementById("root")

);

**#app.jsx:**

import React, { useState } from "react";

import "../node\_modules/bootstrap/dist/css/bootstrap.min.css";

import "./index.css";

import { BrowserRouter as Router, Route, Switch } from "react-router-dom";

import Routes from "./routes/Routes";

import { useDispatch } from "react-redux";

import {

  setTeamId,

  setAddress,

  setCaptainName,

  setEmail,

  setPhone,

  setTeamName,

} from "./redux/slices/teamSlice";

import { useEffect } from "react";

import { css } from "@emotion/react";

import ClipLoader from "react-spinners/ClipLoader";

function App() {

  const dispatch = useDispatch();

  const { sessionStorage } = window;

  let [loading, setLoading] = useState(true);

  let [color, setColor] = useState("#ffffff");

  const override = css`

    display: block;

    margin: 0 auto;

    border-color: red;

  `;

  useEffect(() => {

    const teamId = sessionStorage.getItem("teamId");

    if (teamId) {

      dispatch(setTeamId(sessionStorage.getItem("teamId")));

      dispatch(setTeamName(sessionStorage.getItem("teamname")));

      dispatch(setCaptainName(sessionStorage.getItem("captainname")));

      dispatch(setEmail(sessionStorage.getItem("email")));

      dispatch(setPhone(sessionStorage.getItem("phone")));

      dispatch(setAddress(sessionStorage.getItem("address")));

    } else {

      sessionStorage.clear();

      dispatch(setTeamId(""));

      dispatch(setTeamName(""));

      dispatch(setCaptainName(""));

      dispatch(setEmail(""));

      dispatch(setPhone(""));

      dispatch(setAddress(""));

    }

  }, [window.location]);

  return (

    <div>

      <Routes />

    </div>

  );

}

export default App;

**Routes:**

**#routes.js:**

import React, { useEffect, useState } from "react";

import { Route, Switch, BrowserRouter as Router } from "react-router-dom";

import Menu from "../pages/Menu";

import Home from "../pages/Home";

import Groundsfrontpage from "../pages/Groundsfrontpage";

import Teamfrontpages from "../pages/Teamfrontpages";

import Academydisplaypage from "../pages/Academydisplaypage";

import Login from "../pages/Login";

import Login2 from "../pages/Login2";

import Mylogin from "../pages/Mylogin";

import Footer from "../pages/Footer";

import Teamform from "../pages/Teamform";

import Groundform from "../pages/Groundform";

import Groundprofile from "../pages/Groundprofile";

import Academyform from "../pages/Academyform";

import Academyfrontpage from "../pages/Academyfrontpage";

import Donation from "../pages/Donation";

import Donationform from "../pages/Donationform";

import Profile from "../pages/Profile";

import Booking from "../pages/Booking";

import Booking2 from "../pages/Booking2";

import VisitorProfile from "../pages/VisitorProfile.jsx";

import VisitorGroundProfile from "../pages/VisitorGroundProfile";

import AcademyUsers from "../pages/AcademyUsers";

import { useSelector } from "react-redux";

import { css } from "@emotion/react";

import ClipLoader from "react-spinners/ClipLoader";

function Routes() {

  const storedTeamId = localStorage.getItem("teamId");

  const reduxTeamId = useSelector((state) => state?.teamSlice?.teamId);

  const teamId = storedTeamId || reduxTeamId;

  console.log("teamId", teamId);

  const storedGroundId = localStorage.getItem("groundId");

  const reduxGroundId = useSelector((state) => state?.groundSlice?.groundId);

  const groundId = storedGroundId || reduxGroundId;

  console.log("groundId", groundId);

  const [teamlogin, setTeamLogin] = useState({});

  const [groundlogin, setGroundLogin] = useState({});

  let [loading, setLoading] = useState(true);

  let [color, setColor] = useState("#ffffff");

  const override = css`

    display: block;

    margin: 0 auto;

    border-color: red;

  `;

  return (

    <Router>

      <Menu />

      <Switch>

        <Route exact path="/">

          <Home />

        </Route>

        <Route path="/donation">

          <Donation />

        </Route>

        <Route path="/donationform">

          <Donationform />

        </Route>

        <Route path="/academyuser">

          <AcademyUsers />

        </Route>

        <Route path="/visitorprofile">

          <VisitorProfile />

        </Route>

        <Route path="/visitorgroundprofile">

          <VisitorGroundProfile />

        </Route>

        <Route path="/academies">

          <Academyfrontpage />

        </Route>

        <Route path="/academydisplaypage">

          <Academydisplaypage />

        </Route>

        <Route path="/grounds">

          <Groundsfrontpage />

        </Route>

        <Route eaxct path="/teams">

          <Teamfrontpages />

        </Route>

        <Route path="/booking">

          {teamId || groundId ? <Booking /> : <Login setTeamLogin={setTeamLogin} />}

        </Route>

        {/\* <Route eaxct path="/booking">

          <Booking />

        </Route> \*/}

        <Route path="/booking2">

          <Booking2 />

        </Route>

        <Route path="/teamform">

          <Teamform />

        </Route>

        <Route path="/groundform">

          <Groundform />

        </Route>

        <Route path="/academyform">

          <Academyform />

        </Route>

        <Route path="/mylogin">

          <Mylogin />

        </Route>

        <Route path="/groundprofile">

          {groundId ? (

            <Groundprofile setGroundLogin={setGroundLogin} />

          ) : (

            <Login2 setGroundLogin={setGroundLogin} />

          )}

        </Route>

        <Route path="/login2">

          <Login2 setGroundLogin={setGroundLogin} />

        </Route>

        <Route path="/profile">

          {teamId ? (

            <Profile setTeamLogin={setTeamLogin} />

          ) : (

            <Login setTeamLogin={setTeamLogin} />

          )}

        </Route>

        <Route path="/login">

          <Login setTeamLogin={setTeamLogin} />

        </Route>

      </Switch>

    </Router>

  );

