College Automation Portal - Database Schema & MongoDB Queries

Relational Database Schema

1. Users Table

```
Users {
  user_id: ObjectId (Primary Key)
  student_id: String (Unique, for students)
  email: String (Unique)
  password_hash: String
  first_name: String
  last_name: String
  role: String (ENUM: 'admin', 'teacher', 'student')
  phone: String
  department: String
  created_at: DateTime
  updated_at: DateTime
  is_active: Boolean
}
```

2. Courses Table

```
Courses {
  course_id: ObjectId (Primary Key)
  course_code: String (Unique)
  course_name: String
  description: String
  credits: Number
  semester: String
  department: String
  teacher_id: ObjectId (Foreign Key -> Users.user_id)
  created_at: DateTime
  updated_at: DateTime
  is_active: Boolean
}
```

3. Course_Enrollments Table

Course_Enrollments {

enrollment_id: ObjectId (Primary Key)

student_id: ObjectId (Foreign Key -> Users.user_id)

course_id: ObjectId (Foreign Key -> Courses.course_id)

enrollment_date: DateTime

status: String (ENUM: 'enrolled', 'dropped', 'completed')

}

4. Notes Table

Notes {

note_id: ObjectId (Primary Key)

course_id: ObjectId (Foreign Key -> Courses.course_id)

teacher_id: ObjectId (Foreign Key -> Users.user_id)

title: String

description: String

file_name: String

file_path: String

file_type: String (ENUM: 'pdf', 'ppt', 'doc', 'other')

file_size: Number

upload_date: DateTime

is_active: Boolean

}

5. Grades Table

Grades {

grade_id: ObjectId (Primary Key)

student_id: ObjectId (Foreign Key -> Users.user_id)

course_id: ObjectId (Foreign Key -> Courses.course_id)

teacher_id: ObjectId (Foreign Key -> Users.user_id)

grade_type: String (ENUM: 'internal', 'external')

marks_obtained: Number

total_marks: Number

grade_letter: String

comments: String

created_at: DateTime

updated_at: DateTime

}

6. Assessments Table

Assessments {

assessment_id: ObjectId (Primary Key)

course_id: ObjectId (Foreign Key -> Courses.course_id)

teacher_id: ObjectId (Foreign Key -> Users.user_id)

title: String

description: String

assessment_type: String (ENUM: 'objective', 'subjective', 'mixed')

total_marks: Number

duration_minutes: Number

start_time: DateTime

end_time: DateTime

instructions: String

is_published: Boolean

created_at: DateTime

updated_at: DateTime

}

7. Assessment_Questions Table

Assessment_Questions {

question_id: ObjectId (Primary Key)

assessment_id: ObjectId (Foreign Key -> Assessments.assessment_id)

question_text: String

question_type: String (ENUM: 'mcq', 'short_answer', 'essay')

options: Array (for MCQ questions)

correct_answer: String

marks: Number

order_number: Number

}

8. Student_Assessments Table

Student_Assessments {

submission_id: ObjectId (Primary Key)

assessment_id: ObjectId (Foreign Key -> Assessments.assessment_id)

student_id: ObjectId (Foreign Key -> Users.user_id)

start_time: DateTime

end_time: DateTime

submission_time: DateTime

status: String (ENUM: 'not_started', 'in_progress', 'submitted', 'graded')

total_marks_obtained: Number

auto_graded: Boolean

teacher_feedback: String

created_at: DateTime

updated_at: DateTime

}

9. Student_Answers Table

Student_Answers {

answer_id: ObjectId (Primary Key)

submission_id: ObjectId (Foreign Key -> Student_Assessments.submission_id)

question_id: ObjectId (Foreign Key -> Assessment_Questions.question_id)

student_answer: String

marks_obtained: Number

is_correct: Boolean

graded_at: DateTime

}

10. Announcements Table

Announcements {

announcement_id: ObjectId (Primary Key)

course_id: ObjectId (Foreign Key -> Courses.course_id)

teacher_id: ObjectId (Foreign Key -> Users.user_id)

title: String

content: String

priority: String (ENUM: 'low', 'medium', 'high')

