``

CC6012NP

Module Code & Module Title

CC6012NP Data and Web Development

Assessment Type

Final Report

Semester-

2024/25 Autumn

Student Name: Achyut Tamang

London Met ID: 22068943

College ID: NP04CP4A220011

Assignment Due Date: Thursday, December 19, 2024

Assignment Submission Date: Friday, March 26, 2025

Submitted To:

Ms. Pratibha Gurung

Declaration

I confirm that I understand my coursework needs to be submitted online via My second teacher under the relevant module page before the deadline for my assignment to be accepted and marked. I am fully aware that late submission will be treated as non-submission and a mark of zero will be awarded

Table of Contents:

Table of Figures:

Table of Tables:

1. Introduction:

This coursework is an individual assignment that contributes 40% of the total module marks. It is designed to evaluate students' ability to solve real-world problems, apply critical thinking, and assess database system design and development. The task involves analyzing, designing, and building a web-based database application based on a given business case. Along with the software solution, students must provide detailed documentation explaining the design and implementation process.

1.1. Case Study:

LS Corporation, a mid-sized technology company, faced challenges in managing multiple projects due to scattered systems. This led to inefficiencies, poor communication, and difficulty tracking project progress. To resolve these issues, the company decided to implement a centralized project management system. The goal was to organize data, assign tasks efficiently, and improve overall project tracking.

System Requirement:

- Each user (developer, designer, project manager, etc.) can be involved in multiple projects.
- Every project consists of multiple tasks and milestones (e.g., prototype completion, beta release) with deadlines.
- A user can be assigned to multiple tasks, and each task can have multiple users working on it.
- Tasks may have subtasks, but each subtask belongs to only one task.
- Users can comment on tasks to improve communication.
- Tasks may require multiple resources for completion.

2. Initial ERD:

An Entity Relationship Diagram (ERD) is a visual representation of the relationships between entities within a database. It serves as a crucial tool in database design, helping to illustrate how different data elements interact with one another (Belcic, 2024). In the following initial ERD diagram, the entities are shown in a light pinkish color with a yellow border, primary keys are highlighted in blue, and foreign keys are highlighted in purple.

