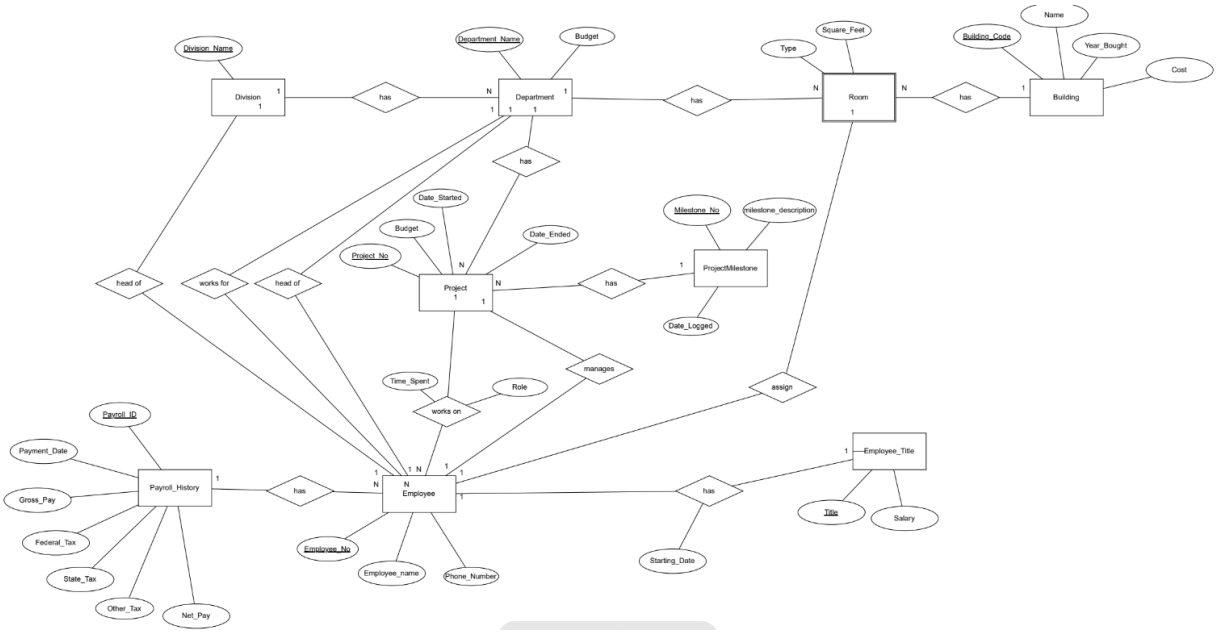
CS631 Final Project Report

Nabid Kabir
Wendy Spooner

Introduction

The goal of this project was to simulate working with real life databases that would be used in higher level corporate management. In order to do this we created a simplified form of this database that mimicked a wide variety of information that companies would need to keep track of. Some of these include data pertaining to: divisions, departments, employees, and projects. All of these attributes are related to each other in some fashion, which requires somewhat database management skills to keep track of it all. We also created web applications to manage and perform CRUD operations on so that we visualize and update our database in real time.

Database



Above is the updated ERD diagram that showcases all entities and relationships between entities. In addition to the entities laid out in the project description and given to us from assignment 4, we added PayrollHistory and ProjectMilestone tables. This is so we can identify payments and project milestones signified by unique identifiers and display them on the application. This will

Here is our step-by-step mapping of the database.

Step 1: Handling Entities

Division (Division_Name)
Department (Department_Name, Budget)
Building (Building_Code, Name, Year_Bought, Cost)
Project (Project_Number, Budget, Date_Started, Date_Ended)
Employee (Employee_No, Employee_Name, Phone_Number)

Employee_Title (Title, Salary)
PayrollHistory(Payroll_ID, Employee_No, Payment_Date, Gross_Pay, Federal_Tax, State_Tax, Other_Tax, Net_Pay)
ProjectMilestone(Milestone_No, Project_No, Milestone_Description, Date_Logged)

Step 2: Weak Entities

Room(Office_Number, Square Feet, Type)

Step 3: One to One Relationships

Division Head of Employee
Division (Division_Name, Employee_No)

Department Head of Employee
Department (Department_Name, Budget, Employee_No)

Employee Has Title
Employee (Employee_No, Employee_Name, , Phone_Number, Title)

Employee Assign Room
EmployeeRoom (Employee_No, Room_Number)

Step 4: One to N Relationships

Department Under Division
Department (Department_Name, Budget, Division_Name)

Employee Works for Department
Employee (Employee_No, Employee_Name, , Phone_Number, Title, Department_Name)

Building has Room
Room (Office_Number, Square_Feet, Type, Building_Code)

Department has Rooms
DepartmentRoom (Department_Name, Room_Number)

Employee Works on Project
EmployeeProject (Employee_No, Project_Number)

