DBMS Assignment 4

Payroll Management System

**Team ID: 7**

**Team Details:**

| **Mahesh Kumawat** | **PES1UG19CS250** |
| --- | --- |
| **Manish P T** | **PES1UG19CS261** |
| **Manish Kote S** | **PES1UG19CS260** |

Simple User interface design for front end:

For our website we have used Node js, postgres, and express js, along with other dependencies.

**Express JS :**  Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications.

**Node JS :**  Node JS is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.

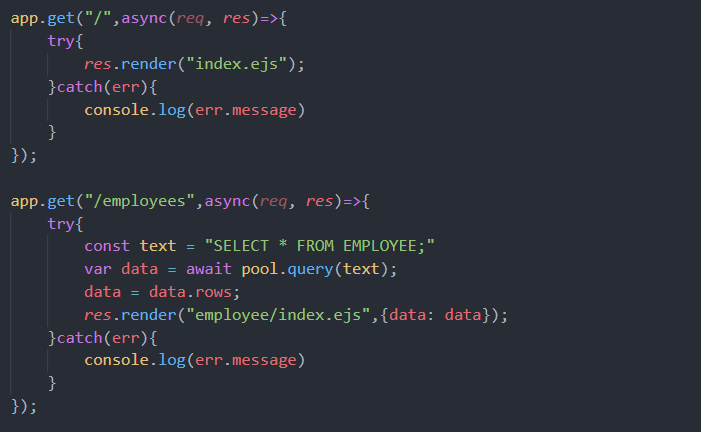
**EJS Template :** EJS is a simple templating language that lets you generate HTML markup with plain JavaScript.

# **pg-pool :** A connection pool for node-postgres

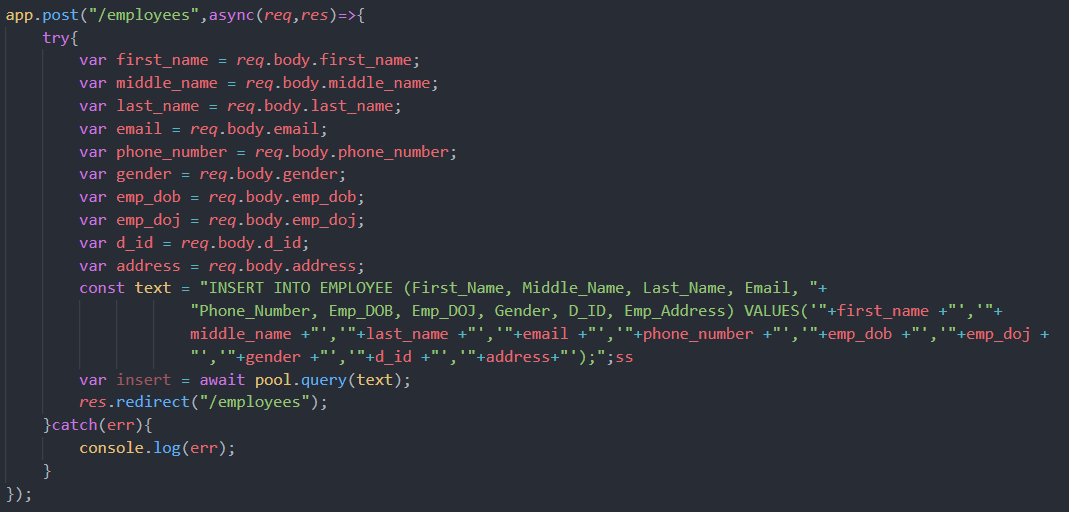


**Bootstrap :** Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

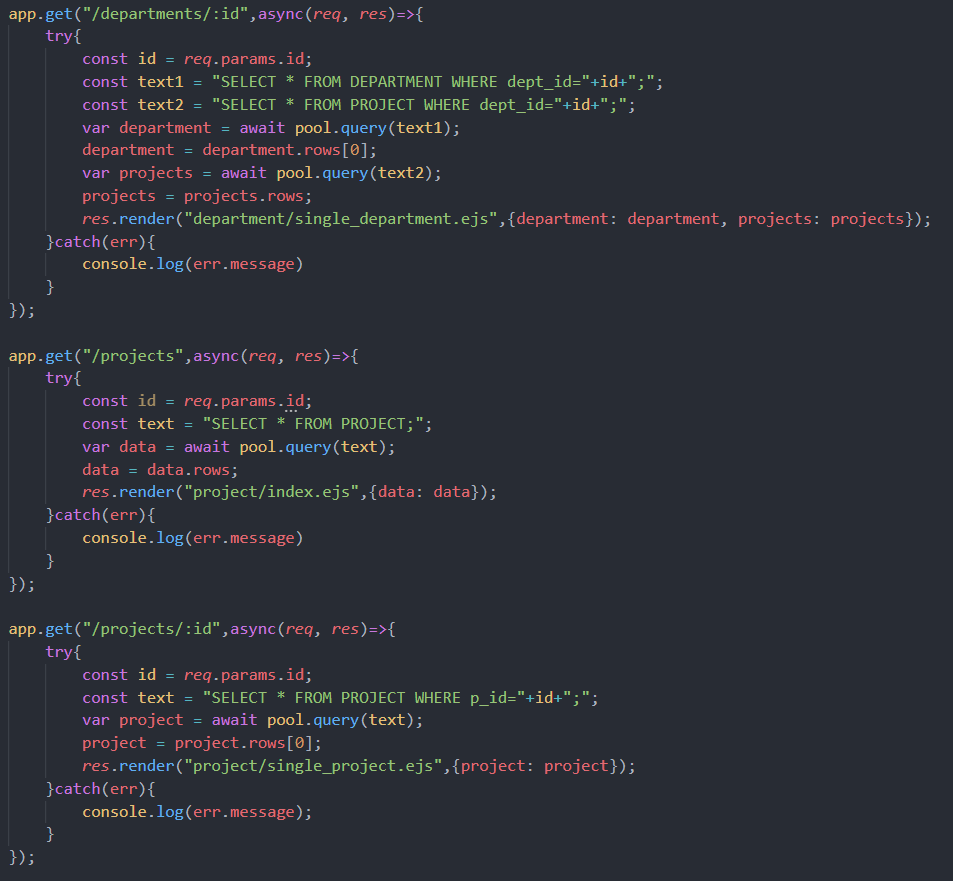
Executing Postgres queries whenever a specific route is requested:

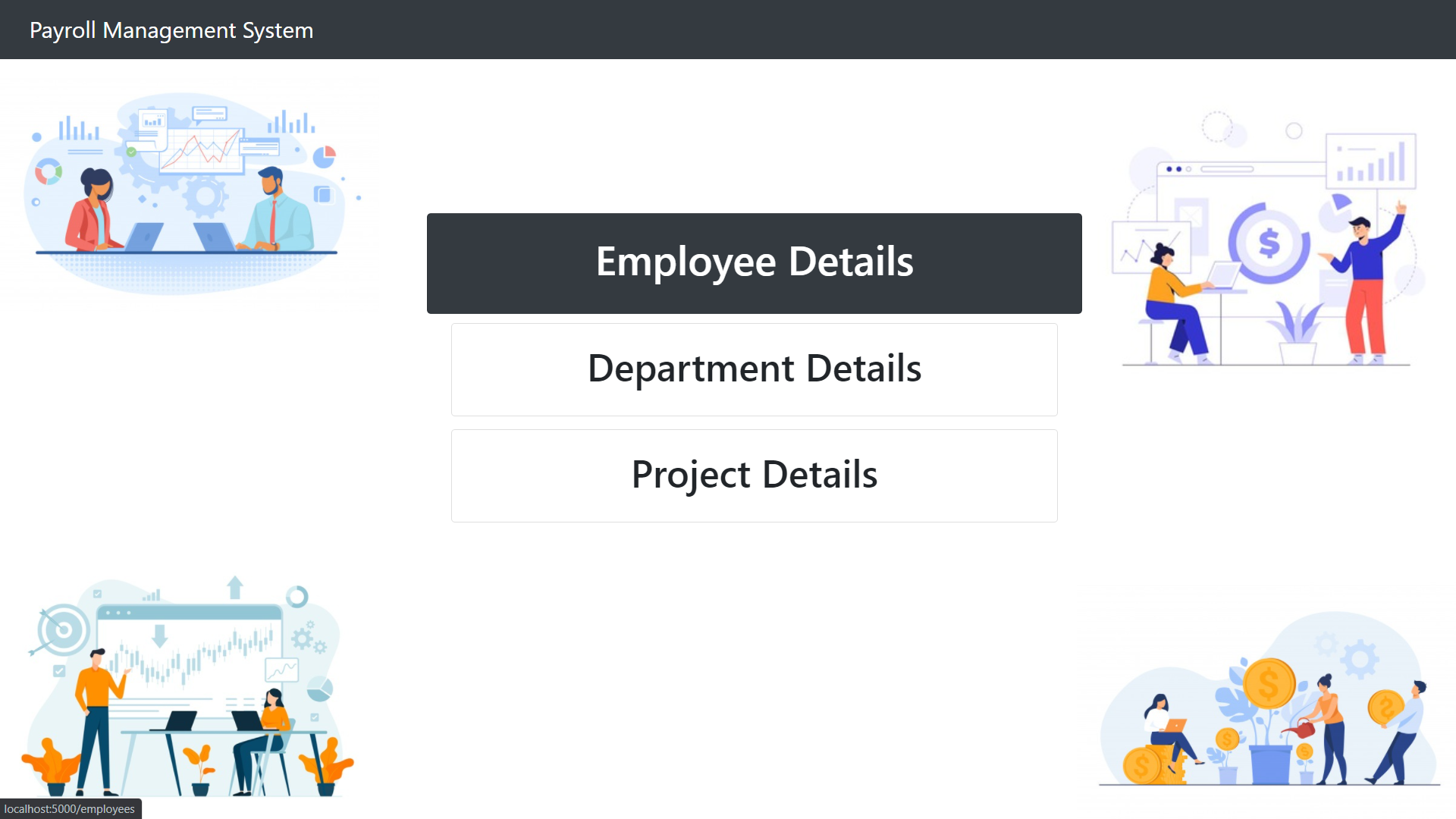


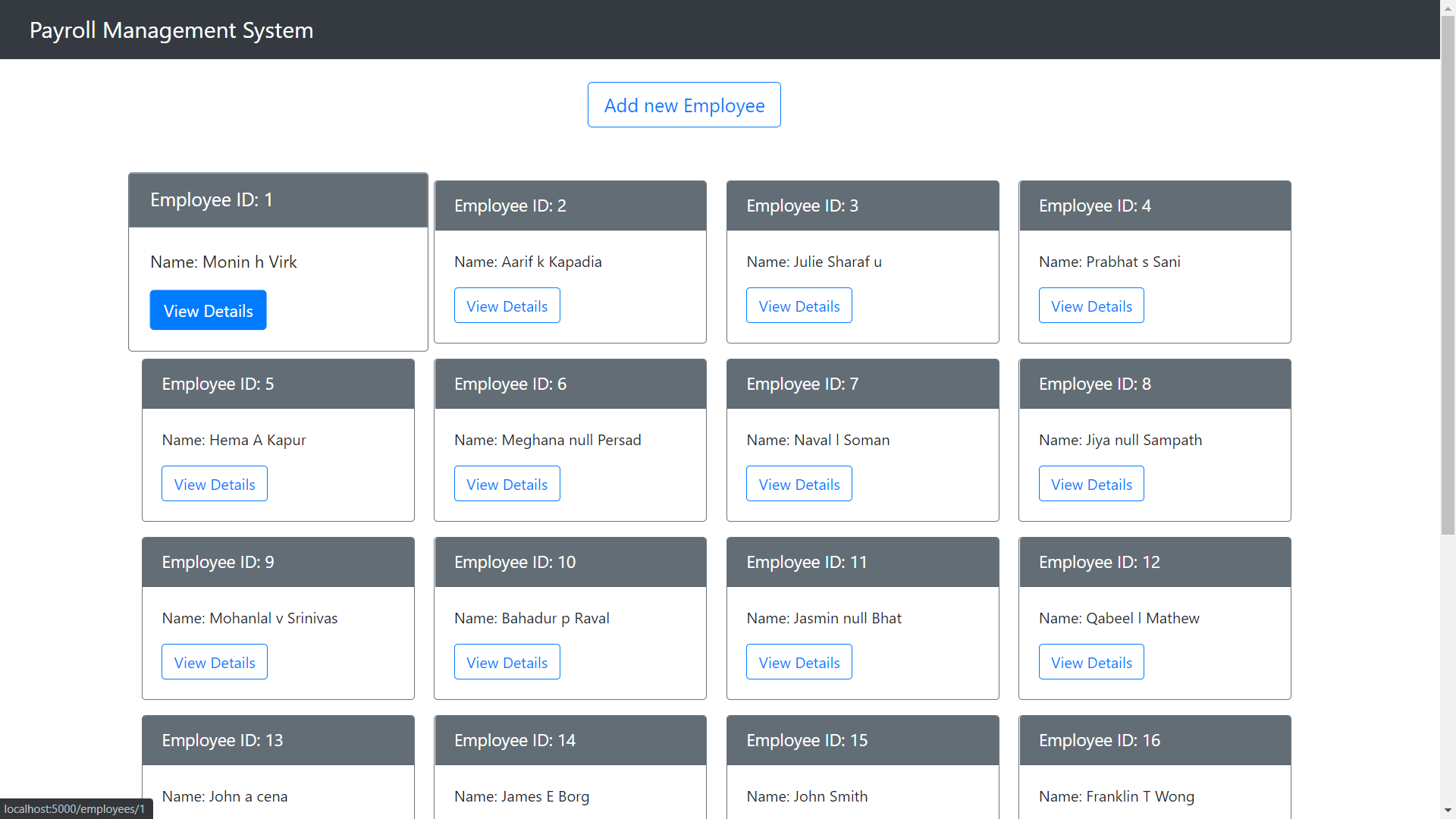


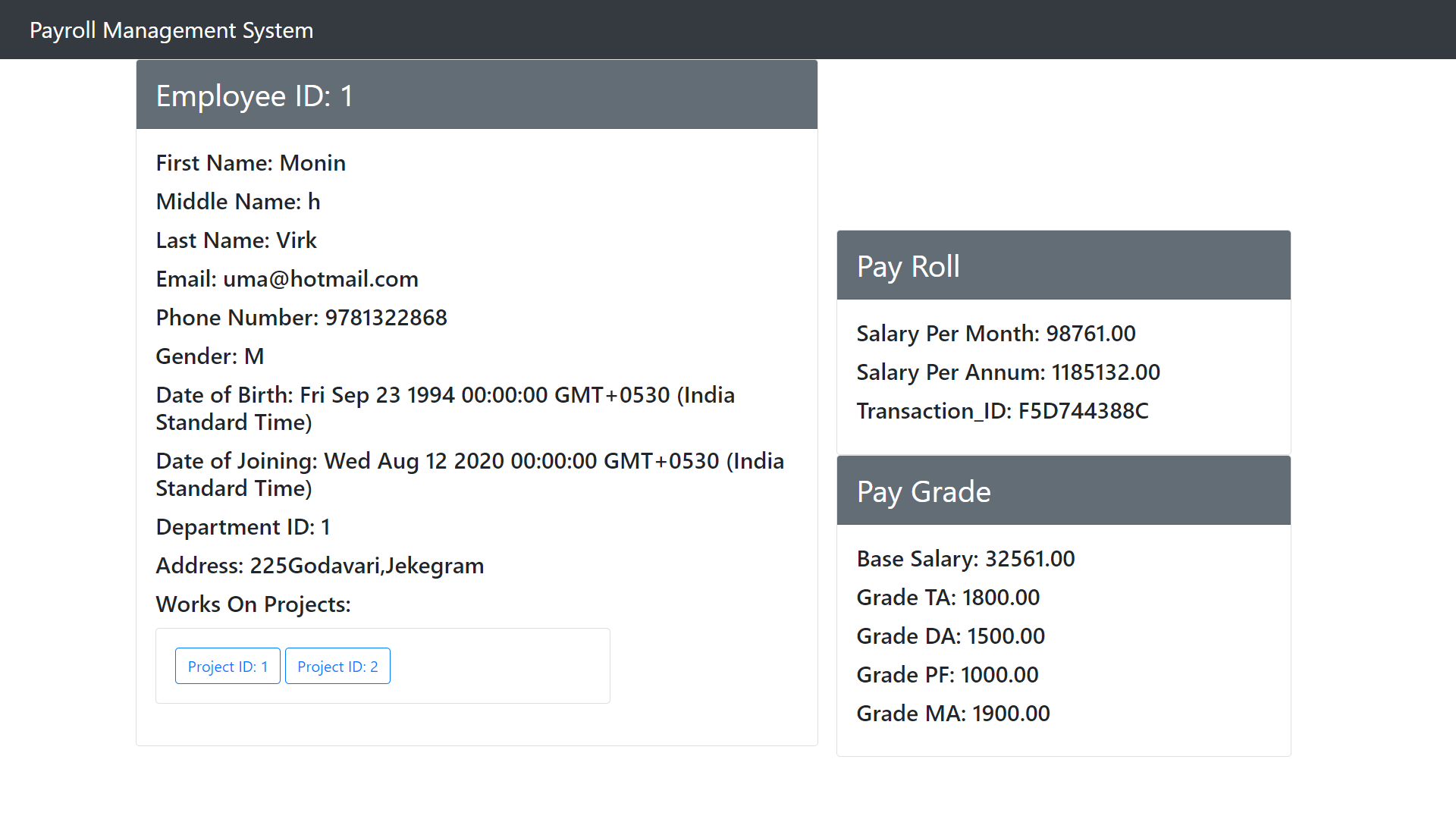




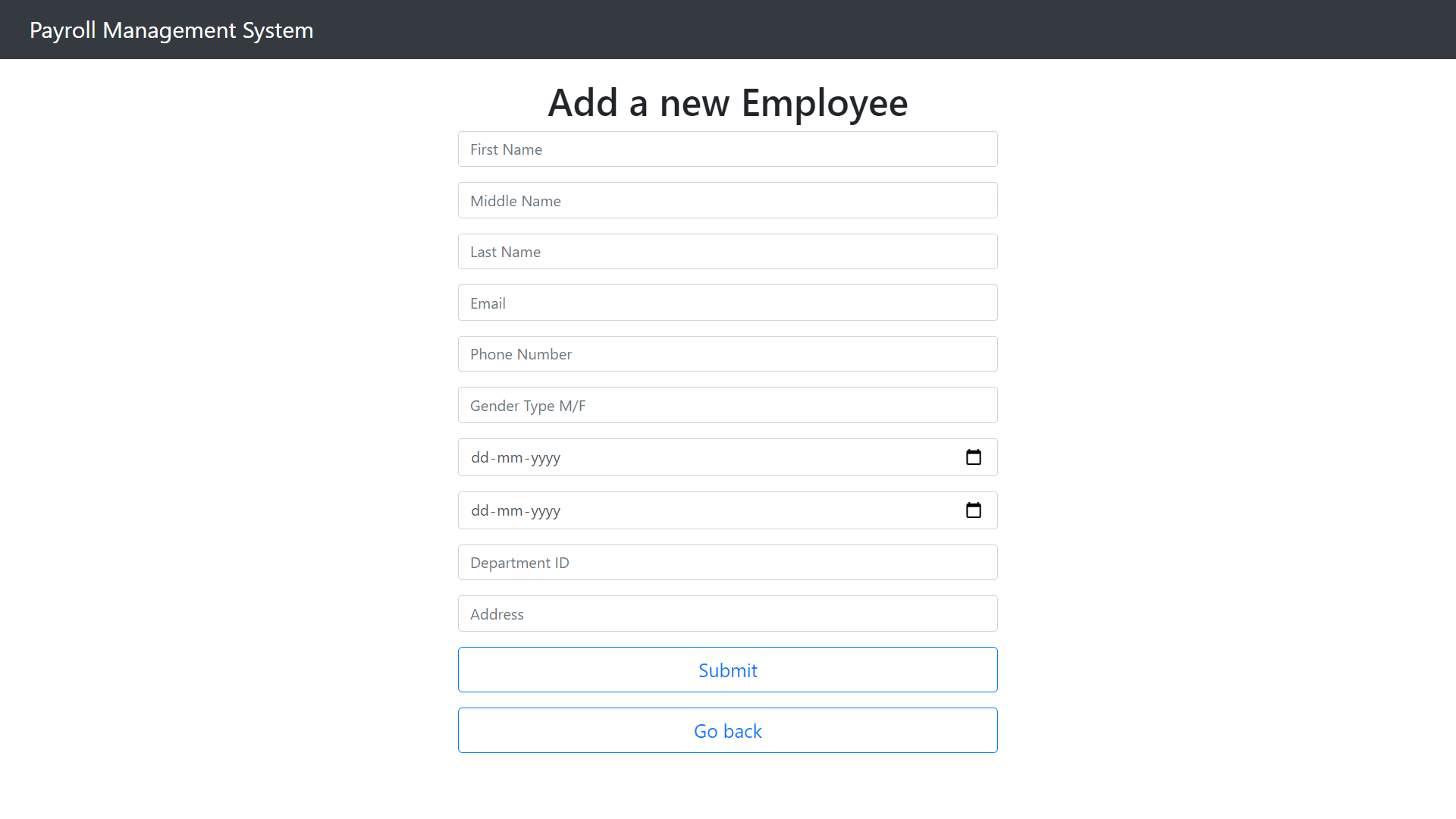


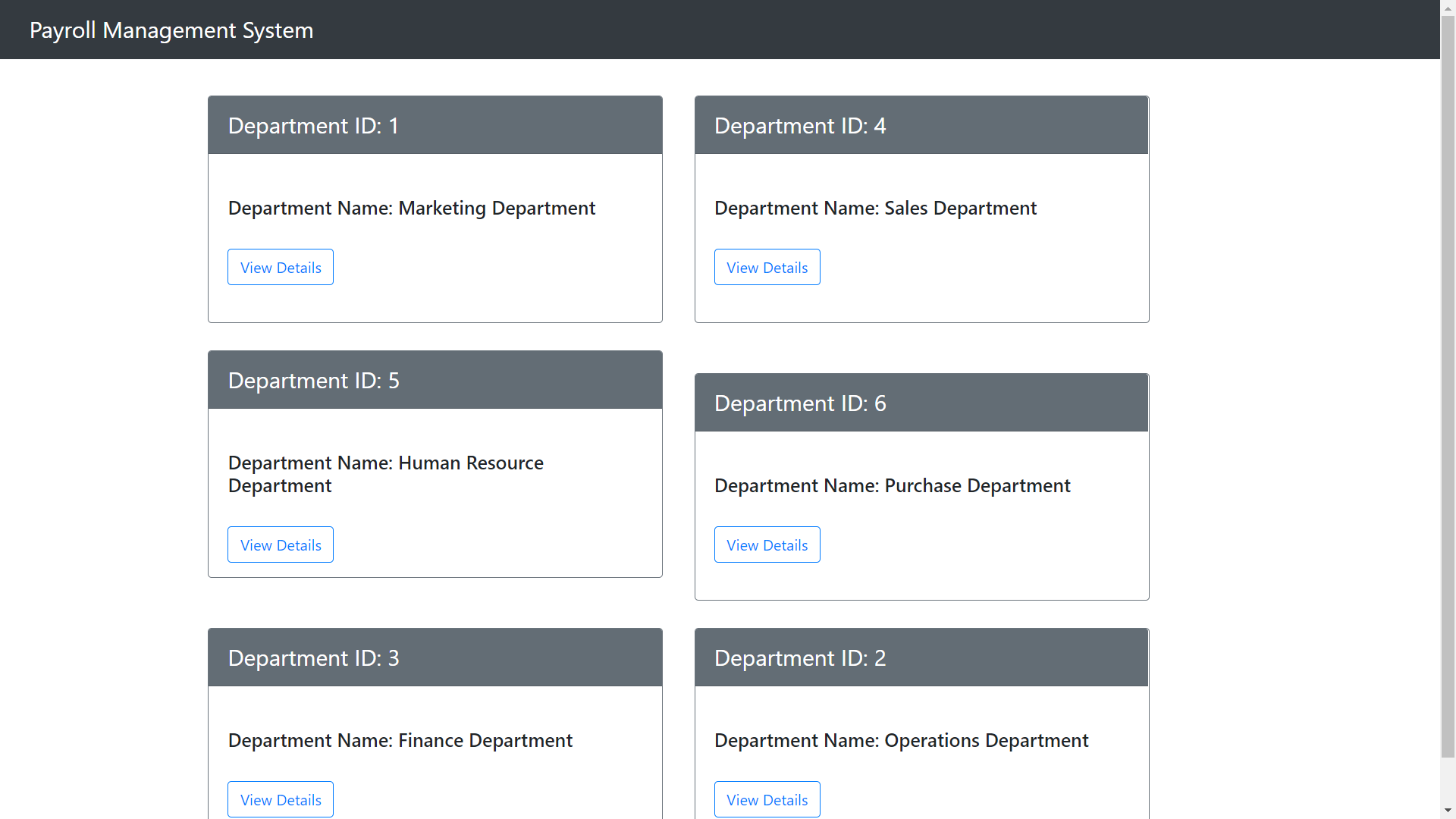