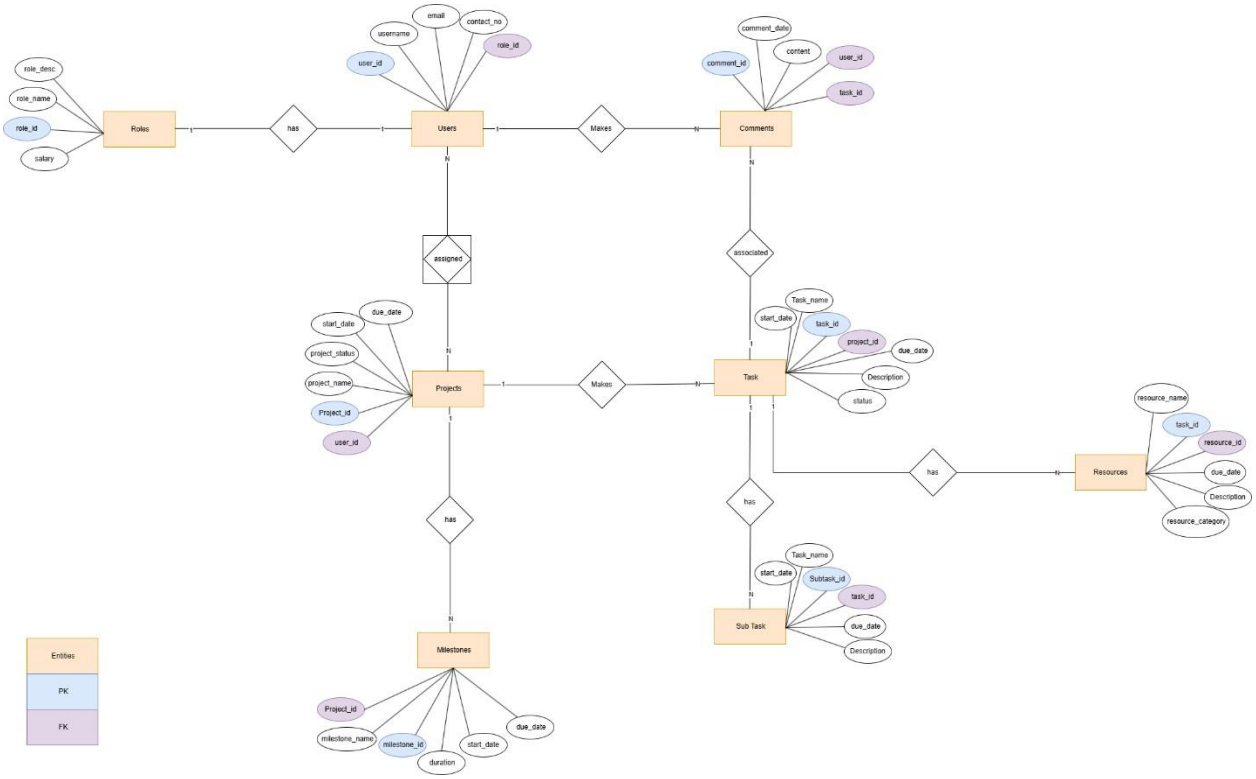


Figure 1: Inital ERD

2.1 Entities and Attributes:

Table 1: Entities & Attributes

Entities	Attributes
Users	userId (Primary Key), username, email contact , role_id (foreign key)
Role	role_id (Primary key), role_name, role_desc, salary
Task	task_id (Primary Key) ,task_name, taskStartDate, taskDueDate, Status, description, project_id (Foreign Key)
SubTask	Subtask_id (Primary key), subtask_name, start_date, due_date, description, task_id (foregin key)
Resources	rescource_id (Primary Key), resource_name, resource_categeory, due_date, description, task_id(foregin key)
Project	project_name, project_id (Primary key), project_status, start_date, due_date, user_id (Foregin Key)
Milestone	milestone_id (Primary Key), milestone_name, duration, start_date, due_date, project_id (Foregin Key)
Comments	Commet_id (Primary key), commet_date, commet_text, task_id (Foregin Key), user_id (Foregin Key)

2.2 Relationship:

- Each users have one role, but each role can be assigned to multiple users.
- Each comment is made by one user, but each user can make multiple comments.
- Each project is assigned to one user, but each user can be assigned to multiple projects.
- Each task belongs to one project, but each project can contain multiple tasks.
- Each milestone is associated with one project, but each project can have multiple milestones.
- Each subtask belongs to one task, but each task can contain multiple subtasks.
- Each resource is associated with one task, but each task can have multiple resources.
- Each comment is linked to one task, but each task can have multiple comments.

3. Normalization:

Normalization is the process of organizing data in a database to minimize redundancy and eliminate undesirable characteristics such as insertion, update, and deletion anomalies. It involves dividing larger tables into smaller ones and linking those using relationships. The goal of normalization is to reduce redundancy and ensure a more efficient and consistent database structure by applying normal forms to the tables (Anon., 2024).

UNF

Unnormalized form (UNF or 0NF), also known as an unnormalized relation or nonfirst normal form (N1NF or NF2), is a database data model (organization of data in a database) which does not meet any of the conditions of database normalization defined by the relational model.

User ID: U-01
User Name: Sam Smith
User Email: Smith@gmail.com
User contact: +9779859697989

Project ID	Project Name	Project Start Date	Project Due Date	Project Status	Task ID	Task Name	Start Date	Due Date	Status
P-01	Enrolment System	2024-01-01	2024-10-23	On going	T-01	Student Registration	2024-01-01	2024-02-20	Completed
P-01	Enrolment System	2024-01-01	2024-10-23	On going	T-02	Student Counselling Form	2024-01-05	2024-06-15	On going
P-02	Attendance System	2024-03-01	2024-12-12	On going	T-03	Biometric Registration	2024-03-10	2024-03-20	Completed

Figure 2: UNF Table

1NF:

First Normal Form (1NF) is a basic rule in organizing data within a database to ensure efficiency and consistency. A table is in 1NF when each column contains only single, indivisible values (atomicity), and there are no repeating groups or duplicate rows. This means that each field holds a unique piece of data, and similar data isn't stored in multiple

columns or rows. Achieving 1NF helps in reducing redundancy and simplifies data management (Fayard, 2025).

Users1: (user_id, user_name, user_email, user_contact, role_id, role_name, role_desc role_salary)

Table 2: 1NF of Users1

user_id	user_name	user_email	user_contact	role_id	role_name	role_desc	role_salary
U001	Achyut	achyut@gmail.com	984770555	R001	Manager	Oversees projects & teams	80000
U002	Simmi	simranpak@gmail.com	9856247056	R002	Developer	Writes & maintance code	60000

Projects1: (user_id*, project_id, project_name, project_start_date, project_due_date, project_status)

Table 3: 1NF of Projects1

user_id	project_id	project_name	project_start_date	project_start_end	project_status
U001	P001	Project Alpha	2025-01-01	2025-06-01	pending
U002	P002	Project Beta	2024-01-0	2024-1-31	completed

Tasks1: (project_id*, task_id, task_name, task_status, task_start_date, task_due_date)

Table 4: 1NF of Tasks1

project_id	task_id	task_name	task_status	task_start_date	task_due_date
P001	T001	Desgin UI	In Progress	2025-01-01	2025-01-31
P002	T002	Code Backend	Completed	2024-12-01	2025-12-01

2NF:

