

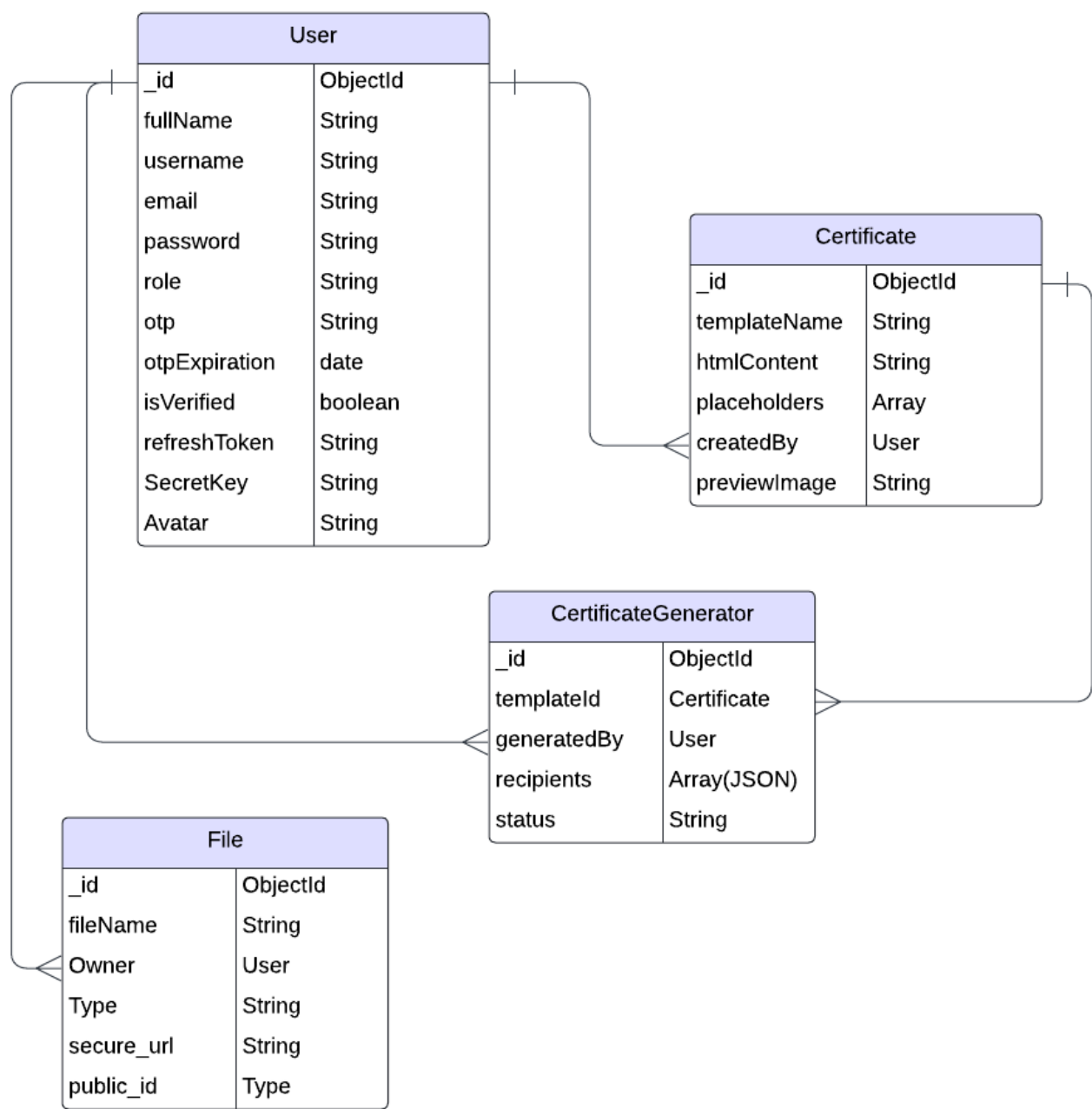
Comsats University Islamabad



Assignment 4

Submitted By:	Muhammad Danyal Rana
Registration No.	FA22-BCS-057
Course Title:	CSC-336
Submitted To:	Mam Sadia Maqbool
Date:	25 December 2024

Visual Representation



Schemas

1. User Schema

```
const userSchema = new Schema({
  fullName: {
    type: String,
    required: true,
    trim: true,
    index: true,
  },
  username: {
    type: String,
    required: true,
    unique: true,
    lowercase: true,
    trim: true,
    index: true, // used for searching
  },
  email: {
    type: String,
    required: true,
    unique: true,
    lowercase: true,
    trim: true,
    match: [/^\S+@\S+\.\S+$/, 'is invalid'],
  },
  password: {
    type: String,
    required: [true, "Password is required"],
  },
  role: {
    type: String,
    enum: ["user", "admin"],
    default: "user",
  },
  otp: {
    type: String,
    default: null
  },
  otpExpiration: {
    type: Date,
    default: null
  },
  isVerified: {
    type: Boolean, default: false
  },
  refreshToken: {
    type: String,
  },
  secretKey: {
    type: String,
    default: null,
  },
  avatar: {
    type: String, // cloudinary url
  },
}, { timestamps: true });

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

2. Certificate Template Schema

```
const certificateTemplateSchema = new mongoose.Schema({
  templateName: {
    type: String,
    required: true
  },
  htmlContent: {
    type: String,
    required: true
  },
  placeholders: [
    {
      type: String
    }
  ],
  createdBy: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  previewImage: {
    type: String
  },
}, { timestamps: true });
```

3. File Schema

```
const fileSchema = new mongoose.Schema({
  fileName: {
    type: String,
    required: true
  },
  owner: {
    type: mongoose.Schema.Types.ObjectId,
    ref: 'User',
    required: true
  },
  type: {
    type: String,
    enum: ["csv", "xlsx", "vnd.openxmlformats-officedocument.spreadsheetml.sheet"],
  },
  public_id: { type: String },
  secure_url: { type: String },
}, {
  timestamps: true
});
```

4. Generated Certificate Schema

```
const genCertificateSchema = new mongoose.Schema({
  templateId: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "CertificateTemplate",
    required: true,
  },
  generatedBy: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "User",
    required: true,
  },
  recipients: [
    {
      name: { type: String, required: true },
      email: { type: String, required: true },
      certificateUrl: { type: String },
    },
  ],
  status: {
    type: String,
    enum: ["completed", "failed"],
    default: "completed"
  },
}, { timestamps: true });
```

Queries

Authentication Queries

- Find user by username:
- User.findOne({ username });

- **Create new user:**
- `const newUser = new User(userData);`
- `newUser.save();`

File Operations

- **Upload a file:**
- `const newFile = new File(fileData);`
- `newFile.save();`
- **Find all files by owner:**
- `File.find({ owner: userId });`
- **Delete a file by ID:**
- `File.findByIdAndDelete(fileId);`

Certificate Operations

- **Generate certificates:**
- `const generatedCertificate = new GenCertificate(certificateData);`
- `generatedCertificate.save();`
- **Find certificates by template ID:**
- `GenCertificate.find({ templateId });`

Security Features

1. **Authentication:** JWT-based authentication.
2. **Authorization:** Role-based access control (user/admin).
3. **Data Validation:** Schema-enforced validation for accurate data.
4. **Cloud Integration:** Secure file storage using Cloudinary.

Conclusion

CertiMeet simplifies the process of generating and distributing certificates with a robust and user-friendly backend structure. The schemas and operations ensure a smooth workflow for template creation, file management, and certificate generation.