PayrollHistory can have one Employees and Employees can have many PayrollHistory
PayrollHistory(Payroll_ID, Employee_No, Payment_Date, Gross_Pay, Federal_Tax, State_Tax, Other_Tax, Net_Pay)

A project can have many milestones but a milestone corresponds to one project
ProjectMilestone(Milestone_No, Project_No, Milestone_Description, Date_Logged)

Step 5: M to N Relationships

No many to many relationships

Step 6: Multivalued Attributes

No multivalued attributes

Step 7: Higher Order Relationships

No higher order relationships

Step 8: Specialization

No specialization

Step 9: Aggregation

No Aggregation

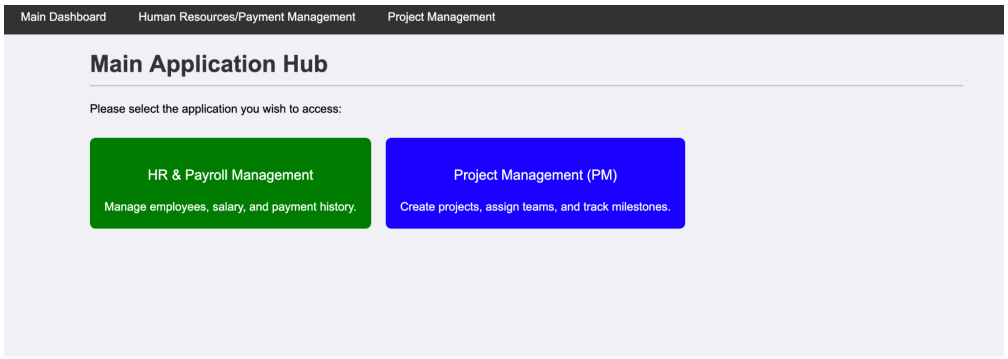
Relational Mapping

Division (Division_Name, Employee_No)
Department (Department_Name, Budget, Division_Name)
Building (Building_Code, Name, Year_Bought, Cost)
Project (Project_Number, Budget, Date_Started, Date_Ended)
Employee (Employee_No, Employee_Name, , Phone_Number, Title, Department_Name)
Employee_Title (Title, Salary)
Room (Office_Number, Square_Feet, Type, Building_Code)
EmployeeRoom (Employee_No, Room_Number)
DepartmentRoom (Department_Name, Room_Number)
EmployeeProject (Employee_No, Project_Number)
PayrollHistory(Payroll_ID, Employee_No, Payment_Date, Gross_Pay, Federal_Tax, State_Tax, Other_Tax, Net_Pay)
ProjectMilestone(Milestone_No, Project_No, Milestone_Description, Date_Logged)

Application Overview

Given the dual use case presented in the application requirements, we opted to create a user friendly application landing page. Here users can select the specific solution they wish to utilize. The landing page consists of a title bar that displays the main dashboard, human resource

application and project management solutions. Below the title bar lives the main application hub that hosts the applications/solutions (the words application and solution will be used interchangeably throughout this document). Users get an overview of the applications and their specific use cases and are able to navigate to their respective application via the application landing page.



Application Design

This aspect of the application was created in order to view the list of current employees at the company as well as manage them and administer payment. Once we had our database created in our local MySQL, we were able to connect it to our application. It is mainly coded in Python, using Flask as the web framework. We utilized SQLAlchemy as a toolkit to easily interact with the database and perform various operations on it. Here is the code to configure our connection to the database.

```python
app = Flask(__name__)

load_dotenv()

db_user = os.environ["DB_USER"]
db_pass = os.environ["DB_PASS"]
db_name = os.environ["DB_NAME"]
secret_key = os.environ["SECRET_KEY"]

if not all([db_user, db_pass, db_name, secret_key]):
    raise EnvironmentError("Missing required environment variables.")

app.config["SECRET_KEY"] = secret_key
app.config["SQLALCHEMY_DATABASE_URI"] = "mysql+mysqlconnector://{db_user}:{db_pass}@localhost/{db_name}".format(
    db_user = db_user, db_pass = db_pass, db_name = db_name
)
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False

db = SQLAlchemy(app)
```

Once we established the connection between our SQLAlchemy app and our locally stored database, we could then create the models that represent the tables in our database.