Second Normal Form (2NF) is a stage in database normalization that builds upon First Normal Form (1NF) to further reduce redundancy and improve data integrity. A table is in 2NF if it is first in 1NF and all non-key attributes are fully dependent on the entire primary key, not just part of it. This means that if a table has a composite primary key (a primary key made up of multiple columns), each non-key attribute must relate to the whole key, not just a portion. By achieving 2NF, we eliminate partial dependencies, leading to a more efficient and consistent database design (Kolli, 2024)**.**

Partial dependency:

Project_id -> project_name, project_start_date, project_due_date, project_status Task_id -> task_name, task_start_date, task_due_date, task_status

Users2 (user_id, user_name, user_email, user_contact, role_id, role_name, role_desc role_salary)

Table 5: 2NF of Users2

user_id	user_name	user_email	user_contact	role_id	role_name	role_desc	role_salary
U001	Achyut	achyut@gmail.com	984770555	R001	Manager	Oversees projects & teams	80000
U002	Simmi	simranpak@gmail.com	9856247056	R002	Developer	Writes & maintained code	60000

User_project2 (user_id*, project_id*)

Table 6: 2NF of User_project2

user_id	project_id
U001	P001
U002	P002

Projects2 (project_id, project_name, project_start_date, project_due_date, project_satuts)

Table 7: 2NF of Projects2

project_id	project_name	project_start_date	project_due_date	project_satuts
P001	Project Alpha	2025-01-01	2025-06-01	Active
P002	Project Beta	2024-01-01	2025-12-31	Completed

Project_task2 (project_id*, task_id*)

Table 8: 2NF Project_task2

project_id	task_id
P001	T001
P002	T002

Tasks2 (task_id, task_name, task_start_date, task_due_date, task_status)

Table 9: 2NF of Tasks2

task_id	task_name	task_start_date	task_due_date	task_satuts
T001	Design UI	2025-01-01	2025-02-01	Initial Setup
T002	Code Backend	2024-01-01	2024-03-01	Testing Phase

3NF:

Third Normal Form (3NF) is a database design principle that builds upon Second Normal Form (2NF) to further reduce redundancy and ensure data integrity. A table is in 3NF if it is already in 2NF and all non-key attributes are not only dependent on the primary key but are also independent of each other, meaning there are no transitive dependencies. In simpler terms, each non-key attribute should provide information directly about the primary key and not about other non-key attributes. Achieving 3NF helps in minimizing data duplication and potential anomalies during data operations (Kozubek, 2020).

Transitive Dependency:

user_id -> role_id -> role_name, role_desc, role_salary

The third normalized form for the system is given below:

Users3 (user_id, user_name, user_email, user_contact,role_id*)

user_id	user_name	user_email	user_contact	role_id
U001	Achyut	achyut@gmail.com	9847730555	R001
U002	Simmi	simranpak@gmail.com	9856247056	R002

Roles3 (role_id, role_name, role_desc, role_salary)

role_id	role_name	role_desc	role_salary
R001	Manager	Oversees projects & teams	80000
R002	Developer	Writes & maintans code	60000

User_project(user_id*, project_id*)

user_id	project_id
U001	P001
U002	P002

Projects2 (project_id, project_name, project_start_date, project_due_date, project_satuts)

project_id	project_name	project_start_date	project_due_date	project_satuts
P001	Project Alpha	2025-01-01	2025-06-01	Active
P002	Project Beta	2024-01-01	2025-12-31	Completed

Project_task3(project_id*, task_id*)

project_id	task_id
P001	T001
P002	T002

Tasks2 (task_id, task_name, task_status, task_start_date, task_due_date)

task_id	task_name	task_start_date	task_due_date	task_satuts
T001	Design UI	2025-01-01	2025-02-01	Initial Setup
T002	Code Backend	2024-01-01	2024-03-01	Testing Phase