**Screenshots of the website:**

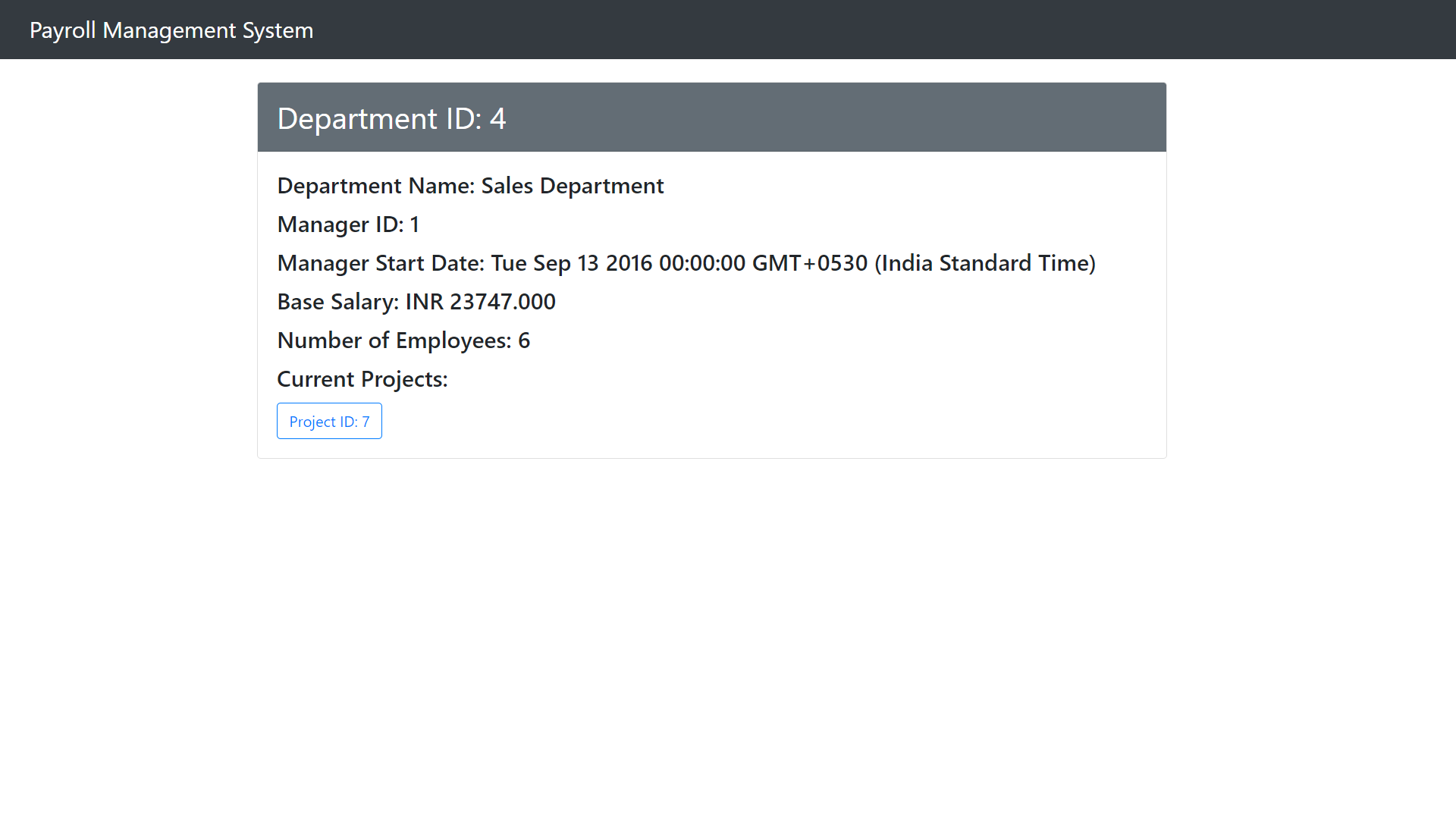
All Employee Details

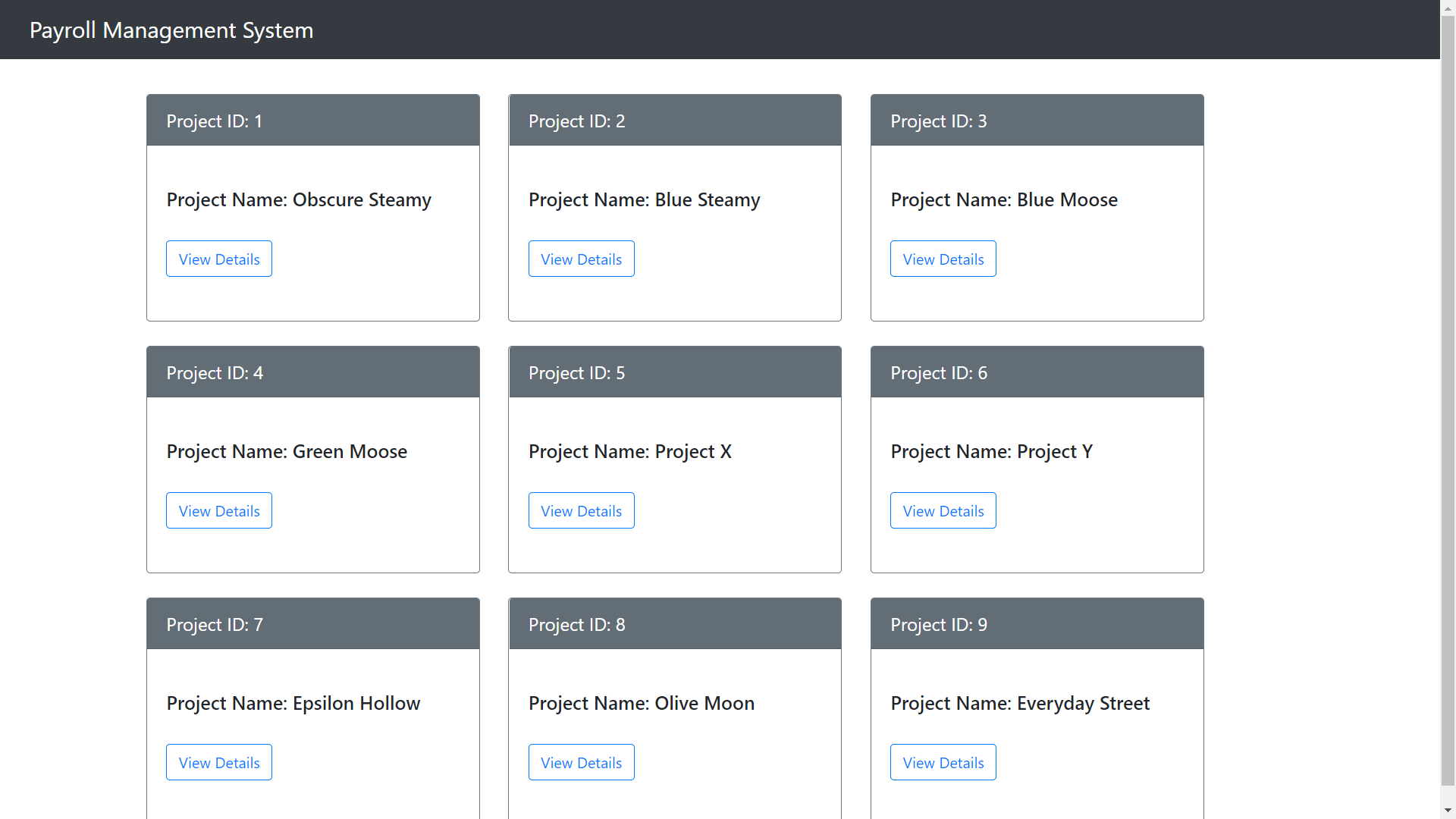
Specific Employee Detail

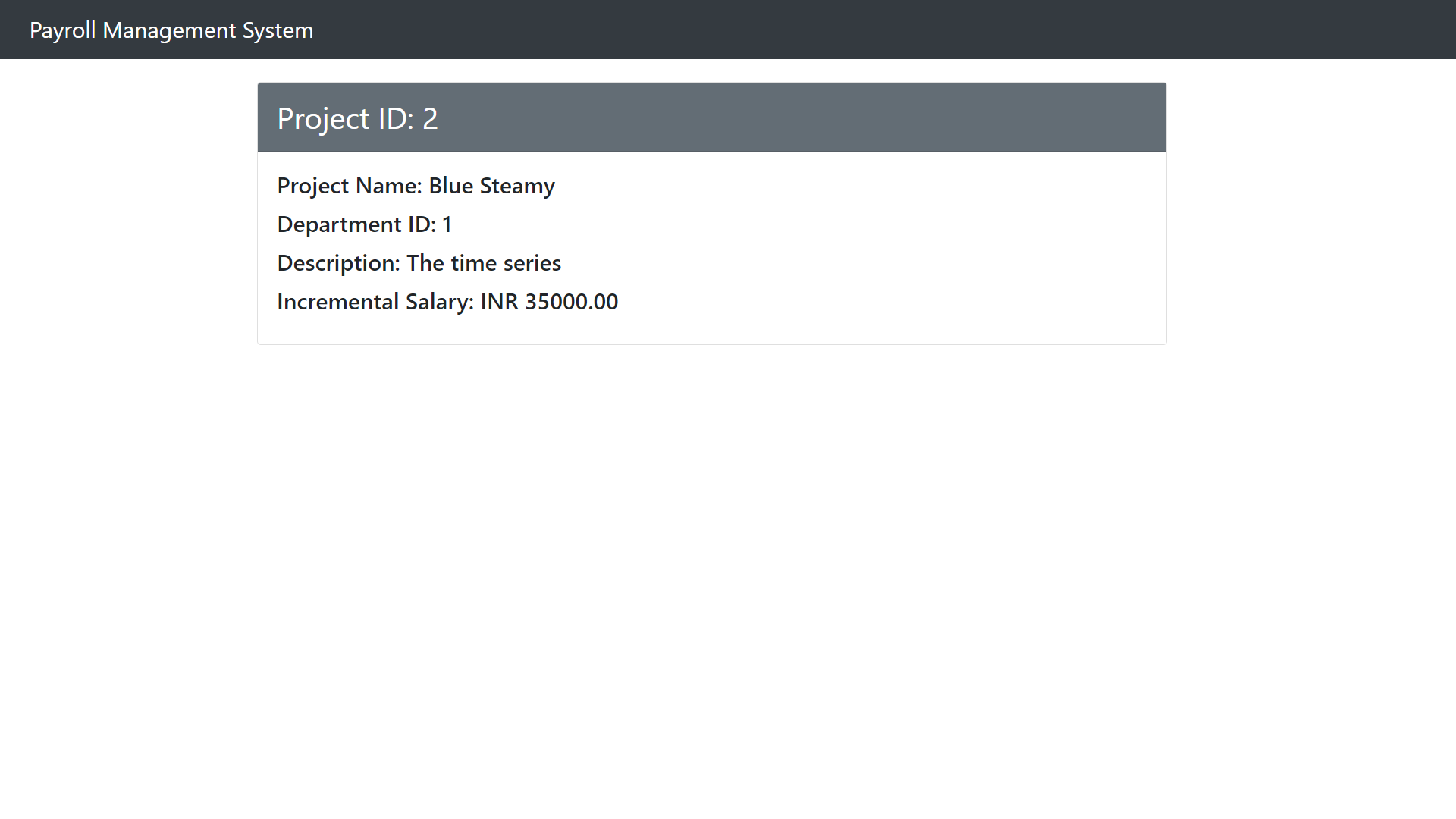
Adding a New Employee to the database



Department Details

Specific Department Details

All Projects

Specific Project Detail

Additional queries:

**● List change in constraints and schema**

**● Apply the changes to the existing database schema**

**● Atleast 5 statement examples for the above case**

**Change in constraints and schema for the existing database:**

1. Adding a check constraint on the employee age such that no employee is above the age of 60.

Code:

| ALTER TABLE EMPLOYEE ADD CONSTRAINT CK1 CHECK(Emp\_DOB > '1961-01-01'); |
| --- |

2. Dropping the unnecessary foreign key constraint which we added during assignment 2.

Code:

| ALTER TABLE DEPARTMENT DROP CONSTRAINT department\_mgr\_id\_fkey; |
| --- |

3. Renaming the long name used for dependents name from Dependent\_Name TO Dpn\_Name(a short one for ease of reference).

Code:

| ALTER TABLE DEPENDENTS RENAME COLUMN Dependent\_Name TO Dpn\_Name; |
| --- |

4. Adding a default value for the attribute Project description as ‘software engineering’. In case of missing the field during insertion.

