

PROJECT MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

**PRIYANKA PATTNAIK
DHRUBA JYOTI GHOSH
SHUBRAT SAHA**

in partial fulfillment for the award of the degree of

**BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE**



DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CENTURION INSTITUTE OF TECHNOLOGY

BHUBANESWAR

CENTURION UNIVERSITY OF TECHNOLOGY&MANAGEMENT::ODISHA

SEPTEMBER 2019

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
CENTURION UNIVERSITY OF TECHNOLOGY & MANAGEMENT
BHUBANESWAR -752050**

BONAFIDE CERTIFICATE

Certified that this project report “*Project Management System*” is the bonafide work of “**Priyanka Pattnaik, Dhruba Jyoti Ghosh & Shubrat Saha**” who carried out the project work under my supervision. This is to further certify to the best of my knowledge, that this project has not been carried out earlier in this institute and the university.

SIGNATURE

Prof. Manoj Kumar Behera

Asst. Professor – Department of Computer Science

Certified that the above-mentioned project has been duly carried out as per the norms of the college and statutes of the university

SIGNATURE

**Dr. Sujata Chakravarty
HEAD OF THE DEPARTMENT**

COMPUTER SCIENCE ENGINEERING

DEPARTMENT SEAL

ACKNOWLEDGEMENTS

We wish to express our profound and sincere gratitude to Prof. Manoj Kumar Behera, Department of Computer Science Engineering, CUTM Bhubaneswar, who guided us into the intricacies of this project non-chalantly with matchless magnanimity.

We thank Dr. Sujata Chakravarty, Head of the Dept. of Computer Science Engineering, CUTM Bhubaneswar and Dr. Prasanta Kumar Mohanty, DEAN, SOET CUTM for extending their support during Course of this investigation.

**PRIYANKA PATTNAIK
DHRUBA JYOTI GHOSH
SHUBRAT SAHA**

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1. CHAPTER – 1 INTRODUCTION		1 - 3
1.1. LAMP		
1.1.1 Why use Linux?		
1.1.2 Why use Apache?		
1.1.3 Why use PHP?		
1.1.4 Why use MySQL?		
1.2. AWS Cloud as a platform		
1.2.1 Amazon EC2		
1.3. Why use REST APIs?		
2. CHAPTER – 2 IDENTIFICATION & SOLUTION MAPPING OF THE PROBLEM STATEMENT		4 - 8
2.1. Solution Overview		
2.2. Tools and Technologies Overview		
2.2.1. PHP		
2.2.2. MySQL		
2.2.3. Apache		
2.2.4. REST APIs		
2.2.5. Amazon EC2		
3. CHAPTER – 3 PROJECT DEVELOPMENT		9 - 34
3.1. Application Background		
3.2. Database Designs & Data Model		
3.2.1 Table Classifications – archived_projects		
3.2.2 Table Classifications – company		
3.2.3 Table Classifications – completed_projects		
3.2.4 Table Classifications – department		
3.2.5 Table Classifications – employee		
3.2.6 Table Classifications – ongoing_projects		
3.3. ER Diagram		
3.4. Proof of Work		
3.4.1 Structure of database		
3.4.2 Structure of tables		
3.4.3 Demonstration of APIs		
3.4.4 Data in Databases		
4. CHAPTER – 4 FURTHER STUDY		35
4.1. Summary		
4.2. Project Scalability		

1. CHAPTER – I INTRODUCTION

The project is named as Project Management System which can provide backend service in managing data for organizations required as a part of client delivery, development cycles and workflows for developing a project. The project is a work of PHP and MySQL which is used to load and maintain databases at the backend and connect with the frontend for delivering contents and accepting requests.



1.1 LAMP

LAMP is an open source Web development platform that uses Linux as the operating system, Apache as the Web server, MySQL as the relational database management system and PHP as the object-oriented scripting language.

1.1.1 Why use Linux?

Linux makes very efficient use of the system's resources. Linux runs on a range of hardware, right from supercomputers to watches. You can give new life to your old and slow Windows system by installing a lightweight Linux system, or even run a NAS or media streamer using a particular distribution of Linux.

1.1.2 Why use Apache?

Apache is an open-source and free web server software that powers around 46% of websites around the world. The official name is Apache HTTP Server, and it's maintained and developed by the Apache Software Foundation. It allows website owners to serve content on the web — hence the name “web server”.

1.1.3 Why use PHP?

PHP is generally used in connecting a database in the backend to display and deliver contents in the front end. It is also a push which acts as an intermediate action performer to collect data in the front end and update the databases behind the curtains.

1.1.4 Why use MySQL?

MySQL is an opensource software under the terms of General Public License, and is also available under a variety of proprietary licenses. It is a relational database management system based on Structured Query Language (SQL). Although it can be used in a wide range of applications, MySQL is most often associated with web applications and online publishing. Using MySQL, enterprises enjoy significant cost savings on projects. The dependability and ease of management that accompany MySQL save troubleshooting time which is otherwise wasted in fixing downtime issues and performance problems.

1.2 AWS Cloud as a platform

The system is a prototype designed in a minute scenario but could be scaled at any level depending upon the business requirement. Generally, hosting such systems in cloud platform can reduce storage, cost and even helps in regular backup of data at any level.



1.2.1 Amazon EC2

Amazon Elastic Compute Cloud forms a central part of Amazon.com's cloud-computing platform, Amazon Web Services, by allowing users to rent virtual computers on which to run their own computer applications.

1.3 Why use REST APIs?

An API, or application programming interface, is essentially a way for apps to borrow functionality and data from each other. REST stands for Representational State Transfer. It is basically an architectural style that defines rules (constraints and properties) for computers to communicate over the Internet.

A REST API is a set of rules that developers follow when they create their API. One of these rules states that we should be able to get a piece of data (called a resource) when we link to a specific URL. Each URL is called a request while the data sent back to us is called a response.

A REST API defines a set of functions which developers can perform requests and receive responses via HTTP protocol such as GET and POST. The World Wide Web (WWW) is an example of a distributed system that uses REST protocol architecture to provide a hypermedia driven interface for websites.

2. CHAPTER – 2 IDENTIFICATION & SOLUTION MAPPING OF THE PROBLEM STATEMENT

The system challenges to solve complexities in the space of handling client delivery, insights generation and workflow integrations with the base platforms as PHP, MySQL and REST APIs.

It is very necessary to follow Agility in the development of projects which requires multiple breakups and transparency at every level. The project aims in lowering the latency of the complexity for the development engineers and project managers and focus in their core areas.

These can further help in upgrading the project qualities and standards without the interference of human actions and can hence meet the deadlines, transparency channels at every stage and upgrade the historic scenes at every levels of development.