4. Database Implementation:

4.1 Granting user and creating tables:

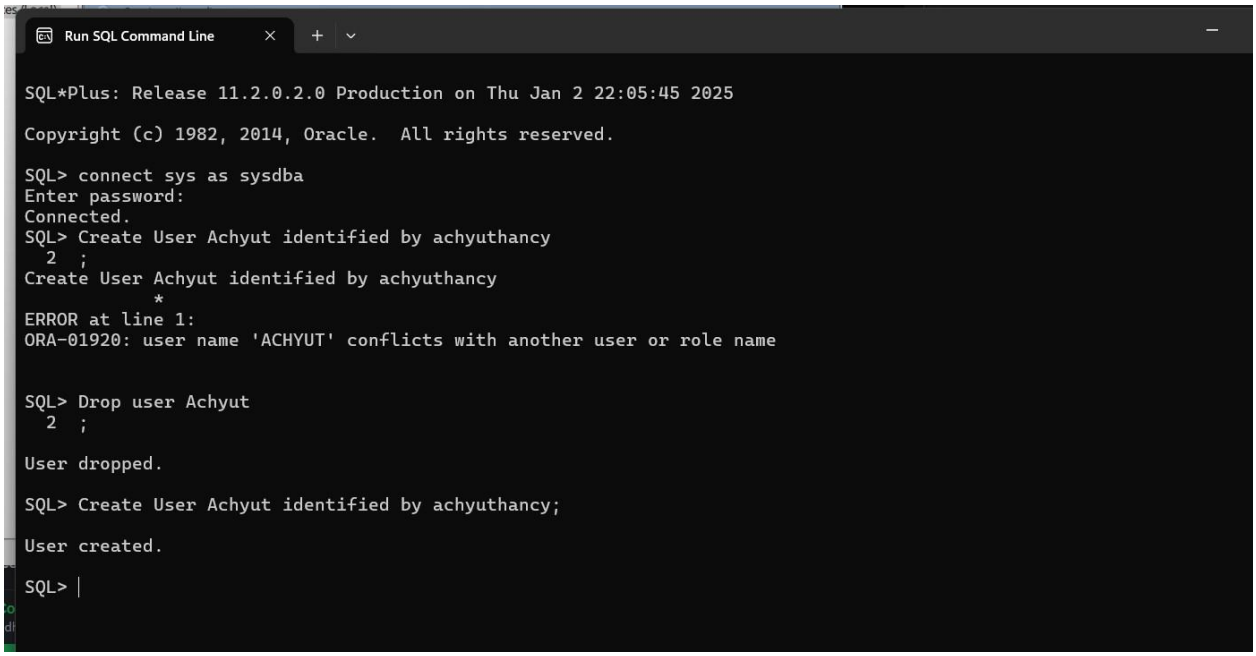


Figure 3: Creating new user

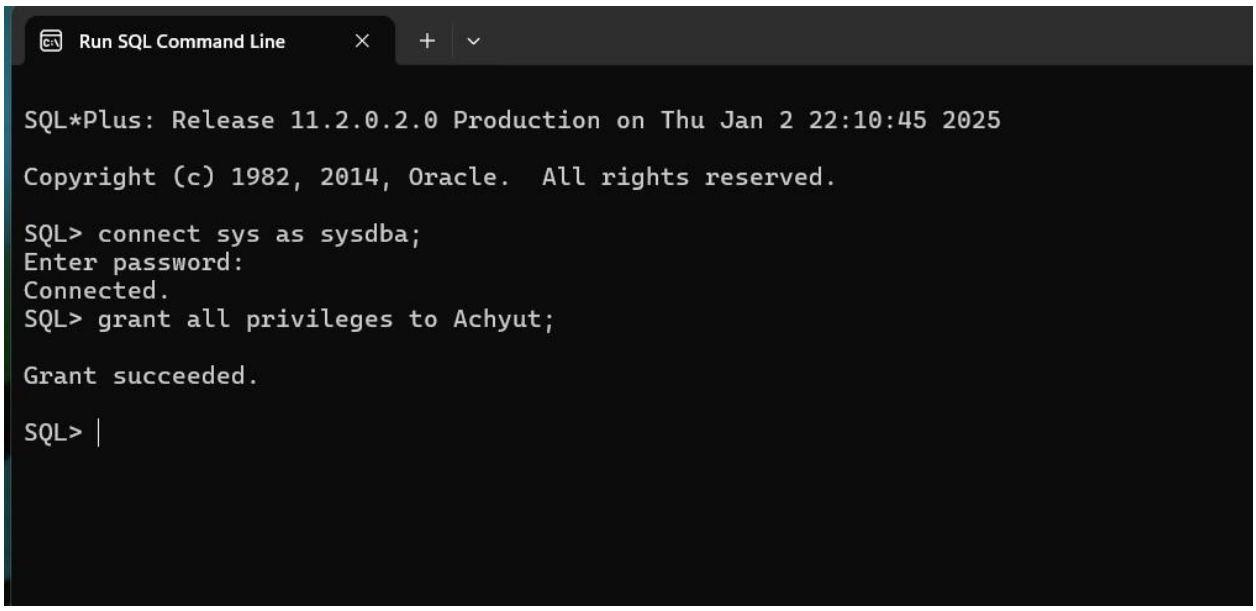


Figure 4: Grant All Privileges

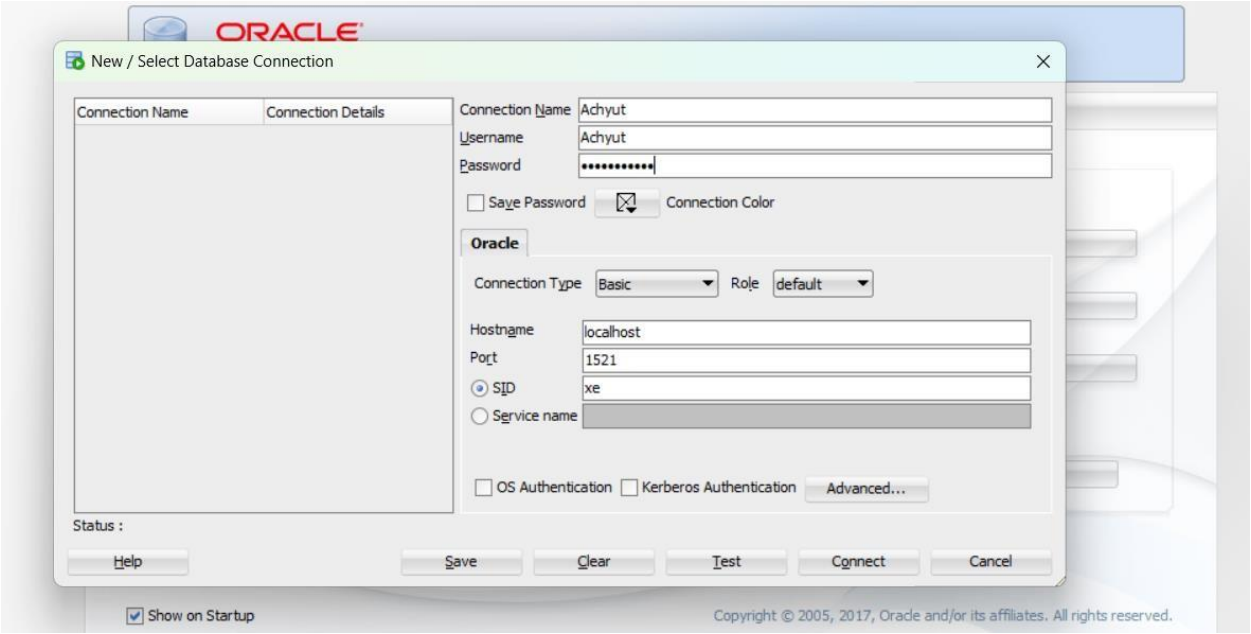


Figure 5: Connecting to Database

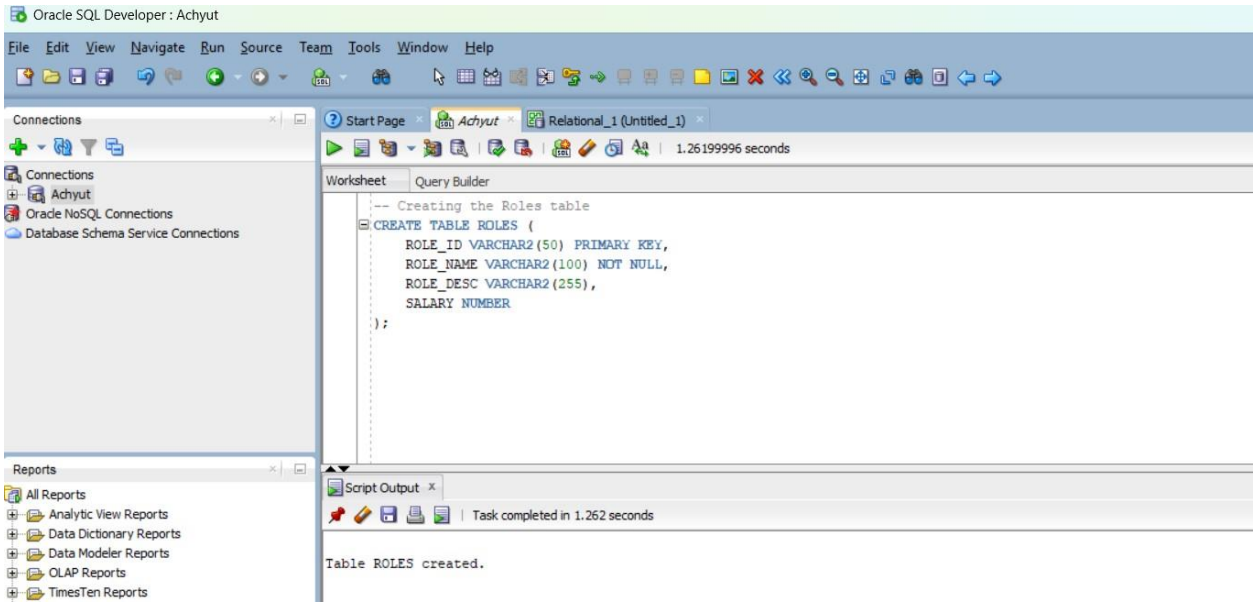


Figure 6: Creating Table for roles

```
-- Creating the Users table
CREATE TABLE USERS (
    USER_ID VARCHAR2(50) PRIMARY KEY,
    ROLE_ID VARCHAR2(50),
    USERNAME VARCHAR2(100) NOT NULL,
    EMAIL VARCHAR2(100) UNIQUE NOT NULL,
    CONTACT_NO VARCHAR2(15),
    FOREIGN KEY (ROLE_ID) REFERENCES ROLES(ROLE_ID)
);
```

