Backend > models >

1. AdminSchema.js

The provided code defines a Mongoose schema for an Admin collection in a MongoDB database. Here's a detailed explanation of each part of the schema:

Importing Mongoose

- Imports the Mongoose library, which is used for modeling and interacting with MongoDB.

Defining the Admin Schema

1. **adminSchema**: Creates a new Mongoose schema for the Admin collection with the following fields:
    - **name**:
        - type: String: Specifies that the name field should be a string.
        - required: true: Indicates that the name field is mandatory.
    - **email**:
        - type: String: Specifies that the email field should be a string.
        - unique: true: Ensures that the email field is unique across all documents in the Admin collection, meaning no two admins can have the same email.
        - required: true: Indicates that the email field is mandatory.
    - **password**:
        - type: String: Specifies that the password field should be a string.
        - required: true: Indicates that the password field is mandatory.
    - **role**:
        - type: String: Specifies that the role field should be a string.
        - default: "Admin": Sets the default value of the role field to "Admin" if no value is provided.
    - **schoolName**:
        - type: String: Specifies that the schoolName field should be a string.
        - unique: true: Ensures that the schoolName field is unique across all documents in the Admin collection, meaning no two admins can be associated with the same school name.
        - required: true: Indicates that the schoolName field is mandatory.

Exporting the Model

- Exports the Mongoose model for the Admin collection, allowing it to be used in other parts of the application.
- mongoose.model("admin", adminSchema) creates a model named admin based on the adminSchema. This model will allow you to perform CRUD (Create, Read, Update, Delete) operations on the Admin collection in the MongoDB database.

Summary

- The adminSchema defines the structure of the Admin collection with the fields name, email, password, role, and schoolName.
- Each field has specific constraints like type, required, unique, and default.
- The model is exported for use in other parts of the application to interact with the Admin collection in MongoDB.

---

2. complainSchema.js

The provided code defines a Mongoose schema for a Complain collection in a MongoDB database. Here's a detailed explanation of each part of the schema:

Importing Mongoose

- Imports the Mongoose library, which is used for modeling and interacting with MongoDB.

Defining the Complain Schema

1. **complainSchema**: Creates a new Mongoose schema for the Complain collection with the following fields:
    - **user**:
        - type: mongoose.Schema.Types.ObjectId: Specifies that the user field should be an ObjectId, which is a reference to another document in a different collection.
        - ref: 'student': Indicates that the user field references the student collection, establishing a relationship between the Complain and Student collections.
        - required: true: Indicates that the user field is mandatory.
    - **date**:
        - type: Date: Specifies that the date field should be a Date object.
        - required: true: Indicates that the date field is mandatory.
    - **complaint**:
        - type: String: Specifies that the complaint field should be a string.
        - required: true: Indicates that the complaint field is mandatory.

- **school**:
  - type: mongoose.Schema.Types.ObjectId: Specifies that the school field should be an ObjectId, which is a reference to another document in a different collection.
  - ref: 'admin': Indicates that the school field references the admin collection, establishing a relationship between the Complain and Admin collections.
  - required: true: Indicates that the school field is mandatory.

Exporting the Model

module.exports = mongoose.model("complain", complainSchema);

- Exports the Mongoose model for the Complain collection, allowing it to be used in other parts of the application.
- mongoose.model("complain", complainSchema) creates a model named complain based on the complainSchema. This model will allow you to perform CRUD (Create, Read, Update, Delete) operations on the Complain collection in the MongoDB database.

Summary

- The complainSchema defines the structure of the Complain collection with the fields user, date, complaint, and school.
- Each field has specific constraints like type, required, and ref (for establishing references to other collections).
- The model is exported for use in other parts of the application to interact with the Complain collection in MongoDB.

This schema facilitates the recording of complaints made by students, associating each complaint with a specific student and school, and including the date and details of the complaint.

3.noticeSchema.js

The provided code defines a Mongoose schema for a Notice collection in a MongoDB database. Here's a detailed explanation of each part of the schema:

Importing Mongoose

- Imports the Mongoose library, which is used for modeling and interacting with MongoDB.