2.1 Solution Overview

Maintain and scale databases with project details in a relational format to monitor the status and workflows of the projects. The system acts as an assistant to service companies providing technical services to clients from scratch or renovating phases.

The backend database is primarily segmented as archives to store projects which have been recognized during the run, ongoing to store projects which are in process of development and completed projects which are already completed and delivered to clients. These are actioned by the employees of the organization who are given the allowance to work on single ongoing projects at a time.

2.2 Tools & Technologies Overview

- 2.2.1 **PHP:** In this project, PHP is used as a connector to connect the database with the front end hence to ease the process of pushing the data into the database and process requests in pulling data out of databases. The multiple tables involved in the projects are archived_projects, company, completed_projects, department, employee and ongoing_projects, which acts independently in storing segmented data. The tables are connected with each other through keys ranging upon the actions and behavior of the data stored.



Benefits of using PHP

- OPEN SOURCE
- EXTENDIBLE
- PLATFORM INDEPENDENT
- LOW DEVELOPMENT AND MAINTENANCE COST
- COMPATIBILITY
- LARGE NUMBER OF DATABASES SUPPORTED

- 2.2.2 **MySQL:** MySQL is used in the project as an SQL server to simplify and generate data as per client requirement out of SQL queries, pushed into the system. It generally supports databases in the relational form and maintains levels of normal forms.



It is mostly used in applications because of the cost savings, dependability and ease of management that saves troubleshooting time which is otherwise wasted in fixing downtime issues.

Benefits of using MySQL

- DATA SECURITY
- ON DEMAND SCALABILITY
- HIGH PERFORMANCE
- ROUND-THE CLOCK UPTIME
- COMPREHENSIVE TRANSACTIONAL SUPPORT
- COMPLETE WORKFLOW CONTROL
- REDUCED TOTAL COST OF OWNERSHIP
- THE FLEXIBILITY OF OPEN SOURCE

- 2.2.3 **Apache:** Apache is an open-source and free web server software that powers around 46% of websites around the world. The official name is Apache HTTP Server, and it's maintained and developed by the Apache Software Foundation. It allows website owners to serve content on the web — hence the name “web server”. The Apache HTTP Server Project is an effort to develop and maintain an open-source HTTP server for modern operating systems including UNIX and Windows. The goal of this project is to provide a secure, efficient and extensible server that provides HTTP services in sync with the current HTTP standards. The Apache HTTP Server ("httpd") was launched in 1995 and it has been the most popular web server on the Internet since April 1996. It has celebrated its 20th birthday as a project in February 2015.



Benefits of using Apache

- OPEN SOURCE
- RELIABLE AND HIGH PERFORMANCE
- EASY TO INSTALL AND USE FULL-FEATURED WEB SERVER
- MORE SUPPORT AND FASTER TURNAROUND TIME
- IMMEDIATE CHANGES

2.2.4 **Linux:** Linux is the best-known and most-used open source operating system. As an operating system, Linux is software that sits underneath all of the other software on a computer, receiving requests from those programs and relaying these requests to the computer's hardware. Linux was created in 1991 by Linus Torvalds, a then-student at the University of Helsinki. Torvalds built Linux as a free and open source alternative to Minix, another Unix clone that was predominantly used in academic settings. He originally intended to name it "Freax," but the administrator of the server Torvalds used to distribute the original code named his directory "Linux" after a combination of Torvalds' first name and the word Unix, and the name stuck.

Benefits of using Linux

- OPEN SOURCE
- SECURITY
- STABILITY
- PRIVACY
- PERFORMANCE
- FLEXIBILITY



2.2.5 **REST APIs:** An application program interface (API) is a set of routines, protocols, and tools for building software applications. Basically, an API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components. APIs are designed for consumption for specific audiences (e.g., mobile developers), they are documented, and they are versioned in a way that users can have certain expectations of its maintenance and lifecycle. REST API is a set of rules that developers follow when they create their API. One of these rules states that we should be able to get a piece of data (called a resource) when we link to a specific URL. Each URL is called a request while the data sent back to you is called a response. It is called REST, because the client initiates transfer of representations of client state. Representational State Transfer (REST) is a style of software architecture for distributed systems such as the World Wide Web. REST has emerged as a predominant Web service design model.

Benefits of using REST APIs

- SEPARATION BETWEEN THE CLIENT AND THE SERVER
- VISIBILITY, RELIABILITY AND SCALABILITY
- INDEPENDENT OF THE TYPE OF PLATFORM OR LANGUAGE

2.2.6 **Amazon EC2:** Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment. Amazon EC2 reduces the time required to obtain and boot new server instances to minutes, allowing you to quickly scale capacity, both up and down, as your computing requirements change. Amazon EC2 changes the economics of computing by allowing you to pay only for capacity that you actually use. Amazon EC2 provides developers the tools to build failure resilient applications and isolate them from common failure scenarios.

Benefits of using EC2

- ELASTIC WEB-SCALE COMPUTING
- COMPLETELY CONTROLLED
- FLEXIBLE CLOUD HOSTING SERVICES
- INTEGRATED
- RELIABLE
- SECURE
- INEXPENSIVE
- EASY TO START



3.1 Application Background

The application is developed, thinking of developing a platform on cloud which can ease the work of professionals at every level of management and development, maintaining a cent percent transparency and accuracy at each stage of progress made. The system is designed to handle bulk data of projects in three major classifications namely archived, ongoing and completed projects giving allowance to employees to work for a single project. The identities are maintained using keys in every tables and ease the process by the use of the technologies mentioned. The application acts as a monitoring tool which can visualize and generate report of the overall operations and project management status hence reducing the efforts of maintaining manual databases and transparency at every level of transactions.

The major focus also is put to make a fresh and dynamic backend with a high rate of scalability in the enterprise environment in the cloud platform. Being in the prototyping stage the Project Management System application is a fully functional and responsive data management system.

3.2 Database Designs & Data Model

The database is designed looking to the dynamic requirements by the solution providers and preservation of data for future analytics and decision-making processes.

The overall database is uniformly categorized as archived_projects, company, completed_projects, department, employee and ongoing_projects. The various tables are inter-connected using keys depending upon the size and nature of data stored.