Figure 7: Creating Tables for Users

```
-- Creating the Projects table
CREATE TABLE PROJECTS (
    PROJECT_ID VARCHAR2(50) PRIMARY KEY,
    USER_ID VARCHAR2(50),
    PROJECT_NAME VARCHAR2(100) NOT NULL,
    PROJECT_STATUS VARCHAR2(50),
    START_DATE DATE NOT NULL,
    DUE_DATE DATE,
    FOREIGN KEY (USER_ID) REFERENCES USERS(USER_ID)
);
```

Figure 8: Creating Table for projects

```
-- Creating the Milestones table
CREATE TABLE MILESTONES (
    MILESTONE_ID VARCHAR2(50) PRIMARY KEY,
    PROJECT_ID VARCHAR2(50),
    MILESTONE_NAME VARCHAR2(100) NOT NULL,
    DURATION NUMBER,
    START_DATE DATE NOT NULL,
    DUE_DATE DATE,
    FOREIGN KEY (PROJECT_ID) REFERENCES PROJECTS(PROJECT_ID)
);
```

Figure 9: Creating Tables for Milestones

```
-- Creating the Tasks table
CREATE TABLE TASKS (
    TASK_ID VARCHAR2(50) PRIMARY KEY,
    PROJECT_ID VARCHAR2(50),
    TASK_NAME VARCHAR2(100) NOT NULL,
    DESCRIPTION VARCHAR2(255),
    STATUS VARCHAR2(50),
    START_DATE DATE NOT NULL,
    DUE_DATE DATE,
    FOREIGN KEY (PROJECT_ID) REFERENCES PROJECTS(PROJECT_ID)
);
```

Figure 10: Creating Tables for Tasks

```
-- Creating the Subtasks table
CREATE TABLE SUBTASKS (
    SUBTASK_ID VARCHAR2(50) PRIMARY KEY,
    TASK_ID VARCHAR2(50),
    DESCRIPTION VARCHAR2(255),
    START_DATE DATE NOT NULL,
    DUE_DATE DATE,
    FOREIGN KEY (TASK_ID) REFERENCES TASKS(TASK_ID)
);
```

Figure 11: Creating tables for Subtask

```
-- Creating the Resources table
CREATE TABLE RESOURCES (
    RESOURCE_ID VARCHAR2(50) PRIMARY KEY,
    RESOURCE_NAME VARCHAR2(100) NOT NULL,
    TASK_ID VARCHAR2(50),
    DESCRIPTION VARCHAR2(255),
    RESOURCE_CATEGORY VARCHAR2(50),
    DUE_DATE DATE,
    FOREIGN KEY (TASK_ID) REFERENCES TASKS(TASK_ID)
);
```

Figure 12: Creating Table for resources

```
-- Creating the Comments table
CREATE TABLE COMMENTS (
    COMMENT_ID VARCHAR2(50) PRIMARY KEY,
    COMMENT_DATE DATE DEFAULT SYSDATE,
    CONTENT CLOB NOT NULL,
    USER_ID VARCHAR2(50),
    TASK_ID VARCHAR2(50),
    FOREIGN KEY (USER_ID) REFERENCES USERS(USER_ID),
    FOREIGN KEY (TASK_ID) REFERENCES TASKS(TASK_ID)
);
```

Figure 13: Creating Tables for comments

4.2 Data Insertion:

```
-- Insert values into Roles table
INSERT INTO ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, SALARY) VALUES ('R001', 'Manager', 'Oversees projects and teams', 80000);
INSERT INTO ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, SALARY) VALUES ('R002', 'Developer', 'Writes and maintains code', 60000);
INSERT INTO ROLES (ROLE_ID, ROLE_NAME, ROLE_DESC, SALARY) VALUES ('R003', 'Tester', 'Tests and ensures quality', 50000);
```

```
Script Output
Task completed in 1.094 seconds

1 row inserted.
```

Figure 14: Inserting Values in roles table

```
-- Insert values into Users table
INSERT INTO USERS (USER_ID, ROLE_ID, USERNAME, EMAIL, CONTACT_NO) VALUES ('U001', 'R001', 'Achyut', 'achyut@gmail.com', '9847730555');
INSERT INTO USERS (USER_ID, ROLE_ID, USERNAME, EMAIL, CONTACT_NO) VALUES ('U002', 'R002', 'Simmi', 'simranpak@gmail.com', '9856247056');
INSERT INTO USERS (USER_ID, ROLE_ID, USERNAME, EMAIL, CONTACT_NO) VALUES ('U003', 'R003', 'Muskan', 'muskangmail.com', '9785102305');
```