Defining the Notice Schema

1. **noticeSchema**: Creates a new Mongoose schema for the Notice collection with the following fields:
   - **title**:
     - type: String: Specifies that the title field should be a string.
     - required: true: Indicates that the title field is mandatory.
   - **details**:
     - type: String: Specifies that the details field should be a string.
     - required: true: Indicates that the details field is mandatory.
   - **date**:
     - type: Date: Specifies that the date field should be a Date object.
     - required: true: Indicates that the date field is mandatory.
   - **school**:
     - type: mongoose.Schema.Types.ObjectId: Specifies that the school field should be an ObjectId, which is a reference to another document in a different collection.
     - ref: 'admin': Indicates that the school field references the admin collection, establishing a relationship between the Notice and Admin collections.
2. **timestamps**:
   - { timestamps: true }: Adds two additional fields, createdAt and updatedAt, to the schema. These fields automatically store the timestamp for when a document is created and last updated, respectively.

Exporting the Model

module.exports = mongoose.model("notice", noticeSchema)

- Exports the Mongoose model for the Notice collection, allowing it to be used in other parts of the application.
- mongoose.model("notice", noticeSchema) creates a model named notice based on the noticeSchema. This model will allow you to perform CRUD (Create, Read, Update, Delete) operations on the Notice collection in the MongoDB database.

Summary

- The noticeSchema defines the structure of the Notice collection with the fields title, details, date, and school.
- Each field has specific constraints like type, required, and ref (for establishing references to the admin collection).
- The schema also includes the { timestamps: true } option to automatically add createdAt and updatedAt fields.

- The model is exported for use in other parts of the application to interact with the Notice collection in MongoDB.

This schema facilitates the recording of notices, associating each notice with a specific school, and including the title, details, and date of the notice.

4.sclassSchema.js

The provided code defines a Mongoose schema for an SClass collection in a MongoDB database. Here's a detailed explanation of each part of the schema:

Importing Mongoose

- Imports the Mongoose library, which is used for modeling and interacting with MongoDB.

Defining the SClass Schema

1. **sclassSchema**: Creates a new Mongoose schema for the SClass collection with the following fields:
    - **sclassName**:
        - type: String: Specifies that the sclassName field should be a string.
        - required: true: Indicates that the sclassName field is mandatory.
    - **school**:
        - type: mongoose.Schema.Types.ObjectId: Specifies that the school field should be an ObjectId, which is a reference to another document in a different collection.
        - ref: 'admin': Indicates that the school field references the admin collection, establishing a relationship between the SClass and Admin collections.
2. **timestamps**:
    - { timestamps: true }: Adds two additional fields, createdAt and updatedAt, to the schema. These fields automatically store the timestamp for when a document is created and last updated, respectively.

Exporting the Model

```
module.exports = mongoose.model("sclass", sclassSchema);
```

- Exports the Mongoose model for the SClass collection, allowing it to be used in other parts of the application.
- mongoose.model("sclass", sclassSchema) creates a model named sclass based on the sclassSchema. This model will allow you to perform CRUD (Create, Read, Update, Delete) operations on the SClass collection in the MongoDB database.

Summary

- The sclassSchema defines the structure of the SClass collection with the fields sclassName and school.
- Each field has specific constraints like type, required, and ref (for establishing references to the admin collection).
- The schema also includes the { timestamps: true } option to automatically add createdAt and updatedAt fields.
- The model is exported for use in other parts of the application to interact with the SClass collection in MongoDB.

This schema facilitates the recording of class details, associating each class with a specific school, and including the class name and a reference to the associated school.

5.studentSchema.js

The provided code defines a Mongoose schema for a Student collection in a MongoDB database. Here's a detailed explanation of each part of the schema:

Importing Mongoose

```
const mongoose = require('mongoose');
```

- Imports the Mongoose library, which is used for modeling and interacting with MongoDB.