Table Classifications

3.2.1 ***archived_projects***: This table holds relational data, having performed good or recognized projects during the development and maintenance run.

archived_projects

Column	Type	Null	Default	Links to	Comments	MIME
<i>id (Primary)</i>	int(11)	No				
name	varchar(50)	No				
start_date	date	No				
completion_date	date	No				
achievement	varchar(30)	No				
cid	int(11)	No		company -> id		
did	int(11)	No		department -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	<i>id</i>	0	A	No	
cid	BTREE	No	No	<i>cid</i>	0	A	No	
did	BTREE	No	No	<i>did</i>	0	A	No	

- 3.2.2 ***company***: The table holds relational data, having details of clients.

company

Column	Type	Null	Default	Links to	Comments	MIME
<i>id (Primary)</i>	int(11)	No				
name	varchar(30)	No				
email	varchar(50)	No				
address	varchar(200)	No				
phone	bigint(20)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	<i>id</i>	1	A	No	
email	BTREE	Yes	No	email	1	A	No	
phone	BTREE	Yes	No	phone	1	A	No	

- 3.2.3 ***completed_projects***: This table holds relational data, having projects delivered post development and even projects under maintenance.

completed_projects

Column	Type	Null	Default	Links to	Comments	MIME
<i>id (Primary)</i>	int(11)	No				
name	varchar(50)	No				
start_date	date	No				
completion_date	date	No				
<i>cid</i>	int(11)	No		company -> id		
<i>did</i>	int(11)	No		department -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	<i>id</i>	0	A	No	
<i>cid</i>	BTREE	No	No	<i>cid</i>	0	A	No	
<i>did</i>	BTREE	No	No	<i>did</i>	0	A	No	

- 3.2.4 ***department***: This table records the department performance and overall description of every department at the solution provider's end.

department

Column	Type	Null	Default	Links to	Comments	MIME
<i>id (Primary)</i>	int(11)	No				
name	varchar(30)	No				
strength	int(11)	No				
email	varchar(50)	No				

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	<i>id</i>	0	A	No	
email	BTREE	Yes	No	email	0	A	No	

- 3.2.5 ***employee***: This table records the department employee's records and performance and overall details of every employee working for projects at the solution provider's end.

employee

Column	Type	Null	Default	Links to	Comments	MIME
<i>id (Primary)</i>	int(11)	No				
name	varchar(50)	No				
email	varchar(50)	No				
phone	bigint(20)	No				
job_title	varchar(20)	No				
pid	int(11)	No		ongoing_projects -> id		
did	int(11)	No		department -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	<i>id</i>	0	A	No	
email	BTREE	Yes	No	email	0	A	No	
phone	BTREE	Yes	No	phone	0	A	No	
did	BTREE	No	No	<i>did</i>	0	A	No	
pid	BTREE	No	No	pid	0	A	No	

- 3.2.6 ***ongoing_projects***: The table holds relational data, having details of client projects that are in the process of development or yet to deliver.

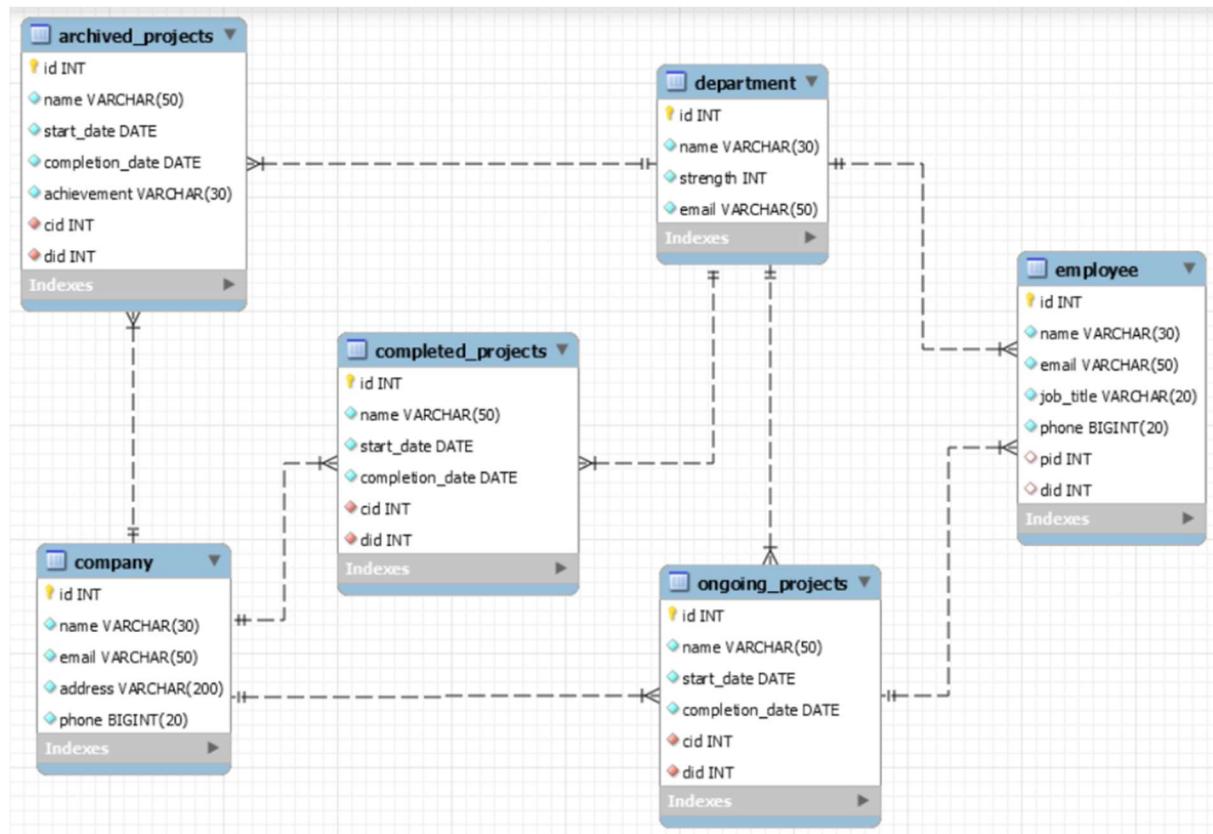
ongoing_projects

Column	Type	Null	Default	Links to	Comments	MIME
<i>id (Primary)</i>	int(11)	No				
name	varchar(50)	No				
start_date	date	No				
deadline	date	No				
cid	int(11)	No		company -> id		
did	int(11)	No		department -> id		

Indexes

Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	<i>id</i>	0	A	No	
cid	BTREE	No	No	cid	0	A	No	
did	BTREE	No	No	<i>did</i>	0	A	No	

3.3 ER Diagram



3.4 Proof of Works

3.4.1 Structure of database

The screenshot shows the phpMyAdmin interface for the 'project_mgmt_system' database. The left sidebar lists databases: New, information_schema, mysql, performance_schema, and project_mgmt_system. The 'New' database is selected. The main area displays the following tables:

Table	Action	Rows	Type	Collation	Size	Overhead
archived_projects	Browse Structure Search Insert Empty Drop	48	InnoDB	latin1_swedish_ci	48 Kib	-
company	Browse Structure Search Insert Empty Drop	48	InnoDB	latin1_swedish_ci	48 Kib	-
completed_projects	Browse Structure Search Insert Empty Drop	48	InnoDB	latin1_swedish_ci	48 Kib	-
department	Browse Structure Search Insert Empty Drop	32	InnoDB	latin1_swedish_ci	32 Kib	-
employee	Browse Structure Search Insert Empty Drop	80	InnoDB	latin1_swedish_ci	80 Kib	-
ongoing_projects	Browse Structure Search Insert Empty Drop	48	InnoDB	latin1_swedish_ci	48 Kib	-
Sum		384			384 Kib	0 B

Below the table list, there is a 'Create table' form with 'Name:' set to 'New' and 'Number of columns:' set to '4'. A 'Go' button is present. At the bottom right, there is an 'Activate Windows' message: 'Activate Windows Go to Settings to activate Windows.'

3.4.2 Structure of tables

The screenshot shows the phpMyAdmin interface for the 'archived_projects' table within the 'project_mgmt_system' database. The left sidebar shows the database structure. The main area displays the table structure with the following columns:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)	latin1_swedish_ci		No	None	AUTO_INCREMENT		Change Drop More
2	name	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
3	start_date	date			No	None			Change Drop More
4	completion_date	date			No	None			Change Drop More
5	achievement	varchar(30)	latin1_swedish_ci		No	None			Change Drop More
6	cid	int(11)			No	None			Change Drop More
7	did	int(11)			No	None			Change Drop More

Below the table structure, there is a 'Indexes' section with two entries:

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Drop	PRIMARY	BTREE	Yes	No	id	0	A	No	
Drop	cid	BTREE	No	No	cid	0	A	No	
Drop	did	BTREE	No	No	did	0	A	No	

At the bottom, there is a 'Partitions' section with the message: 'No partitioning defined!'. An 'Activate Windows' message is also present at the bottom right.

phpMyAdmin

Server: localhost | Database: project_mgmt_system | Table: company

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)	latin1_swedish_ci	No	None		AUTO_INCREMENT		Change Drop More
2	name	varchar(30)	latin1_swedish_ci	No	None				Change Drop More
3	email	varchar(50)	latin1_swedish_ci	No	None				Change Drop More
4	address	varchar(200)	latin1_swedish_ci	No	None				Change Drop More
5	phone	bigint(20)		No	None				Change Drop More

Add 1 column(s) after phone Go

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	id	2	A	No	
Edit Drop	email	BTREE	Yes	No	email	2	A	No	
Edit Drop	phone	BTREE	Yes	No	phone	2	A	No	

Create an index on 1 columns Go

Partitions

No partitioning defined!

Activate Windows
Go to Settings to activate Windows Partition table

phpMyAdmin

Server: localhost | Database: project_mgmt_system | Table: completed_projects

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)	latin1_swedish_ci	No	None		AUTO_INCREMENT		Change Drop More
2	name	varchar(50)	latin1_swedish_ci	No	None				Change Drop More
3	start_date	date		No	None				Change Drop More
4	completion_date	date		No	None				Change Drop More
5	cld	int(11)		No	None				Change Drop More
6	did	int(11)		No	None				Change Drop More

Add 1 column(s) after did Go

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	id	0	A	No	
Edit Drop	cld	BTREE	No	No	cld	0	A	No	
Edit Drop	did	BTREE	No	No	did	0	A	No	

Create an index on 1 columns Go

Partitions

No partitioning defined!

Activate Windows
Go to Settings to activate Windows Partition table

phpMyAdmin

Server: localhost Database: project_mgmt_system Table: department

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	
2	name	varchar(30)	latin1_swedish_ci		No	None			
3	strength	int(11)			No	None			
4	email	varchar(50)	latin1_swedish_ci		No	None			

Check all With selected Browse Change Drop Primary Unique Index Fulltext

Print Propose table structure Move columns Normalize

Add 1 column(s) after email Go

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
	PRIMARY	BTREE	Yes	No	id	0	A	No	
	email	BTREE	Yes	No	email	0	A	No	

Create an index on 1 columns Go

Partitions

No partitioning defined!

Activate Windows Go to Settings to activate Windows.

phpMyAdmin

Server: localhost Database: project_mgmt_system Table: employee

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Table structure Relation view

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	
2	name	varchar(50)	latin1_swedish_ci		No	None			
3	email	varchar(50)	latin1_swedish_ci		No	None			
4	phone	bignit(20)			No	None			
5	job_title	varchar(20)	latin1_swedish_ci		No	None			
6	pid	int(11)			No	None			
7	did	int(11)			No	None			

Check all With selected Browse Change Drop Primary Unique Index Fulltext

Print Propose table structure Move columns Normalize

Add 1 column(s) after did Go

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
	PRIMARY	BTREE	Yes	No	id	0	A	No	
	email	BTREE	Yes	No	email	0	A	No	
	phone	BTREE	Yes	No	phone	0	A	No	
	did	BTREE	Yes	No	did	0	A	No	
	pid	BTREE	No	No	pid	0	A	No	

Create an index on 1 columns Go

Partitions

Activate Windows Go to Settings to activate Windows.

Table structure

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)	latin1_swedish_ci	No	None		AUTO_INCREMENT		Change Drop More
2	name	varchar(50)	latin1_swedish_ci	No	None				Change Drop More
3	start_date	date		No	None				Change Drop More
4	deadline	date		No	None				Change Drop More
5	cid	int(11)		No	None				Change Drop More
6	did	int(11)		No	None				Change Drop More

Indexes

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Drop	PRIMARY	BTREE	Yes	No	id	0	A	No	
Edit Drop	cid	BTREE	No	No	cid	0	A	No	
Edit Drop	did	BTREE	No	No	did	0	A	No	

Create an index on columns [Go](#)

Partitions

No partitioning defined

Activate Windows
Go to Settings to activate Windows.

Partition table

3.4.3 Demonstration of APIs

POST <http://123.136.253> [Send](#) [Save](#)

Body

```
{
  "name": "EY",
  "address": "New Delhi",
  "phone": 1166718000,
  "email": "contactus@in.ey.com"
}
```

Status: 200 OK Time: 781 ms

Activate Windows
Go to Settings to activate Windows.