Code:

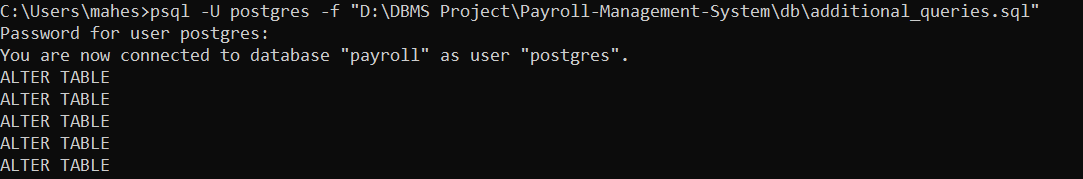
| ALTER TABLE PROJECTS ALTER COLUMN P\_Desc SET DEFAULT 'software engineering'; |
| --- |

5(and 6). Adding a field called project\_locations to the relation PROJECT and altering the table to make the attribute as a foreign key referring to Dept\_location.

Code:

| ALTER TABLE PROJECTS ADD COLUMN IF NOT EXISTS Project\_locations VARCHAR(255);  ALTER TABLE PROJECTS ADD CONSTRAINT FK1 FOREIGN KEY(Project\_locations) REFERENCES DEPT\_LOCATIONS(Dept\_location); |
| --- |

OUTPUT:



Changes in schema due to possible expansion:

Possible expansion could be in areas of growth of number of departments,

Could lead to creating various divisions among the department itself leading to new Projects related to respective departments. And expansion of geographical working space to accommodate the above additions.

All the above-mentioned changes can be easily accommodable into the existing schemas and will lead to only a vertical expansion of tables, i.e., addition of more rows, which isn’t an issue.

Other sets of possible changes that can be considered are reforms in the working load on each employee by limiting their work\_hrs to a maximum of 25 hrs. Which would also make equal work distribution possible.

| ALTER TABLE WORKS\_ON ADD CONSTRAINT CK1 CHECK(Work\_Hours < 25); |
| --- |

The last change that can be proposed is the addition of an attendance system for the employees.

We would have to an additional relation to the database, which will result in a change in schema,

Where for the table, columns need to be added for every working day, which will take a Boolean value,

And a foreign key referencing the employee\_id from the table EMPLOYEE has to be added for maintaining a separate record of each employee’s attendance.

Migration of existing database to any other flavor of relational database and WHY?

PostgreSQL is an Object Relational Database Management System (ORDBMS) whereas MySQL is a community driven DBMS system. Database migration from flat files has been done manually based on mongoDB objects . So if we are changing to a nosql database we would choose Column-Oriented Databases because while a relational database stores data in rows and reads data row by row, a column store is organized as a set of columns. This means that when you want to run analytics on a small number of columns, you can read those columns directly without consuming memory with the unwanted data. Columns are often of the same type and benefit from more efficient compression, making reads even faster. Columnar databases can quickly aggregate the value of a given column.

**Why choose MySQL:**

Here, are some important reasons for using MYSQL:

* Supports features like Master-Slave Replication, Scale-Out
* It supports Offload Reporting, Geographic Data Distribution, etc.
* Very Low overhead with MyISAM storage engine when used for read-mostly applications
* Support for Memory storage engine for frequently used tables
* Query Cache for repeatedly used statements
* We can easily learn and troubleshoot MySQL from different sources like blogs, white papers, and books

**Steps for migration:**

**1. Download the script and unpack the archive.**

wget http://www.lightbox.ca/pg2mysql/pg2mysql-1.9.tar.bz2

wget https://documentation.easyredmine.com/s/TXDM47G

tar -xvf pg2mysql-1.9.tar.bz2

unzip pg2mysql-1.9.zip

**2. Install php**

sudo apt install php7.4-cli

**3. We have to create a dump of PostgreSQL database in .sql format, and make sure we use "--format p --inserts"**

sudo -u postgres pg\_dump --format p --inserts DBNAME\_HERE > /path/to/file.sql

**4. Switch to pg2mysql-1.9 folder**

cd pg2mysql-1.9/

**5. Run**

php pg2mysql\_cli.php /path/to/pd/dump/file.sql /path/where/to/save/mysql/file.sql

We will see some lines like

Completed! 30820 lines 5539 sql chunks

**6. Now a file with modified sql dump will be created. We have to replace MyISAM to InnoDB everywhere**

**7. Now we may restore this dump into a clear mysql database.**

**8. Because the script doesn't save indexes we have to add them manually to every table (it will work without indexes but it may cause serious performance issues). There is a list of all indexes that should exist in MySQL tables generated by the regular easy redmine application. We need to add them manually or via our own custom script.**

**Additional Code added:  
Added Trigger functions for Employee count in Department.**

**CREATE OR REPLACE FUNCTION employee\_insert()**

**RETURNS TRIGGER**

**LANGUAGE PLPGSQL**

**AS**

**$$**

**BEGIN**

**UPDATE department**

**SET Number\_of\_Emp = Number\_of\_Emp + 1**

**WHERE Dept\_ID = NEW.D\_ID;**

**RETURN NEW;**

**END;**

**$$;**

**CREATE OR REPLACE FUNCTION employee\_delete()**

**RETURNS TRIGGER**

**LANGUAGE PLPGSQL**

**AS**

**$$**

**BEGIN**

**UPDATE department**

**SET Number\_of\_Emp = Number\_of\_Emp - 1**

**WHERE Dept\_ID = OLD.D\_ID;**

**RETURN NEW;**

**END;**

**$$;**

**CREATE TRIGGER increment\_count**

**AFTER INSERT**

**ON employee**

**FOR EACH ROW**

**EXECUTE PROCEDURE employee\_insert();**

**CREATE TRIGGER decrement\_count**

**AFTER DELETE**

**ON employee**

**FOR EACH ROW**

**EXECUTE PROCEDURE employee\_delete();**

Team Contribution:

**All three of us have contributed equally to all the given tasks**

**Time spent: Mahesh - 5hrs, Manish PT - 5hrs , Manish Kote - 4hrs.**