Figure 15: Inserting Values in users table

```
-- Insert values into Projects table
INSERT INTO PROJECTS (PROJECT_ID, USER_ID, PROJECT_NAME, PROJECT_STATUS, START_DATE, DUE_DATE) VALUES ('P001', 'U001', 'Project Alpha', 'Active', TO_DATE('2025-01-01', 'YYYY-MM-DD'), TO_DATE('202
INSERT INTO PROJECTS (PROJECT_ID, USER_ID, PROJECT_NAME, PROJECT_STATUS, START_DATE, DUE_DATE) VALUES ('P002', 'U002', 'Project Beta', 'Completed', TO_DATE('2024-01-01', 'YYYY-MM-DD'), TO_DATE('2
INSERT INTO PROJECTS (PROJECT_ID, USER_ID, PROJECT_NAME, PROJECT_STATUS, START_DATE, DUE_DATE) VALUES ('P003', 'U003', 'Project Gamma', 'Planned', TO_DATE('2025-07-01', 'YYYY-MM-DD'), TO_DATE('20
```

Figure 16: Inserting values in Project table

```
-- Insert values into Milestones table
INSERT INTO MILESTONES (MILESTONE_ID, PROJECT_ID, MILESTONE_NAME, DURATION, START_DATE, DUE_DATE) VALUES ('M001', 'P001', 'Initial Setup', 30, TO_DATE('2025-01-01', 'YYYY-MM-DD'), TO_DATE('2025-0
INSERT INTO MILESTONES (MILESTONE_ID, PROJECT_ID, MILESTONE_NAME, DURATION, START_DATE, DUE_DATE) VALUES ('M002', 'P002', 'Testing Phase', 60, TO_DATE('2024-10-01', 'YYYY-MM-DD'), TO_DATE('2024-1
INSERT INTO MILESTONES (MILESTONE_ID, PROJECT_ID, MILESTONE_NAME, DURATION, START_DATE, DUE_DATE) VALUES ('M003', 'P003', 'Planning Stage', 15, TO_DATE('2025-07-01', 'YYYY-MM-DD'), TO_DATE('2025-

-- Insert values into Tasks table
```

Figure 17: Inserting values in milestone table

```
-- Insert values into Tasks table
INSERT INTO TASKS (TASK_ID, PROJECT_ID, TASK_NAME, DESCRIPTION, STATUS, START_DATE, DUE_DATE) VALUES ('T001', 'P001', 'Design UI', 'Create user interface', 'In Progress',
TO_DATE('2025-01-01', 'YYYY-MM-DD'), TO_DATE('2025-02-01', 'YYYY-MM-DD'));
INSERT INTO TASKS (TASK_ID, PROJECT_ID, TASK_NAME, DESCRIPTION, STATUS, START_DATE, DUE_DATE) VALUES ('T002', 'P002', 'Code Backend', 'Develop API endpoints', 'Completed',
TO_DATE('2024-01-01', 'YYYY-MM-DD'), TO_DATE('2024-03-01', 'YYYY-MM-DD'));
INSERT INTO TASKS (TASK_ID, PROJECT_ID, TASK_NAME, DESCRIPTION, STATUS, START_DATE, DUE_DATE) VALUES ('T003', 'P003', 'Plan Architecture', 'Define system architecture', 'Planned',
TO_DATE('2025-07-01', 'YYYY-MM-DD'), TO_DATE('2025-07-15', 'YYYY-MM-DD'));
```

Figure 18: Inserting values in task table

```
-- Insert values into Subtasks table
INSERT INTO SUBTASKS (SUBTASK_ID, TASK_ID, DESCRIPTION, START_DATE, DUE_DATE) VALUES ('ST001', 'T001', 'Design homepage',
TO_DATE('2025-01-01', 'YYYY-MM-DD'), TO_DATE('2025-01-15', 'YYYY-MM-DD'));
INSERT INTO SUBTASKS (SUBTASK_ID, TASK_ID, DESCRIPTION, START_DATE, DUE_DATE) VALUES ('ST002', 'T002', 'Develop login API',
TO_DATE('2024-01-05', 'YYYY-MM-DD'), TO_DATE('2024-01-20', 'YYYY-MM-DD'));
INSERT INTO SUBTASKS (SUBTASK_ID, TASK_ID, DESCRIPTION, START_DATE, DUE_DATE) VALUES ('ST003', 'T003', 'Research tech stack',
TO_DATE('2025-07-01', 'YYYY-MM-DD'), TO_DATE('2025-07-05', 'YYYY-MM-DD'));
```