The screenshot shows the Postman application interface. At the top, there are two tabs: 'http://13.233.136.253' and 'http://13.233.136.253'. The main header bar includes 'No Environment' and various settings icons. Below the header, the request details are shown: 'POST' method, URL 'http://13.233.136.253/rest_pms/api/company/update.php', and a 'Body' tab selected. The body content is a JSON object:

```
1 * {
2     "id": "1",
3     "name": "EY",
4     "address": "New Delhi",
5     "email": "contactus@in.ey.com",
6     "phone": "1166718000"
7 }
```

Below the body, the response section shows a status of '200 OK' and a time of '655 ms'. The response body is also JSON:

```
1 * {
2     "message": "Updated"
3 }
```

The screenshot shows the Postman application interface. At the top, there are two tabs: 'http://13.233.136.253' and 'http://13.233.136.253'. The main header bar includes 'No Environment' and various settings icons. Below the header, the request details are shown: 'GET' method, URL 'http://13.233.136.253/rest_pms/api/company/read.php', and an 'Authorization' tab selected. The body content is set to 'No Auth'. The response section shows a status of '200 OK' and a time of '720 ms'. The response body is JSON:

```
1 * [
2     {
3         "id": "1",
4         "name": "EY",
5         "address": "Kolkata",
6         "phone": "1166718000",
7         "email": "contactus@in.ey.com"
8     }
9 ]
```

The screenshot shows the Postman interface with the following details:

- URL:** http://13.233.136.253/rest_pms/api/company/read_single.php?id=1
- Status:** 200 OK
- Time:** 675 ms
- Body (Pretty JSON):**

```

1 * [
2   "id": "1",
3   "name": "EV",
4   "address": "Kolkata",
5   "email": "contactus@ey.com",
6   "phone": "1166718000"
7 ]
  
```

The screenshot shows the Postman interface with the following details:

- URL:** http://13.233.136.253/rest_pms/api/company/delete.php
- Status:** 200 OK
- Time:** 374 ms
- Body (JSON application/json):**

```

1 * {
2   "id": 2
3 }
  
```

- Response Body (Pretty JSON):**

```

1 * {
2   "message": "Deleted"
3 }
  
```

Filter

History Collections Clear all

Today

- POST** http://13.127.121.188/rest_pms/api/department/create.php
- POST** http://13.127.121.188/rest_pms/api/department/create.php
- POST** http://13.232.125.119/rest_pms/api/department/create.php

Yesterday

- POST** http://13.232.125.119/rest_pms/api/department/create.php
- GET** http://13.232.125.119/rest_pms/api/department/read.php
- GET** 13.232.125.119/rest_pms/api/company/read.php
- GET** http://13.232.125.119/rest_pms/api/company/read.php
- POST** http://13.233.136.253/rest_pms/api/company/update.php
- GET** http://13.233.136.253/rest_pms/api/company/read_single.php?id=1
- GET** http://13.233.136.253/rest_pms/api/company/read.php
- POST** http://13.233.136.253/rest_pms/api/company/delete.php
- POST** http://13.233.136.253/rest_pms/api/company/delete.php

Body Headers (11) Test Results Status: 200 OK Time: 712 ms

```
1 * [
2   "name": "Technical",
3   "strength": 0,
4   "email": "technial@pms.com"
5 ]
```

Pretty Raw Preview JSON

```
1 * [
2   "message": "Created"
3 ]
```

Activate Windows Go to Settings to activate Windows.

Filter

History Collections Clear all

Today

- POST** http://13.127.121.188/rest_pms/api/department/update.php
- POST** http://13.127.121.188/rest_pms/api/department/update.php
- POST** http://13.127.121.188/rest_pms/api/department/create.php
- POST** http://13.127.121.188/rest_pms/api/department/create.php
- POST** http://13.232.125.119/rest_pms/api/department/create.php

Yesterday

- POST** http://13.232.125.119/rest_pms/api/department/create.php
- GET** http://13.232.125.119/rest_pms/api/department/read.php
- GET** 13.232.125.119/rest_pms/api/company/read.php
- GET** http://13.232.125.119/rest_pms/api/company/read.php
- POST** http://13.233.136.253/rest_pms/api/company/update.php
- GET** http://13.233.136.253/rest_pms/api/company/read_single.php?id=1
- GET** http://13.233.136.253/rest_pms/api/company/read.php

Body Headers (11) Test Results Status: 200 OK Time: 302 ms

```
1 * [
2   "id": 3,
3   "name": "Technical",
4   "strength": 1,
5   "email": "tech@pms.com"
6 ]
```

Pretty Raw Preview JSON

```
1 * [
2   "message": "Updated"
3 ]
```

Activate Windows Go to Settings to activate Windows.

History Collections Clear all

GET http://13.127.121.188/rest_pms/api/department/read.php

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (11) Test Results

Pretty Raw Preview JSON

```

1: [
2:   {
3:     "id": "3",
4:     "name": "Technical",
5:     "strength": "1",
6:     "email": "tech@pms.com"
7:   }
8: ]

```

Status: 200 OK Time: 288 ms

Activate Windows
Go to Settings to activate Windows.

History Collections Clear all

GET http://13.127.121.188/rest_pms/api/department/read_single.php?id=3

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (11) Test Results

Pretty Raw Preview JSON

```

1: {
2:   "id": "3",
3:   "name": "Technical",
4:   "strength": "1",
5:   "email": "tech@pms.com"
6: }

```

Status: 200 OK Time: 304 ms

Activate Windows
Go to Settings to activate Windows.

The screenshot shows the Postman application interface. The left sidebar displays a history of API requests made today and yesterday. The main panel shows a POST request to `http://13.127.121.188/rest_pms/api/department/delete.php`. The request body is set to `JSON (application/json)` and contains the following JSON payload:

```

1 {
2   "id": 3
3 }

```

The response status is `200 OK` with a time of `587 ms`. The response body is:

```

1 {
2   "message": "Deleted"
3 }

```

A watermark for "Activate Windows" is visible in the bottom right corner.

The screenshot shows the Postman application interface. The left sidebar displays a history of API requests made today and yesterday. The main panel shows a POST request to `http://13.127.121.188/rest_pms/api/ongoing_projects/create.php`. The request body is set to `JSON (application/json)` and contains the following JSON payload:

```

1 {
2   "name": "Hospital Management System",
3   "start_date": "2019-09-01",
4   "deadline": "2019-09-10",
5   "cid": 1,
6   "did": 4
7 }

```

The response status is `200 OK` with a time of `315 ms`. The response body is:

```

1 {
2   "message": "Created"
3 }

```

A watermark for "Activate Windows" is visible in the bottom right corner.