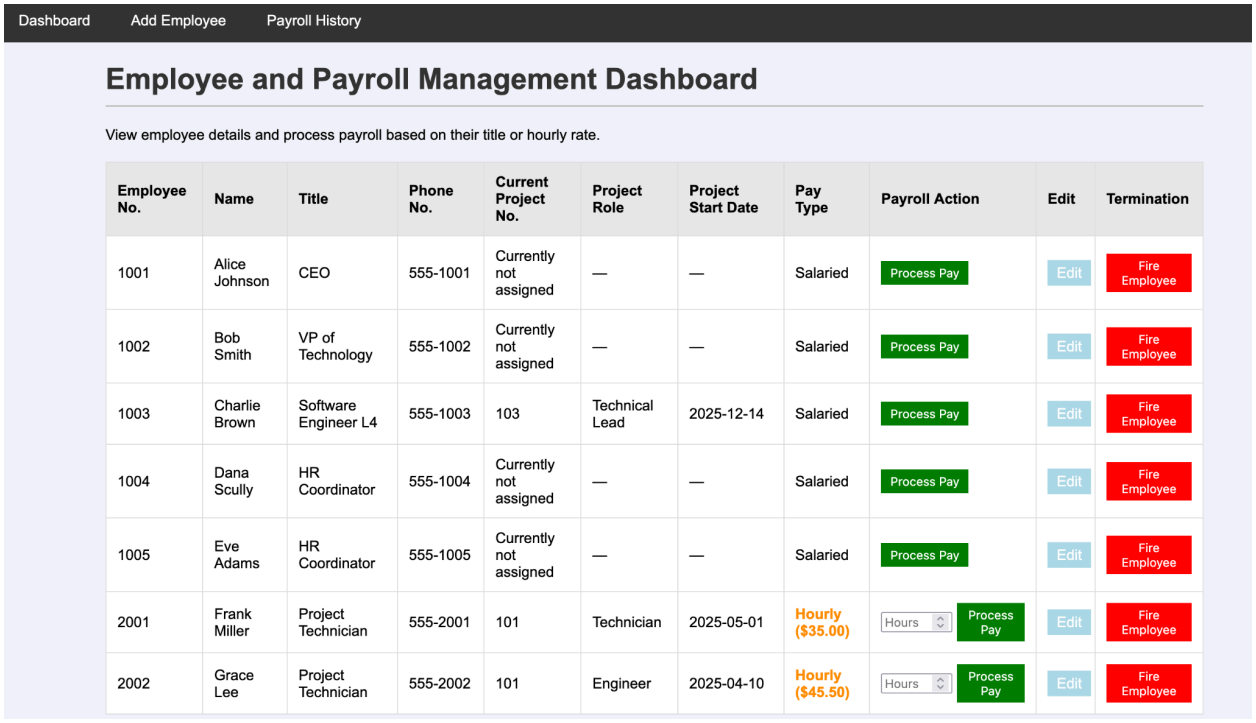
```python
class Division(db.Model):
    __tablename__ = 'Division'
    Division_Name = db.Column(db.String(100), primary_key=True)
    Head_Employee_No = db.Column(db.Integer, db.ForeignKey('Employee.Employee_No'), unique=True)
    departments = db.relationship('Department', backref='division', lazy=True)
    employees = db.relationship('Employee', foreign_keys='Employee.Division_Name', backref='direct_division', lazy=True)

class Employee(db.Model):
    __tablename__ = "Employee"
    Employee_No = db.Column(db.Integer, primary_key=True)
    Employee_Name = db.Column(db.String(200))
    Phone_Number = db.Column(db.String(20))
    Title = db.Column(db.String(100), db.ForeignKey('EmployeeTitle.Title'))

    Department_Name = db.Column(db.String(100), db.ForeignKey('Department.Department_Name'), nullable=True)
    Division_Name = db.Column(db.String(100), db.ForeignKey('Division.Division_Name'), nullable=True)

    Is_Hourly = db.Column(db.Boolean, default=False)
    Hourly_Rate = db.Column(db.Numeric(10, 2))

    projects_assigned = db.relationship(
        'EmployeeProject',
        back_populates='employee',
        lazy=True,
        primaryjoin="and_(Employee.Employee_No == EmployeeProject.Employee_No, EmployeeProject.Date_Ended == None)"
    )
    payroll_history = db.relationship('PayrollHistory', backref='employee', lazy=True)

    Is_Active = db.Column(db.Boolean, default=True)
```

We then created the models for the rest of our database as seen in screenshots below.

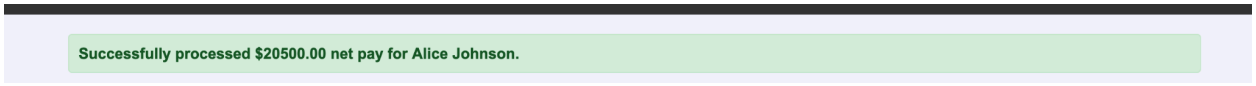
```python
class Department(db.Model):...

class Project(db.Model):...

class Building(db.Model):...

class Room(db.Model):...

class EmployeeTitle(db.Model):...

class EmployeeProject(db.Model):...

class PayrollHistory(db.Model):...
```

With our models created we were then tasked with creating the routes for the web pages that our application would navigate to in order to perform various tasks. The code for this will be provided. The first route is the main dashboard that showcases all the current employees in the company as well as further information on them.