Defining the Student Schema Fields in the Schema

1. **Basic Information Fields**:
    - **name**:
        - type: String: Specifies that the name field should be a string.
        - required: true: Indicates that the name field is mandatory.
    - **rollNum**:

        - type: Number: Specifies that the rollNum field should be a number.
        - required: true: Indicates that the rollNum field is mandatory.
    - **password**:
        - type: String: Specifies that the password field should be a string.
        - required: true: Indicates that the password field is mandatory.
2. **Class and School References**:
    - **sclassName**:
        - type: mongoose.Schema.Types.ObjectId: Specifies that the sclassName field should be an ObjectId.
        - ref: 'sclass': Indicates that the sclassName field references the sclass collection.
        - required: true: Indicates that the sclassName field is mandatory.
    - **school**:
        - type: mongoose.Schema.Types.ObjectId: Specifies that the school field should be an ObjectId.
        - ref: 'admin': Indicates that the school field references the admin collection.
        - required: true: Indicates that the school field is mandatory.
3. **Role Field**:
    - **role**:
        - type: String: Specifies that the role field should be a string.
        - default: "Student": Sets the default value of the role field to "Student".
4. **Exam Results Field**:
    - **examResult**:
        - Array of objects, each object containing:
            - **subName**:
                - type: mongoose.Schema.Types.ObjectId: Specifies that the subName field should be an ObjectId.
                - ref: 'subject': Indicates that the subName field references the subject collection.
            - **marksObtained**:
                - type: Number: Specifies that the marksObtained field should be a number.
                - default: 0: Sets the default value of the marksObtained field to 0.
5. **Attendance Field**:
    - **attendance**:
        - Array of objects, each object containing:
            - **date**:
                - type: Date: Specifies that the date field should be a date.
                - required: true: Indicates that the date field is mandatory.
            - **status**:
                - type: String: Specifies that the status field should be a string.
                - enum: ['Present', 'Absent']: Restricts the status field to either "Present" or "Absent".
                - required: true: Indicates that the status field is mandatory.
            - **subName**:
                - type: mongoose.Schema.Types.ObjectId: Specifies that the subName field should be an ObjectId.
                - ref: 'subject': Indicates that the subName field references the subject collection.
                - required: true: Indicates that the subName field is mandatory.

Exporting the Model

- Exports the Mongoose model for the Student collection, allowing it to be used in other parts of the application.
- mongoose.model("student", studentSchema) creates a model named student based on the studentSchema. This model will allow you to perform CRUD (Create, Read, Update, Delete) operations on the Student collection in the MongoDB database.

Summary

- The studentSchema defines the structure of the Student collection with fields for storing the student's name, roll number, password, class, school, role, exam results, and attendance records.
- Each field has specific constraints, and some fields reference other collections, establishing relationships between them.
- The model is exported for use in other parts of the application to interact with the Student collection in MongoDB.

6.subjectSchema.js

The provided code defines a Mongoose schema for a Subject collection in a MongoDB database. Here is a detailed explanation of each part of the schema:

Importing Mongoose

const mongoose = require("mongoose");

- Imports the Mongoose library, which is used for modeling and interacting with MongoDB.

Defining the Subject Schema

Fields in the Schema

1. **Basic Information Fields**:
    - **subName**:
        - type: String: Specifies that the subName field should be a string.
        - required: true: Indicates that the subName field is mandatory.
    - **subCode**:
        - type: String: Specifies that the subCode field should be a string.
        - required: true: Indicates that the subCode field is mandatory.
    - **sessions**:
        - type: String: Specifies that the sessions field should be a string.
        - required: true: Indicates that the sessions field is mandatory.
2. **Class and School References**:
    - **sclassName**:
        - type: mongoose.Schema.Types.ObjectId: Specifies that the sclassName field should be an ObjectId.
        - ref: 'sclass': Indicates that the sclassName field references the sclass collection.
        - required: true: Indicates that the sclassName field is mandatory.
    - **school**:
        - type: mongoose.Schema.Types.ObjectId: Specifies that the school field should be an ObjectId.
        - ref: 'admin': Indicates that the school field references the admin collection.
3. **Teacher Reference**:
    - **teacher**:
        - type: mongoose.Schema.Types.ObjectId: Specifies that the teacher field should be an ObjectId.
        - ref: 'teacher': Indicates that the teacher field references the teacher collection.