The screenshot shows the Postman interface with a successful API call. The URL is `http://13.127.121.188/rest_pms/api/ongoing_projects/update.php`. The request method is POST. The Body tab shows JSON data:

```

1 * {
2     "id": 2,
3     "name": "Hospital Management System",
4     "start_date": "2019-09-01",
5     "deadline": "2019-09-14",
6     "cid": "1",
7     "did": 4
8 }

```

The response status is 200 OK with a time of 316 ms. The response body is:

```

1 * [
2     {
3         "message": "Updated"
4     }
5 ]

```

Activate Windows
Go to Settings to activate Windows.

The screenshot shows the Postman interface with a successful API call. The URL is `http://13.127.121.188/rest_pms/api/ongoing_projects/read.php`. The request method is GET. The Body tab shows JSON data:

```

1 * [
2     {
3         "id": 2,
4         "name": "Hospital Management System",
5         "start_date": "2019-09-01",
6         "deadline": "2019-09-14",
7         "cid": "1",
8         "did": 4
9     }
10 ]

```

The response status is 200 OK with a time of 337 ms. The response body is:

```

1 * [
2     {
3         "id": 2,
4         "name": "Hospital Management System",
5         "start_date": "2019-09-01",
6         "deadline": "2019-09-14",
7         "cid": "1",
8         "did": 4
9     }
10 ]

```

Activate Windows
Go to Settings to activate Windows.

The screenshot shows the Postman interface with the following details:

- URL:** http://13.127.121.188/rest_pms/api/ongoing_projects/read_single.php?id=2
- Method:** GET
- Body:** JSON response (Pretty, Raw, Preview, JSON)


```
1 = {
2   "id": "2",
3   "name": "Hospital Management System",
4   "start_date": "2019-09-01",
5   "deadline": "2019-09-14",
6   "cid": "1",
7   "did": "4"
8 }
```
- Status:** 200 OK, Time: 620 ms
- Authorization:** No Auth
- Headers:** (1)
- Tests:**
- Code:**

The screenshot shows the Postman interface with the following details:

- URL:** http://13.127.121.188/rest_pms/api/ongoing_projects/delete.php
- Method:** POST
- Body:** JSON (application/json) (Pretty, Raw, Preview, JSON)


```
1 = {
2   "id": 2
3 }
```
- Status:** 200 OK, Time: 586 ms
- Authorization:** No Auth
- Headers:** (1)
- Tests:**
- Code:**

The screenshot shows the Postman application interface. The left sidebar displays a history of API requests made today, including various GET and POST requests to URLs like /rest_pms/api/completed_projects/* and /ongoing_projects/*.

The main workspace shows a POST request to http://13.127.121.188/rest_pms/api/completed_projects/create.php. The request body is set to JSON (application/json) and contains the following data:

```

1 * {
2   "name": "Airlines Management System",
3   "start_date": "2019-01-01",
4   "completion_date": "2019-02-14",
5   "cid": 1,
6   "did": 4
7 }

```

The response status is 200 OK, and the message is "Created".

This screenshot shows another POST request to http://13.127.121.188/rest_pms/api/completed_projects/update.php. The request body is set to JSON (application/json) and contains the following data:

```

1 * {
2   "id": 1,
3   "name": "Airlines Management System",
4   "start_date": "2019-01-10",
5   "completion_date": "2019-02-14",
6   "cid": 1,
7   "did": 4
8 }

```

The response status is 200 OK, and the message is "Updated".

The screenshot shows the Postman application interface. The left sidebar displays a history of requests made today, including various GET and POST requests to the URL `http://13.127.121.188/rest_pms/api/completed_projects`. The main panel shows a specific request for reading completed projects. The request URL is `http://13.127.121.188/rest_pms/api/completed_projects/read.php`. The response status is 200 OK, and the response time is 297 ms. The response body is displayed in JSON format:

```

1 = [
2   {
3     "id": "1",
4     "name": "Airlines Management System",
5     "start_date": "2019-01-10",
6     "completion_date": "2019-02-14",
7     "cid": "1",
8     "did": "4"
9   }
10 ]

```

Activate Windows
Go to Settings to activate Windows.

The screenshot shows the Postman application interface. The left sidebar displays a history of requests made today, including various GET and POST requests to the URL `http://13.127.121.188/rest_pms/api/completed_projects`. The main panel shows a specific request for reading a single completed project by ID. The request URL is `http://13.127.121.188/rest_pms/api/completed_projects/read_single.php?id=1`. The response status is 200 OK, and the response time is 567 ms. The response body is displayed in JSON format:

```

1 = [
2   {
3     "id": "1",
4     "name": "Airlines Management System",
5     "start_date": "2019-01-10",
6     "completion_date": "2019-02-14",
7     "cid": "1",
8     "did": "4"
9   }
10 ]

```

Activate Windows
Go to Settings to activate Windows.

Postman interface showing a successful DELETE request to `http://13.127.121.188/rest_pms/api/completed_projects/delete.php` with ID 1. The response body is `{ "message": "Deleted" }`.

Postman interface showing a successful POST request to `http://13.127.121.188/rest_pms/api/archived_projects/create.php`. The request body contains project details: `{ "id": 1, "name": "Airlines Management System", "start_date": "2019-01-10", "completion_date": "2019-02-14", "achievement": "Project of the year", "xid": 1, "sid": 1 }`. The response body is `{ "message": "Created" }`.

The screenshot shows the Postman interface with a successful API call. The URL is `http://13.127.121.188/rest_pms/api/archived_projects/update.php`. The Body tab displays the JSON payload sent to the server:

```

1 - [
2   {
3     "id": 1,
4     "name": "Airlines Management System",
5     "start_date": "2019-01-01",
6     "completion_date": "2019-02-14",
7     "achievement": "Project of the year",
8     "cid": 1,
9     "did": 4
10   }
11 ]

```

The response status is 200 OK, and the message is "Updated".

The screenshot shows the Postman interface with a successful API call. The URL is `http://13.127.121.188/rest_pms/api/archived_projects/read.php`. The Body tab displays the JSON response received from the server:

```

1 - [
2   {
3     "id": 1,
4     "name": "Airlines Management System",
5     "start_date": "2019-01-01",
6     "completion_date": "2019-02-14",
7     "achievement": "Project of the year",
8     "cid": 1,
9     "did": 4
10   }
11 ]

```

The response status is 200 OK.