}

export default Routes;

**Redux:**

**#groundReducer.js:**

import {combineReducer} from "redux";

import groundSlice from "./slices/groundSlice";

export default groundReducers({

    groundSlice

})

**#rootReducer.js:**

import { combineReducers } from "redux";

import utilitySlice from './slices/utilitySlice'

import teamSlice from "./slices/teamSlice"

import groundSlice from "./slices/groundSlice";

export default combineReducers({

    utilitySlice,

    teamSlice,

    groundSlice

});

**#store.js:**

import { configureStore } from "@reduxjs/toolkit";

import rootReducer from "./rootReducer";

export default configureStore({

    reducer: rootReducer,

});

**#teamReducer.js:**

import {combineReducer} from "redux";

import teamSlice from "./slices/teamSlice";

export default teamReducers({

    teamSlice

})

**Actions:  
#authAction.js:**

import { login } from "../../service/Auth";

import { LOGIN\_SUCCESS, LOGIN\_ERROR } from "../slices/authenticationSlice";

export const loginAction = (authData) => async (dispatch) => {

    const res = await login(authData);

    if (res?.data?.login === true) {

        localStorage.setItem("login", true);

        localStorage.setItem("token", res?.data?.token);

        localStorage.setItem("nav", JSON.stringify(res?.data?.menu));

        localStorage.setItem("userName", res?.data?.tblUser?.userName);

        localStorage.setItem("userEmail", res?.data?.tblUser?.email);

        localStorage.setItem("isActive", res?.data?.tblUser?.isActive);

        localStorage.setItem("userRole", res?.data?.tblUser?.roleId);

        localStorage.setItem("userId", res?.data?.tblUser?.userId);

        dispatch(LOGIN\_SUCCESS(res?.data));

        return res;

    } else {

        dispatch(LOGIN\_ERROR("Error"));

        return res;

    }

};

**#loadingAction.js:**

import { setTeamData } from "../slices/utilitySlice";

export const teamAction = (data) => async (dispatch) => {

    dispatch(setTeamData(data));

};

**Dist:**

**#utilitySlice.dev.js:**

"use strict";

Object.defineProperty(exports, "\_\_esModule", {

  value: true

});

exports["default"] = exports.setTeamData = exports.slice = void 0;

var \_toolkit = require("@reduxjs/toolkit");

function \_toConsumableArray(arr) { return \_arrayWithoutHoles(arr) || \_iterableToArray(arr) || \_nonIterableSpread(); }

function \_nonIterableSpread() { throw new TypeError("Invalid attempt to spread non-iterable instance"); }

function \_iterableToArray(iter) { if (Symbol.iterator in Object(iter) || Object.prototype.toString.call(iter) === "[object Arguments]") return Array.from(iter); }

function \_arrayWithoutHoles(arr) { if (Array.isArray(arr)) { for (var i = 0, arr2 = new Array(arr.length); i < arr.length; i++) { arr2[i] = arr[i]; } return arr2; } }

function ownKeys(object, enumerableOnly) { var keys = Object.keys(object); if (Object.getOwnPropertySymbols) { var symbols = Object.getOwnPropertySymbols(object); if (enumerableOnly) symbols = symbols.filter(function (sym) { return Object.getOwnPropertyDescriptor(object, sym).enumerable; }); keys.push.apply(keys, symbols); } return keys; }

function \_objectSpread(target) { for (var i = 1; i < arguments.length; i++) { var source = arguments[i] != null ? arguments[i] : {}; if (i % 2) { ownKeys(source, true).forEach(function (key) { \_defineProperty(target, key, source[key]); }); } else if (Object.getOwnPropertyDescriptors) { Object.defineProperties(target, Object.getOwnPropertyDescriptors(source)); } else { ownKeys(source).forEach(function (key) { Object.defineProperty(target, key, Object.getOwnPropertyDescriptor(source, key)); }); } } return target; }

function \_defineProperty(obj, key, value) { if (key in obj) { Object.defineProperty(obj, key, { value: value, enumerable: true, configurable: true, writable: true }); } else { obj[key] = value; } return obj; }

var slice = (0, \_toolkit.createSlice)({

  name: "utilitySlice",

  initialState: {

    teamData: []

  },

  reducers: {

    setTeamData: function setTeamData(state, action) {

      return \_objectSpread({}, state, {

        teamData: [].concat(\_toConsumableArray(state.teamData), [action.payload])

      });

    }

  }

});

exports.slice = slice;

var setTeamData = slice.actions.setTeamData;

exports.setTeamData = setTeamData;

var \_default = slice.reducer;

exports["default"] = \_default;

**Constant.js:**

import axios from "axios";

export const API = axios.create({

  baseURL: 'http://localhost:5000',

});

**Pages:**