created_at: DateTime

updated_at: DateTime

is_active: Boolean

}

MongoDB Collection Creation Queries

1. Create Users Collection

```
db.createCollection("users", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["email", "password_hash", "first_name", "last_name", "role"],
      properties: {
        student_id: { bsonType: "string" },
        email: { bsonType: "string" },
        password_hash: { bsonType: "string" },
        first_name: { bsonType: "string" },
        last_name: { bsonType: "string" },
        role: { enum: ["admin", "teacher", "student"] },
        phone: { bsonType: "string" },
        department: { bsonType: "string" },
        created_at: { bsonType: "date" },
        updated_at: { bsonType: "date" },
        is_active: { bsonType: "bool" }
      }
    }
  }
});
```

```
// Create indexes
db.users.createIndex({ "email": 1 }, { unique: true });
db.users.createIndex({ "student_id": 1 }, { unique: true, sparse: true });
db.users.createIndex({ "role": 1 });
```

2. Create Courses Collection

```
db.createCollection("courses", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["course_code", "course_name", "teacher_id"],
      properties: {
        course_code: { bsonType: "string" },
        course_name: { bsonType: "string" },
        description: { bsonType: "string" },
        credits: { bsonType: "number" },
        semester: { bsonType: "string" },
        department: { bsonType: "string" },
        teacher_id: { bsonType: "objectId" },
        created_at: { bsonType: "date" },
        updated_at: { bsonType: "date" },
        is_active: { bsonType: "bool" }
      }
    }
  }
});

// Create indexes
db.courses.createIndex({ "course_code": 1 }, { unique: true });
db.courses.createIndex({ "teacher_id": 1 });
db.courses.createIndex({ "department": 1 });
```

3. Create Course_Enrollments Collection

```
db.createCollection("course_enrollments", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["student_id", "course_id", "enrollment_date"],
      properties: {
        student_id: { bsonType: "objectId" },
        course_id: { bsonType: "objectId" },
        enrollment_date: { bsonType: "date" },
        status: { enum: ["enrolled", "dropped", "completed"] }
      }
    }
  }
});

// Create indexes
db.course_enrollments.createIndex({ "student_id": 1, "course_id": 1 }, { unique: true });
db.course_enrollments.createIndex({ "course_id": 1 });
```

4. Create Notes Collection

```
db.createCollection("notes", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["course_id", "teacher_id", "title", "file_name", "file_path"],
      properties: {
        course_id: { bsonType: "objectId" },
        teacher_id: { bsonType: "objectId" },
        title: { bsonType: "string" },
        description: { bsonType: "string" },
        file_name: { bsonType: "string" },
```

```
      file_path: { bsonType: "string" },
      file_type: { enum: ["pdf", "ppt", "doc", "other"] },
      file_size: { bsonType: "number" },
      upload_date: { bsonType: "date" },
      is_active: { bsonType: "bool" }
    }
  }
 }
});

// Create indexes
db.notes.createIndex({ "course_id": 1 });
db.notes.createIndex({ "teacher_id": 1 });
db.notes.createIndex({ "upload_date": -1 });
```

5. Create Grades Collection

```
db.createCollection("grades", {
 validator: {
  $jsonSchema: {
   bsonType: "object",
   required: ["student_id", "course_id", "teacher_id", "grade_type", "marks_obtained",
"total_marks"],
   properties: {
    student_id: { bsonType: "objectId" },
    course_id: { bsonType: "objectId" },
    teacher_id: { bsonType: "objectId" },
    grade_type: { enum: ["internal", "external"] },
    marks_obtained: { bsonType: "number" },
    total_marks: { bsonType: "number" },
    grade_letter: { bsonType: "string" },
    comments: { bsonType: "string" },
    created_at: { bsonType: "date" },
```

```
      updated_at: { bsonType: "date" }
    }
   }
  }
});

// Create indexes
db.grades.createIndex({ "student_id": 1, "course_id": 1 });
db.grades.createIndex({ "course_id": 1 });
db.grades.createIndex({ "teacher_id": 1 });
```

6. Create Assessments Collection

```
db.createCollection("assessments", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["course_id", "teacher_id", "title", "assessment_type", "total_marks",
"duration_minutes"],
      properties: {
        course_id: { bsonType: "objectId" },
        teacher_id: { bsonType: "objectId" },
        title: { bsonType: "string" },
        description: { bsonType: "string" },
        assessment_type: { enum: ["objective", "subjective", "mixed"] },
        total_marks: { bsonType: "number" },
        duration_minutes: { bsonType: "number" },
        start_time: { bsonType: "date" },
        end_time: { bsonType: "date" },
        instructions: { bsonType: "string" },
        is_published: { bsonType: "bool" },
        created_at: { bsonType: "date" },
        updated_at: { bsonType: "date" }
```

```
    }
   }
  }
});

// Create indexes
db.assessments.createIndex({ "course_id": 1 });
db.assessments.createIndex({ "teacher_id": 1 });
db.assessments.createIndex({ "start_time": 1, "end_time": 1 });
```