Filter

History Collections Clear all

Today

GET http://13.127.121.188/rest_pms/api/archived_projects/read_single.php?id=1

GET http://13.127.121.188/rest_pms/api/archived_projects/read.php

POST http://13.127.121.188/rest_pms/api/archived_projects/update.php

POST http://13.127.121.188/rest_pms/api/archived_projects/create.php

POST http://13.127.121.188/rest_pms/api/completed_projects/delete.php

GET http://13.127.121.188/rest_pms/api/completed_projects/read_single.php?id=1

GET http://13.127.121.188/rest_pms/api/completed_projects/read.php

POST http://13.127.121.188/rest_pms/api/completed_projects/update.php

POST http://13.127.121.188/rest_pms/api/completed_projects/create.php

POST http://13.127.121.188/rest_pms/api/ongoing_projects/create.php

POST http://13.127.121.188/rest_pms/api

No Environment

Send Save Code

Type No Auth

Body Cookies Headers (11) Test Results

Pretty Raw Preview JSON

```

1 - [
2   "id": "1",
3   "name": "Airlines Management System",
4   "start_date": "2019-01-01",
5   "completion_date": "2019-02-14",
6   "achievement": "Project of the year",
7   "x_id": "1",
8   "did": "4"
9 ]

```

Status: 200 OK Time: 305 ms

Activate Windows
Go to Settings to activate Windows.

Filter

History Collections Clear all

Today

PUT http://13.127.121.188/rest_pms/api/archived_projects/delete.php

GET http://13.127.121.188/rest_pms/api/archived_projects/read_single.php?id=1

GET http://13.127.121.188/rest_pms/api/archived_projects/read.php

POST http://13.127.121.188/rest_pms/api/archived_projects/update.php

POST http://13.127.121.188/rest_pms/api/archived_projects/create.php

POST http://13.127.121.188/rest_pms/api/completed_projects/delete.php

GET http://13.127.121.188/rest_pms/api/completed_projects/read_single.php?id=1

GET http://13.127.121.188/rest_pms/api/completed_projects/read.php

POST http://13.127.121.188/rest_pms/api/completed_projects/update.php

POST http://13.127.121.188/rest_pms/api/completed_projects/create.php

POST http://13.127.121.188/rest_pms/api

No Environment

Send Save Code

Body form-data x-www-form-urlencoded raw binary JSON (application/json)

```

1 - [
2   "1"
3 ]

```

Body Cookies Headers (11) Test Results

Pretty Raw Preview JSON

```

1 - [
2   "message": "Deleted"
3 ]

```

Status: 200 OK Time: 298 ms

Activate Windows
Go to Settings to activate Windows.

PUT http://13.127.121.188/rest_pms/api/employee/create.php

```

1 * {
2   "name": "Dhruba Jyoti Ghosh",
3   "job_title": "admin",
4   "email": "djh@pms.com",
5   "phone": 7978422701,
6   "pid": 3,
7   "did": 4
8 }

```

Status: 200 OK Time: 298 ms

```

1 * {
2   "message": "Created"
3 }

```

Activate Windows
Go to Settings to activate Windows.

PUT http://13.127.121.188/rest_pms/api/employee/update.php

```

1 * {
2   "id": 3,
3   "name": "Dhruba Jyoti Ghosh",
4   "job_title": "manager",
5   "email": "djh@pms.com",
6   "phone": 7978422701,
7   "pid": 3,
8   "pid": 3,
9   "did": 4
}

```

Status: 200 OK Time: 330 ms

```

1 * {
2   "message": "Updated"
3 }

```

Activate Windows
Go to Settings to activate Windows.

Filter

History Collections Clear all

Today

GET http://13.127.121.188/rest_pms/api/employee/read.php

PUT http://13.127.121.188/rest_pms/api/employee/update.php

PUT http://13.127.121.188/rest_pms/api/employee/update.php

PUT http://13.127.121.188/rest_pms/api/employee/create.php

PUT http://13.127.121.188/rest_pms/api/employee/create.php

PUT http://13.127.121.188/rest_pms/api/employee/create.php

PUT http://13.127.121.188/rest_pms/api/archived_projects/delete.php

GET http://13.127.121.188/rest_pms/api/archived_projects/read_single.php?id=1

GET http://13.127.121.188/rest_pms/api/archived_projects/read.php

POST http://13.127.121.188/rest_pms/api/archived_projects/update.php

POST http://13.127.121.188/rest_pms/api/archived_projects/update.php

POST http://13.127.121.188/rest_pms/api/archived_projects/delete.php

POST http://13.127.121.188/rest_pms/api/archived_projects/read.php

POST http://13.127.121.188/rest_pms/api/archived_projects/create.php

POST http://13.127.121.188/rest_pms/api/archived_projects/delete.php

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (11) Test Results

Pretty Raw Preview JSON

```

1 - [
2   {
3     "id": "3",
4     "name": "Dhruba Jyoti Ghosh",
5     "email": "djjg@pms.com",
6     "phone": "7978422701",
7     "job_title": "manager",
8     "pid": "3",
9     "did": "3"
10   }
11 ]

```

Status: 200 OK Time: 686 ms

Activate Windows
Go to Settings to activate Windows.

Filter

History Collections Clear all

Today

GET http://13.127.121.188/rest_pms/api/employee/read_single.php?id=3

GET http://13.127.121.188/rest_pms/api/employee/read.php

PUT http://13.127.121.188/rest_pms/api/employee/update.php

PUT http://13.127.121.188/rest_pms/api/employee/update.php

PUT http://13.127.121.188/rest_pms/api/employee/create.php

PUT http://13.127.121.188/rest_pms/api/employee/create.php

PUT http://13.127.121.188/rest_pms/api/archived_projects/delete.php

GET http://13.127.121.188/rest_pms/api/archived_projects/read_single.php?id=1

GET http://13.127.121.188/rest_pms/api/archived_projects/read.php

POST http://13.127.121.188/rest_pms/api/archived_projects/update.php

POST http://13.127.121.188/rest_pms/api/archived_projects/update.php

POST http://13.127.121.188/rest_pms/api/archived_projects/delete.php

POST http://13.127.121.188/rest_pms/api/archived_projects/read.php

POST http://13.127.121.188/rest_pms/api/archived_projects/create.php

POST http://13.127.121.188/rest_pms/api/archived_projects/delete.php

Authorization Headers (1) Body Pre-request Script Tests

Type No Auth

Body Cookies Headers (11) Test Results

Pretty Raw Preview JSON

```

1 - [
2   {
3     "id": "3",
4     "name": "Dhruba Jyoti Ghosh",
5     "email": "djjg@pms.com",
6     "phone": "7978422701",
7     "job_title": "manager",
8     "pid": "3",
9     "did": "4"
10   }
11 ]

```

Status: 200 OK Time: 322 ms

Activate Windows
Go to Settings to activate Windows.

The screenshot shows the Postman interface with a successful DELETE request to `http://13.127.121.188/rest_pms/api/employee/delete.php`. The request body is set to `JSON (application/json)` and contains the JSON object `{ "id": 3 }`. The response status is `200 OK` with a time of `705 ms`, and the message is `"message": "Deleted"`.

