Center for Professional Courses
Gujarat University

Subject: Backend Application Development Assignment
Faculty: Meezan Mallick
Course: M.Sc. IT Data Management and Visual Insights (5 years Integrated)
SEMESTER 4

Deadline to Submit: 20 April 2025

Part A: Express + MongoDB (MVC) CRUD

Question 1: Student Management System (CRUD)

Objective:
Create a simple Student Management system using Node.js, Express, MongoDB, and MVC architecture.

Database: school
Collection: students
Fields:

```
{
  "name": "Student Name",
  "rollno": "101",
  "class": "10th Grade",
  "section": "A"
}
```

Tasks:
- GET /students → List all students
- POST /students → Add a new student
- PUT /students/:id → Update a student's details
- DELETE /students/:id → Delete a student

Requirements:
- Use Mongoose for the schema.
- Follow MVC pattern:
  - Model → Mongoose Student Schema
  - Controller → All CRUD logic

- Routes → API Endpoints
- Return JSON response for all APIs.

Question 2: Employee Management System (CRUD)

Objective:
Create an Employee Management system using Node.js, Express, MongoDB, and MVC architecture.

Database: company
Collection: employees
Fields:

```
{
  "name": "Employee Name",
  "position": "Software Engineer",
  "department": "IT",
  "salary": 60000
}
```

Tasks:
- GET /employees → List all employees
- POST /employees → Add a new employee
- PUT /employees/:id → Update employee details
- DELETE /employees/:id → Delete an employee

Requirements:
- Validate that salary must be a positive number.
- Follow MVC structure.
- Handle errors if employee ID is not found.

Part B: File System (FS) Module

Question 3: Simple File Creation and Reading

Objective:
Use Node.js fs module to create a file, write into it, and read from it.

Tasks:
1. Create a file named message.txt.
2. Write the following text into the file:
   "Hello, this is a sample text written using Node.js!"
3. After writing, read the contents of message.txt.

4. Print the file contents to the console.

Requirements:
- Use Node's built-in fs module.
- Use either fs.writeFile and fs.readFile (callback style) or fs.promises.writeFile and fs.promises.readFile (async/await).
- Handle any errors if they occur during writing or reading.