7. Create Assessment_Questions Collection

```
db.createCollection("assessment_questions", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["assessment_id", "question_text", "question_type", "marks", "order_number"],
      properties: {
        assessment_id: { bsonType: "objectId" },
        question_text: { bsonType: "string" },
        question_type: { enum: ["mcq", "short_answer", "essay"] },
        options: { bsonType: "array" },
        correct_answer: { bsonType: "string" },
        marks: { bsonType: "number" },
        order_number: { bsonType: "number" }
      }
    }
  }
});

// Create indexes
db.assessment_questions.createIndex({ "assessment_id": 1 });
db.assessment_questions.createIndex({ "assessment_id": 1, "order_number": 1 });
```

8. Create Student_Assessments Collection

```
db.createCollection("student_assessments", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["assessment_id", "student_id"],
      properties: {
        assessment_id: { bsonType: "objectId" },
        student_id: { bsonType: "objectId" },
        start_time: { bsonType: "date" },
        end_time: { bsonType: "date" },
        submission_time: { bsonType: "date" },
        status: { enum: ["not_started", "in_progress", "submitted", "graded"] },
        total_marks_obtained: { bsonType: "number" },
        auto_graded: { bsonType: "bool" },
        teacher_feedback: { bsonType: "string" },
        created_at: { bsonType: "date" },
        updated_at: { bsonType: "date" }
      }
    }
  }
});

// Create indexes
db.student_assessments.createIndex({ "assessment_id": 1, "student_id": 1 }, { unique: true });
db.student_assessments.createIndex({ "student_id": 1 });
db.student_assessments.createIndex({ "status": 1 });
```

9. Create Student_Answers Collection

```
db.createCollection("student_answers", {
  validator: {
    $jsonSchema: {
```

```
      bsonType: "object",
      required: ["submission_id", "question_id", "student_answer"],
      properties: {
        submission_id: { bsonType: "objectId" },
        question_id: { bsonType: "objectId" },
        student_answer: { bsonType: "string" },
        marks_obtained: { bsonType: "number" },
        is_correct: { bsonType: "bool" },
        graded_at: { bsonType: "date" }
      }
    }
  }
});

// Create indexes
db.student_answers.createIndex({ "submission_id": 1 });
db.student_answers.createIndex({ "question_id": 1 });
db.student_answers.createIndex({ "submission_id": 1, "question_id": 1 }, { unique: true });
```

10. Create Announcements Collection

```
db.createCollection("announcements", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["course_id", "teacher_id", "title", "content"],
      properties: {
        course_id: { bsonType: "objectId" },
        teacher_id: { bsonType: "objectId" },
        title: { bsonType: "string" },
        content: { bsonType: "string" },
        priority: { enum: ["low", "medium", "high"] },
        created_at: { bsonType: "date" },
```

```
      updated_at: { bsonType: "date" },
      is_active: { bsonType: "bool" }
    }
   }
  }
});

// Create indexes
db.announcements.createIndex({ "course_id": 1 });
db.announcements.createIndex({ "teacher_id": 1 });
db.announcements.createIndex({ "created_at": -1 });
```

Key Design Considerations

1. **User Roles**: The schema supports three user roles (admin, teacher, student) with role-based access control.
2. **Course Management**: Teachers can manage multiple courses, and students can enroll in multiple courses through the enrollment table.
3. **File Management**: Notes table stores file metadata with paths for actual file storage (local or cloud).
4. **Assessment System**: Supports both objective (MCQ) and subjective questions with automatic grading capabilities.
5. **Flexible Grading**: Separate grades table for internal and external assessments with detailed feedback.
6. **Performance Optimization**: Proper indexing on frequently queried fields and foreign key relationships.
7. **Data Integrity**: MongoDB validators ensure data consistency and required field validation.
8. **Scalability**: Schema designed to handle large numbers of users, courses, and assessments efficiently.

This schema provides a solid foundation for the College Automation Portal with room for future enhancements like real-time chat, analytics, and notification systems.