3.4.4 Data in Database

The screenshot shows the phpMyAdmin interface connected to the database `project_mgmt_system`. The `company` table is selected, displaying two rows of data:

	id	name	email	address	phone
1	EY	contactus@in ey com	New Delhi	1166718000	
2	HCL	contactus@hcl.in	New Delhi	5325252412	

phpMyAdmin

Server: localhost | Database: project_mgmt_system | Table: department

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 2 (3 total, Query took 0.0002 seconds.)

SELECT * FROM `department`

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Options id name strength email

Edit Copy Delete 4 Development 0 tech@pms.com
Edit Copy Delete 5 Testing 0 testing@pms.com
Edit Copy Delete 6 Documenters 0 docs@pms.com

Check all With selected: Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

Activate Windows
Go to Settings to activate Windows.

Console

	id	name	strength	email
4	Development	0	tech@pms.com	
5	Testing	0	testing@pms.com	
6	Documenters	0	docs@pms.com	

phpMyAdmin

Server: localhost | Database: project_mgmt_system | Table: ongoing_projects

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 1 (2 total, Query took 0.0002 seconds.)

SELECT * FROM `ongoing_projects`

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Options id name start_date deadline cid did

Edit Copy Delete 3 Hospital Management System 2019-09-01 2019-09-14 1 4
Edit Copy Delete 4 Student Grading System 2019-09-02 2019-09-19 2 6

Check all With selected: Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

Activate Windows
Go to Settings to activate Windows.

Console

	id	name	start_date	deadline	cid	did
3	Hospital Management System	2019-09-01	2019-09-14	1	4	
4	Student Grading System	2019-09-02	2019-09-19	2	6	

phpMyAdmin

Server: localhost | Database: project_mgmt_system | Table: completed_projects

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

SELECT * FROM `completed_projects`

Profiling Edit inline | Edit Explain SQL Create PHP code Refresh

	id	name	start_date	completion_date	cld	did
1	2	Airlines Management System	2019-01-21	2019-02-22	1	5

Show all Number of rows: 25 Filter rows: Search this table

Options Edit Copy Delete With selected: Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table

Query results operations Print Copy to clipboard Export Display chart Create view

Activate Windows Go to Settings to activate Windows.

Console

The screenshot shows the phpMyAdmin interface for the completed_projects table. The table has columns: id, name, start_date, completion_date, cld, and did. There is one row with id 2, name 'Airlines Management System', start_date '2019-01-21', completion_date '2019-02-22', cld 1, and did 5. Navigation buttons like 'Show all' and 'Filter rows' are visible. Below the table, there are 'Query results operations' buttons for Print, Copy to clipboard, Export, Display chart, and Create view.

phpMyAdmin

Server: localhost | Database: project_mgmt_system | Table: archived_projects

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 0 (1 total, Query took 0.0002 seconds.)

SELECT * FROM `archived_projects`

Profiling Edit inline | Edit Explain SQL Create PHP code Refresh

	id	name	start_date	completion_date	achievement	cld	did
1	2	Admission Management System	2019-03-12	2019-06-04	Project of the year	1	4

Show all Number of rows: 25 Filter rows: Search this table

Options Edit Copy Delete With selected: Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table

Query results operations Print Copy to clipboard Export Display chart Create view

Activate Windows Go to Settings to activate Windows.

Console

The screenshot shows the phpMyAdmin interface for the archived_projects table. The table has columns: id, name, start_date, completion_date, achievement, cld, and did. There is one row with id 2, name 'Admission Management System', start_date '2019-03-12', completion_date '2019-06-04', achievement 'Project of the year', cld 1, and did 4. Navigation buttons like 'Show all' and 'Filter rows' are visible. Below the table, there are 'Query results operations' buttons for Print, Copy to clipboard, Export, Display chart, and Create view.

phpMyAdmin

Recent Favorites

information_schema
mysql
performance_schema
project_mgmt_system
New
archived_projects
company
completed_projects
department
employee
ongoing_projects

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 2 (3 total, Query took 0.0002 seconds.)

SELECT * FROM `employee`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

	id	name	email	phone	job_title	pid	did
<input type="checkbox"/>	4	Priyanka Pattnaik	pp@pms.com	9798126039	manager	3	5
<input type="checkbox"/>	5	Shubrat Saha	ss@pms.com	8700798716	manager	4	6
<input type="checkbox"/>	6	Dhruba Jyoti Ghosh	djg@pms.com	9798422701	manager	3	4

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Check all With selected: Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Query results operations

Print Copy to clipboard Export Display chart Create view

Activate Windows
Go to Settings to activate Windows.

Console

The screenshot shows the phpMyAdmin interface for a MySQL database named 'project_mgmt_system'. The left sidebar lists various databases and tables. The 'employee' table is selected in the main workspace. A query 'SELECT * FROM `employee`' is run, and the results are displayed in a table with columns: id, name, email, phone, job_title, pid, and did. There are three rows of data. Below the table are navigation links like 'Show all', 'Filter rows', 'Sort by key', and 'Check all'. At the bottom, there's a 'Query results operations' section with options like 'Print', 'Copy to clipboard', 'Export', 'Display chart', and 'Create view'. A watermark for 'Activate Windows' is visible at the bottom right.

CHAPTER – 4 FURTHER STUDY

4.1 Summary

We have overall summarized the project embedding technologies and hosting in cloud platform, in order to simplify the complex algorithms, analyzing insights out of historical data and using to generate reports to enhance business growth.

The project currently is a prototype but can be genuinely used by businesses in keeping regular project records and other functional requirements as their project development practice primary resource.

4.2 Project Scalability

As discussed in the above segmentations regarding the project status, the system is currently a prototype which has a lot of works remaining for deploying into the market as a finished product.

Scalability of the product is extreme at every level and can be genuinely used by the service provider if focused more on the UI/UX, security and data storage.

Creating a genuine attractive and cognitive front end is primarily required to deploy the project to start using the functionalities. The application can be automatically scaled up and scaled down by AWS as per the traffic load using Elastic Load Balancer.

CENTURION UNIVERSITY OF TECHNOLOGY & MANAGEMENT

School of Engineering & Technology

The Minor Project I is developed as a part of the B.Tech curriculum, to fulfil the credit requirements of 2 credits under the Basket V, of the CBCS 2016-20 academic syllabus. The Project is not aimed to fulfil commercial requirements and only to practically implement the knowledge gained during academic sessions.



Centurion
UNIVERSITY

*Shaping Lives...
Empowering Communities...*