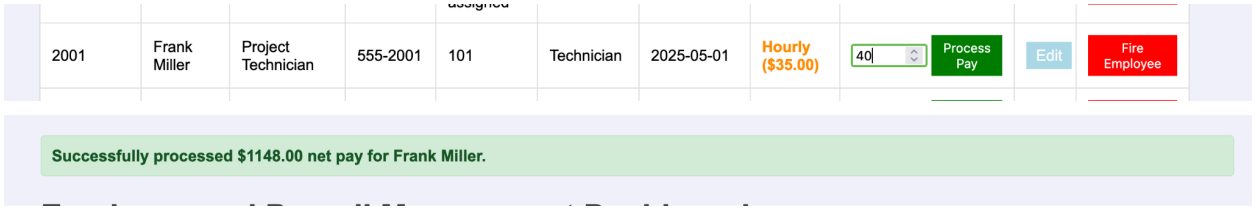
Human Resources/Payment Dashboard



As you can see, the top bar is used to quickly navigate to our three main pages, which are the dashboard, a page to add employees, and a page to view payroll history. On our dashboard we have information on every employee in the company. This information includes their employee number, name, title, phone number, current project, the role in their current project, the date they were assigned to the project, as well as various actions to update our database. We also display information on whether or not they are salaried or hourly employees brought on for a specific project. This is important information for the next column which is the functionality to process payment for individual employees. Salaried employees have a set dollar amount that gets deposited to their account every month. This amount is preset when the employee is created. For hourly employees, the amount per hour is preset, and the amount of hours they should be compensated for must be inputted.



We can see that when pressing the process pay button for a salaried employee like Alice Johnson, her monthly salary is deposited.



Here, we can also see that hourly employees can be paid based on any amount of hours that they have worked. This is showcasing a weekly salary for the employee Frank Miller, which is the standard of 40 hours a week.

We can then track these payment records as well as the taxes that must be paid in our payroll history page.