Timestamps Option

- timestamps: true: Adds createdAt and updatedAt fields to the schema. These fields automatically store the creation and last update times of each document.

Exporting the Model

module.exports = mongoose.model("subject", subjectSchema);

- Exports the Mongoose model for the Subject collection, allowing it to be used in other parts of the application.
- mongoose.model("subject", subjectSchema) creates a model named subject based on the subjectSchema. This model will allow you to perform CRUD (Create, Read, Update, Delete) operations on the Subject collection in the MongoDB database.

Summary

- The subjectSchema defines the structure of the Subject collection with fields for storing the subject's name, code, sessions, class reference, school reference, and teacher reference.
- Each field has specific constraints, and some fields reference other collections, establishing relationships between them.
- The model is exported for use in other parts of the application to interact with the Subject collection in MongoDB.

7.teacherSchema.js

The provided code defines a Mongoose schema for a Teacher collection in a MongoDB database. Here is a detailed explanation of each part of the schema:

Importing Mongoose

- Imports the Mongoose library, which is used for modeling and interacting with MongoDB.

Defining the Teacher Schema

Fields in the Schema

1. **Basic Information Fields**:
    - **name**:
        - type: String: Specifies that the name field should be a string.
        - required: true: Indicates that the name field is mandatory.
    - **email**:
        - type: String: Specifies that the email field should be a string.

        - unique: true: Ensures that each email is unique across the collection.
        - required: true: Indicates that the email field is mandatory.
    - **password**:
        - type: String: Specifies that the password field should be a string.
        - required: true: Indicates that the password field is mandatory.
    - **role**:
        - type: String: Specifies that the role field should be a string.
        - default: "Teacher": Sets the default value of the role field to "Teacher".
2. **References to Other Collections**:
    - **school**:
        - type: mongoose.Schema.Types.ObjectId: Specifies that the school field should be an ObjectId.
        - ref: 'admin': Indicates that the school field references the admin collection.
        - required: true: Indicates that the school field is mandatory.
    - **teachSubject**:
        - type: mongoose.Schema.Types.ObjectId: Specifies that the teachSubject field should be an ObjectId.
        - ref: 'subject': Indicates that the teachSubject field references the subject collection.
    - **teachSclass**:
        - type: mongoose.Schema.Types.ObjectId: Specifies that the teachSclass field should be an ObjectId.
        - ref: 'sclass': Indicates that the teachSclass field references the sclass collection.
        - required: true: Indicates that the teachSclass field is mandatory.
3. **Attendance Field**:
    - **attendance**: An array of attendance records:
        - **date**:
            - type: Date: Specifies that the date field should be a date.
            - required: true: Indicates that the date field is mandatory.
        - **presentCount**:
            - type: String: Specifies that the presentCount field should be a string.
        - **absentCount**:
            - type: String: Specifies that the absentCount field should be a string.

Timestamps Option

- timestamps: true: Adds createdAt and updatedAt fields to the schema. These fields automatically store the creation and last update times of each document.

Exporting the Model

- Exports the Mongoose model for the Teacher collection, allowing it to be used in other parts of the application.
- mongoose.model("teacher", teacherSchema) creates a model named teacher based on the teacherSchema. This model will allow you to perform CRUD (Create, Read, Update, Delete) operations on the Teacher collection in the MongoDB database.

Summary

- The teacherSchema defines the structure of the Teacher collection with fields for storing the teacher's name, email, password, role, school reference, subject reference, class reference, and attendance records.
- Each field has specific constraints, and some fields reference other collections, establishing relationships between them.
- The model is exported for use in other parts of the application to interact with the Teacher collection in MongoDB.