I have the following website that will be used by an admin, authors and moderators

Admin posts events in which there many sections

Authors choose an event and choose a section in the event and file a form of submission and upload a file

Moderators are assigned to sections by the admin when the admin is creating the event, whenever the admin ads a new sections he assign a moderator to it.

The flow of the website is as follows,   
the website has a home landing page, and header where there are three buttons,   
. be a moderator

. be an author

. sign in

And also events nav button and home

Each button takes the users to its corresponding form where he can register,

**Moderator prospective**.

Mod land on the home page, press the “be a moderator” button, and fill the form,

After submitting the form an email verification is send to the mod email address,

Mod go to his email address and press the email verification link

Mods needs to be verified be email and admin as well, so they can’t sign in to there account unless they are verified by email and admin, so after verifying their email they need to contact the admin to verify them.

After admin verify a mod, he can log in to his account

**Admin prospective**.

Admin is seeded to the database be developers

Admin go to home page of the website, press sign in and fill the sign in form, then a sidebar appears, where there are navs manging events, mods, news, submissions

Admin navigates to the evens sections and press add event,

Admin fill out the new event form and can ad many sections and for each section he assign a moderator.

After submitting the form , the event is saved as draft until the admin changes the status of the event to another status like, upcoming, ongoing finished.

Events are viewed publicly on the events page on the website, but drafts are not viewed

**Author prospectives**

Author register like mods but no need for admin verification

If author is signed in, in the public events page, he can open an event and an add submission button is appeared under the opened event, but submission buttons is not viewed if the user is not an author and also if the event is not ongoing

When an author filles the form and upload a doc file the submission is saved and pedding for reviewing by the mods

Author can check the status of the submissions in his submissions tab

Mods have submission page where only submissions of the sections in which they are assigned are appeard, and there is a grading form, where they can put grade for the submission from 1 to 10 and change the status of the submission from pending to accepted or rejected and can also add a comment

After all submissions are graded and event is finished, admin can change the status of the event to finished and press button to publish winners,

When publish winners is pressed in the event page, tables of winners in each secession is appeared

Models of the website  
const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const userSchema = new Schema(

    {

        password: {

            type: String,

            required: true,

        },

        email: {

            type: String,

            required: true,

            unique: true,

            trim: true,

            match: [/.+@.+\..+/, "Please enter a valid email address"],

        },

        phoneNumber: {

            type: String,

            trim: true,

        },

        verifiedByEmail: {

            type: Boolean,

            default: false,

        },

        role: {

            type: String,

            enum: ["author", "admin", "mod"],

            default: "author",

        },

        firstName: {

            type: String,

            trim: true,

            required: true,

        },

        lastName: {

            type: String,

            trim: true,

            required: true,

        },

        middleName: {

            type: String,

            trim: true,

        },

        dateOfBirth: {

            type: Date,

        },

    },

    {

        timestamps: true,

        discriminatorKey: "userType",

        collection: "users",

    }

);

const User = mongoose.model("User", userSchema);

module.exports = User;

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const User = require('./userModel');

const modSchema = new Schema({

    faculty: {

        type: String,

        required: true,

        trim: true,

    },

    department: {

        type: String,

        required: true,

        trim: true,

    },

    jobTitle: {

        type: String,

        required: true,

        trim: true,

    },

    verifiedByAdmin: {

        type: Boolean,

        default: false,

    },

});

const Mod = User.discriminator('Mod', modSchema);

module.exports = Mod;

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const User = require('./userModel');

const authorSchema = new Schema({

    participantStatus: {

        type: String,

        enum: ["young scientist", "specialist", "undergraduate", "masters", "graduate"],

        trim: true,

        required: true,

    },

    region: {

        type: String,

        trim: true,

        required: true,

    },

    city: {

        type: String,

        trim: true,

        required: true,

    },

    university: {

        type: String,

        trim: true,

        required: true,

    },

    faculty: {

        type: String,

        trim: true,

        required: true,

    },

    department: {

        type: String,

        trim: true,

        required: true,

    },

    course: {

        type: Number,

        trim: true,

        required: true,

    },

    groupNumber:{

        type: String,

        trim: true,

        required: true,

    },

    submissions: [{

        type: Schema.Types.ObjectId,

        ref: 'Submission',

    }],

});

const Author = User.discriminator('Author', authorSchema);

module.exports = Author;

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const eventSchema = new Schema(

    {

        name: {

            type: String,

            required: true,

            trim: true,

        },

        description: {

            type: String,

            trim: true,

        },

        status: {

            type: String,

            enum: ["draft", "upcoming", "ongoing", "finished"],

            default: "draft",

        },

        sections: [

            {

                type: Schema.Types.ObjectId,

                ref: "Section",

            },

        ],

        createdBy: {

            type: Schema.Types.ObjectId,

            ref: "User",

            required: true,

        },

    },

    {

        timestamps: true,

        collection: "events",

    }

);

const Event = mongoose.model("Event", eventSchema);

module.exports = Event;

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const sectionSchema = new Schema(

    {

        name: {

            type: String,

            required: true,

            trim: true,

        },

        order: {

            type: Number,

            required: true,

        },

        mod: {

            type: Schema.Types.ObjectId,

            ref: "Mod",

            required: true,

        },

        event: {

            type: Schema.Types.ObjectId,

            ref: "Event",

        },

    },

    {

        timestamps: true,

        collection: "sections",

    }

);

const Section = mongoose.model("Section", sectionSchema);

module.exports = Section;

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const coauthorSchema = new Schema({

    fullName: {

        type: String,

        required: true,

        trim: true,

    },

    university: {

        type: String,

        required: true,

        trim: true,

    },

});

const submissionSchema = new Schema(

    {

        workName: {

            type: String,

            required: true,

            trim: true,

        },

        withPublication: {

            type: Boolean,

        },

        supervisorName: {

            type: String,

            required: true,

            trim: true,

        },

        supervisorAcademicDegree: {

            type: String,

            required: true,

            trim: true,

        },

        author: {

            type: Schema.Types.ObjectId,

            ref: "Author",

            required: true,

        },

        coauthors: [coauthorSchema],

        event: {

            type: Schema.Types.ObjectId,

            ref: "Event",

            required: true,

        },

        section: {

            type: Schema.Types.ObjectId,

            ref: "Section",

            required: true,

        },

        comments: [

            {

                moderator: {

                    type: Schema.Types.ObjectId,

                    ref: "Mod",

                    required: true,

                },

                comment: {

                    type: String,

                    required: true,

                    trim: true,

                },

                createdAt: {

                    type: Date,

                    default: Date.now,

                },

            },

        ],

        file: {

            type: String,

            required: true,

        },

        status: {

            type: String,

            enum: ["pendding", "accepted", "rejected"],

            default: "pendding",

        },

        grade: {

            type: Number,

            min: 0,

            max: 10,

        },

        grader: {

            type: Schema.Types.ObjectId,

            ref: "Mod",

        },

    },

    {

        timestamps: true,

        collection: "submissions",

    }

);

const Submission = mongoose.model("Submission", submissionSchema);

module.exports = Submission;

const mongoose = require("mongoose");

const Schema = mongoose.Schema;

const newsSchema = new Schema(

    {

        title: {

            type: String,

            trim: true,

        },

        content: {

            type: String,

            trim: true,

        },

        createdBy: {

            type: Schema.Types.ObjectId,

            ref: 'User',

            required: true,

        },

    },

    {

        timestamps: true,

        collection: "news",

    }

);

const News = mongoose.model("News", newsSchema);

module.exports = News;