Payroll History

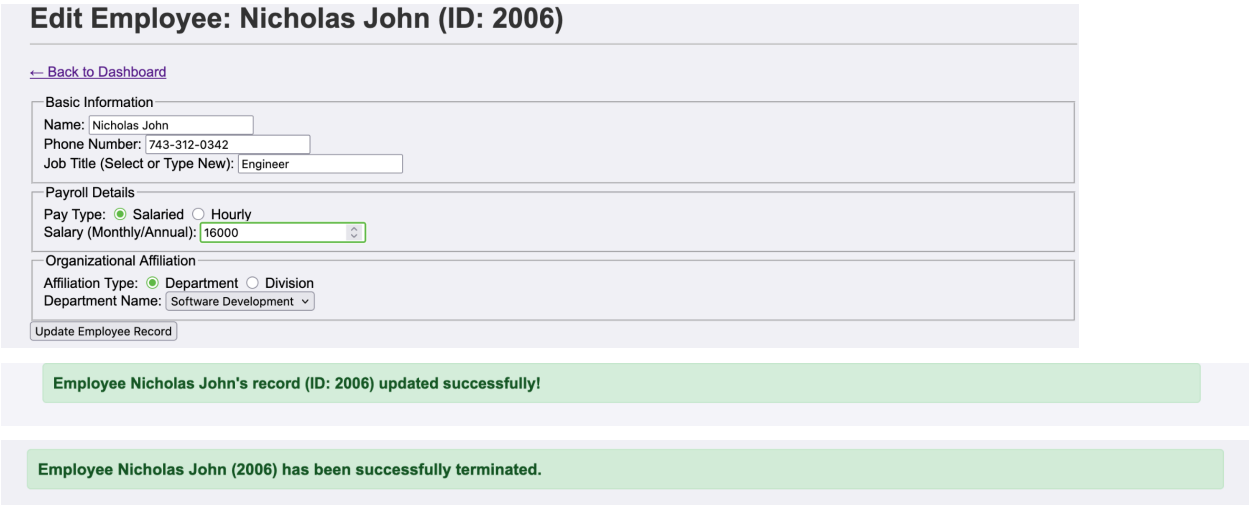
← Back to Employee Dashboard

Date	Employee No.	Employee Name	Gross Pay	Federal Tax	State Tax	Other Deductions	Net Pay
2025-12-14	1004	Dana Scully	$6000.00	$600.00	$300.00	$180.00	**$4920.00**
2025-12-14	1001	Alice Johnson	$25000.00	$2500.00	$1250.00	$750.00	**$20500.00**
2025-12-14	2004	Jane Mary	$2400.00	$240.00	$120.00	$72.00	**$1968.00**
2025-12-14	2003	Dale Earnheardt	$7500.00	$750.00	$375.00	$225.00	**$6150.00**
2025-12-14	1001	Alice Johnson	$25000.00	$2500.00	$1250.00	$750.00	**$20500.00**
2025-12-14	2001	Frank Miller	$1400.00	$140.00	$70.00	$42.00	**$1148.00**
2025-12-14	1004	Dana Scully	$6000.00	$600.00	$300.00	$180.00	**$4920.00**
2025-12-14	1001	Alice Johnson	$25000.00	$2500.00	$1250.00	$750.00	**$20500.00**
2025-12-14	1002	Bob Smith	$18000.00	$1800.00	$900.00	$540.00	**$14760.00**
2025-12-14	2001	Frank Miller	$2800.00	$280.00	$140.00	$84.00	**$2296.00**
2025-12-14	2005	John	$6000.00	$600.00	$300.00	$180.00	**$4920.00**
2025-12-14	2005	John	$6000.00	$600.00	$300.00	$180.00	**$4920.00**
2025-12-12	2001	Frank Miller	$1400.00	$140.00	$70.00	$42.00	**$1148.00**
2025-12-12	2001	Frank Miller	$1400.00	$140.00	$70.00	$42.00	**$1148.00**

We also have functionality to add employees to our database as well. This is easily accessed by pressing the add employee redirect button in the top bar. Once pressed, it brings you to this page.

On this page we have multiple inputs that must be filled in so that we can add all the right information for the new employee in our database. The inputs for various

After an employee has been added, if there are any changes to be recorded to the employee data, the users are able to edit employee information. For example, in the illustration below, Nicholas John has opted to leave the organization for a new opportunity. As a result, Nicholas needs to be removed from the active list of employees and see his employee information has been successfully updated to terminated to match his current employment status.



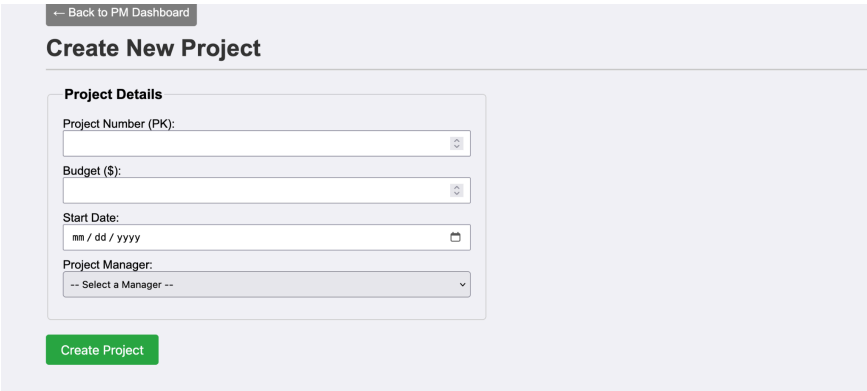
Project Management Interface

This aspect of the application was designed to create projects, associate or assign project manager/team members, view project statistics (number of employees assigned to a project and total number of hours worked per project). In addition to the aforementioned statistics, the application also tracks project milestones, providing information on status of blocks of work which are categorized as tasks. The application associates a status with each assigned task, tracking progress of task through to completion.

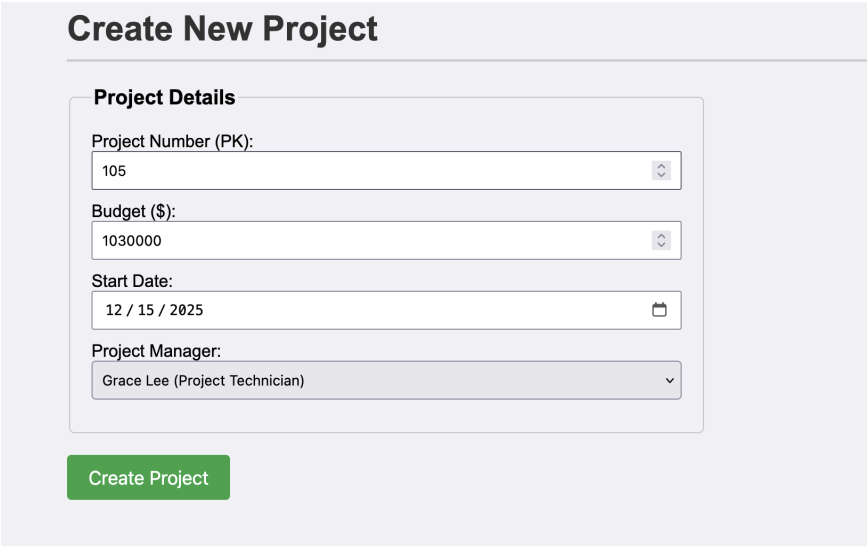
Given that the Project Management solution sits on the same infrastructure as the HR/Pay Roll solution, the construction is the same. The solution is connected to our local MySQL database with a web framework provided by Flask and automation driven by Python. We utilized SQLAlchemy as a toolkit to drive interaction between our application and database to carry out the specified CRUD functions as outlined in the requirements for the development of the application. Below is a screenshot of the code configuration driving the application:

Given that both solutions sit on the same system. We created a landing page that allows users to select the solution that they would like to navigate to. Once selection occurs, there is an authentication process which is necessary to initiate a connection between the app and the database. Please see screenshot below:

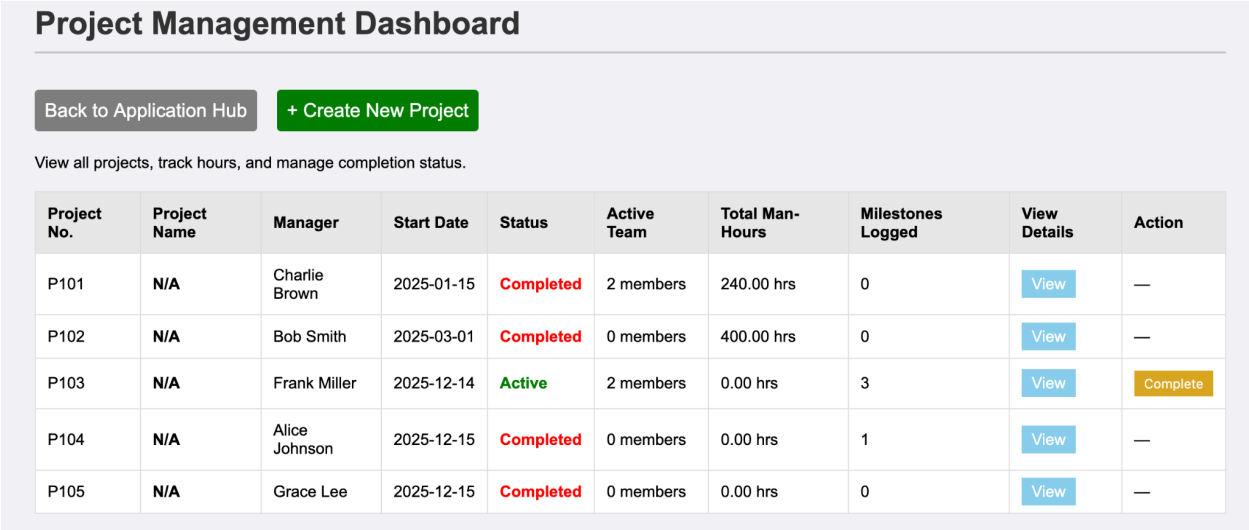
Similar to the HR/Payroll solution, our project management solution has a user friendly input form. That allows users to input data related to specified requirements/application use case into form. An example of this can be seen in screenshot below:



Once data has been added to the form, users can commit the data to the database by clicking on the “create project” button. Please see screenshot below:

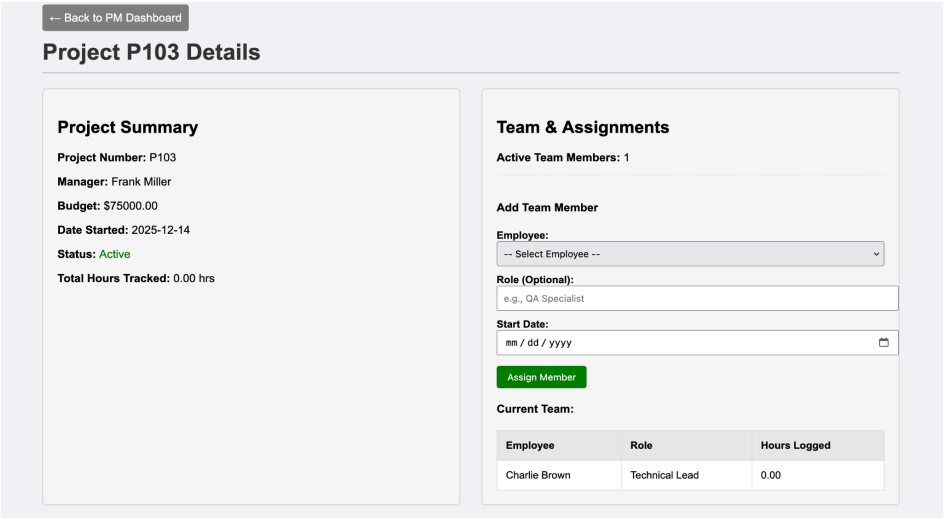


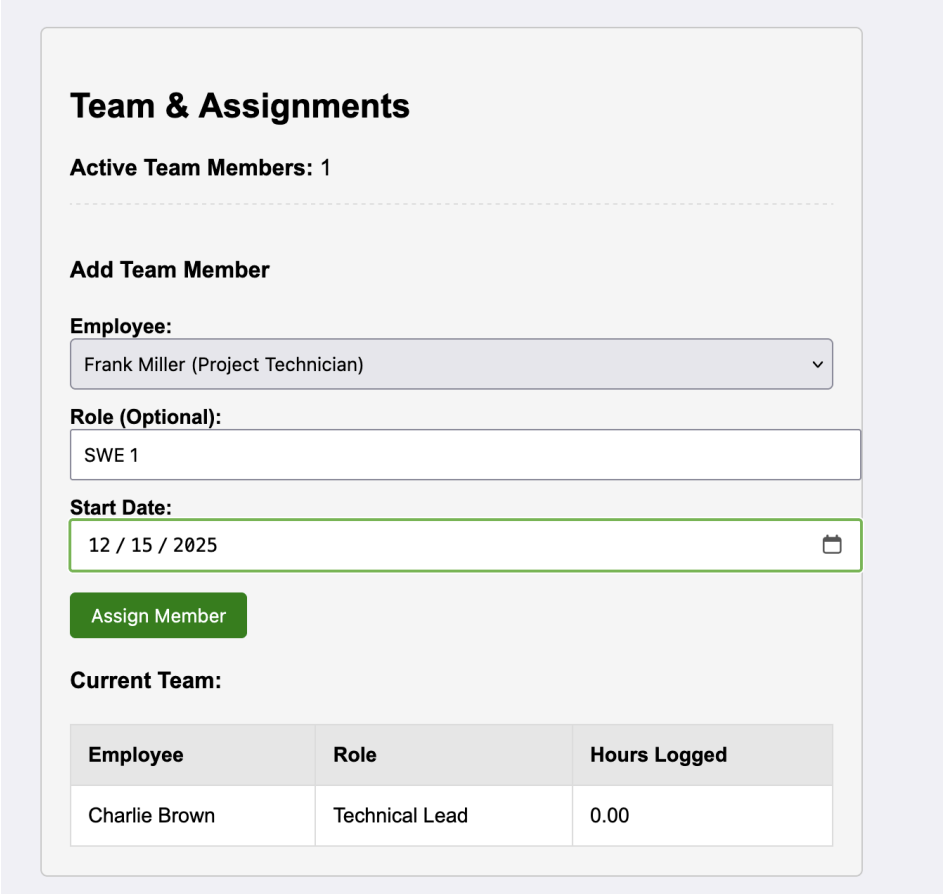
Once information has been submitted, the data is stored in a table format. Please see illustration below:



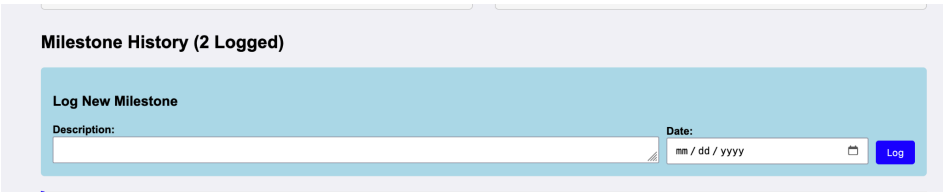
Consistent with the application requirements, relevant project information is being tracked. Each project is associated with a unique identifier/project number. The assigned project manager is tracked. Certain data points relevant to the project are displayed here such as: total number of hours tracking on each project as well as project status. It is easy to differentiate between projects that are completed vs active as well as the project milestones currently logged on each project.

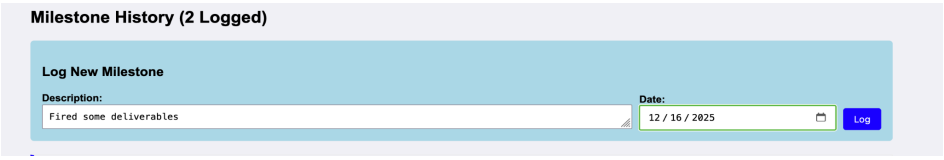
In a bid to get more granular as it relates to project requirements, the solution provides the ability to assign team members to a project. In the illustration below, Frank Miller has been assigned as the project manager for P103. Frank has a budget of $75,000 and he believes that adding a technical lead will add more value to the project and so Frank can opt to assign Charlie Brown as the technical lead. Frank can continue to add additional employees as needed. This is depicted in the illustrations below. Note, that employee field is a picklist data type, this is due to the fact that Frank can select a team member from the employee database. The metrics collected on team members are aligned to requirements which includes the number of hours logged to each project. Note, the number of active team members associated with a project is also being tracked and displayed.



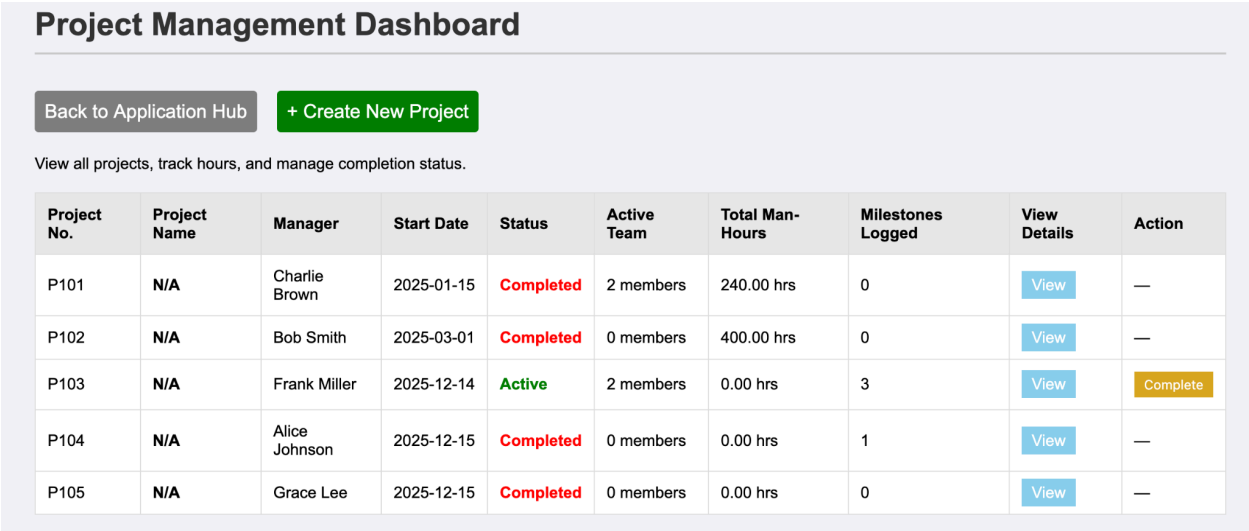


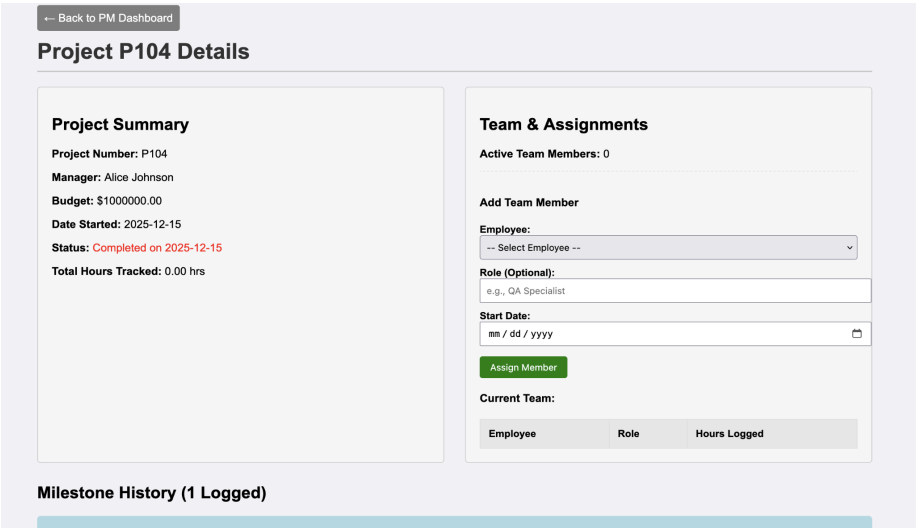
The solution also includes the functionality to add milestones. The task associated with the milestone is listed in the description files and once the relevant information associated with the milestone has been inputted and committed to the database that information is displayed in the table as well. Please see screenshot below:





Additionally, from the project data table or list view, users are able to create new projects or navigate back to the application landing page/application hub. A user can also view additional details related to a project by clicking on the “view” button under the view details column. The view details button open an interface that contains relevant project data such as status, assigned resource, hours logged, budget etc.





Reflection

Throughout this project, we ran into many problems when it came to setup as well as trying to get our query code to work properly. With little experience in coding web applications in Flask, there was an initial learning curve when it came to creating models and app routes. However, once we understood how routing worked with HTML files, it was relatively smooth sailing from there. Creating the HTML files was complex as well since we used Jinja2 to make the templating look pretty.

Overall, we learned a great deal about creating a functional full stack application using Flask and MySQL. Using the knowledge gained over the course of the course, we were able to create a database and populate it with fake data using relational arguments. One of the most fulfilling parts of this project was being able to showcase this ability by connecting the database to a Flask application and performing CRUD operations to interact with this database.