Figure 19: Inserting Values in Subtask table

```
-- Insert values into Resources table
INSERT INTO RESOURCES (RESOURCE_ID, RESOURCE_NAME, TASK_ID, DESCRIPTION, RESOURCE_CATEGORY, DUE_DATE) VALUES ('R001', 'Laptop', 'T001', 'High-performance laptop', 'Hardware',
TO_DATE('2025-02-01', 'YYYY-MM-DD'));
INSERT INTO RESOURCES (RESOURCE_ID, RESOURCE_NAME, TASK_ID, DESCRIPTION, RESOURCE_CATEGORY, DUE_DATE) VALUES ('R002', 'API Documentation', 'T002', 'Guidelines for API use', 'Document',
TO_DATE('2024-03-01', 'YYYY-MM-DD'));
INSERT INTO RESOURCES (RESOURCE_ID, RESOURCE_NAME, TASK_ID, DESCRIPTION, RESOURCE_CATEGORY, DUE_DATE) VALUES ('R003', 'Database', 'T003', 'PostgreSQL setup', 'Software',
TO_DATE('2025-07-15', 'YYYY-MM-DD'));
```

Figure 20: Inserting values into resource table

```
-- Insert values into Comments table
INSERT INTO COMMENTS (COMMENT_ID, COMMENT_DATE, CONTENT, USER_ID, TASK_ID) VALUES ('C001', SYSDATE, 'Great progress so far!', 'U001', 'T001');
INSERT INTO COMMENTS (COMMENT_ID, COMMENT_DATE, CONTENT, USER_ID, TASK_ID) VALUES ('C002', SYSDATE, 'Please review the API documentation.', 'U002', 'T002');
INSERT INTO COMMENTS (COMMENT_ID, COMMENT_DATE, CONTENT, USER_ID, TASK_ID) VALUES ('C003', SYSDATE, 'Architecture looks solid.', 'U003', 'T003');
```

Figure 21: Inserting values in comments tables

5. Final ERD:

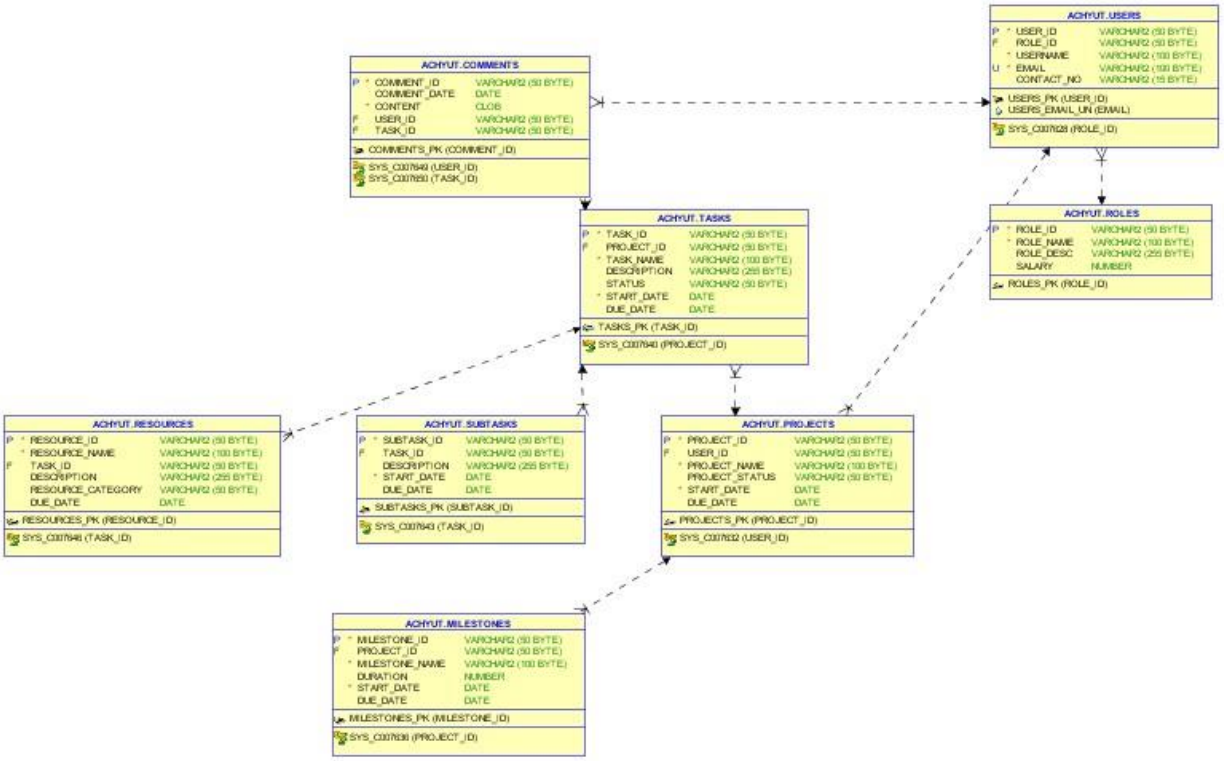
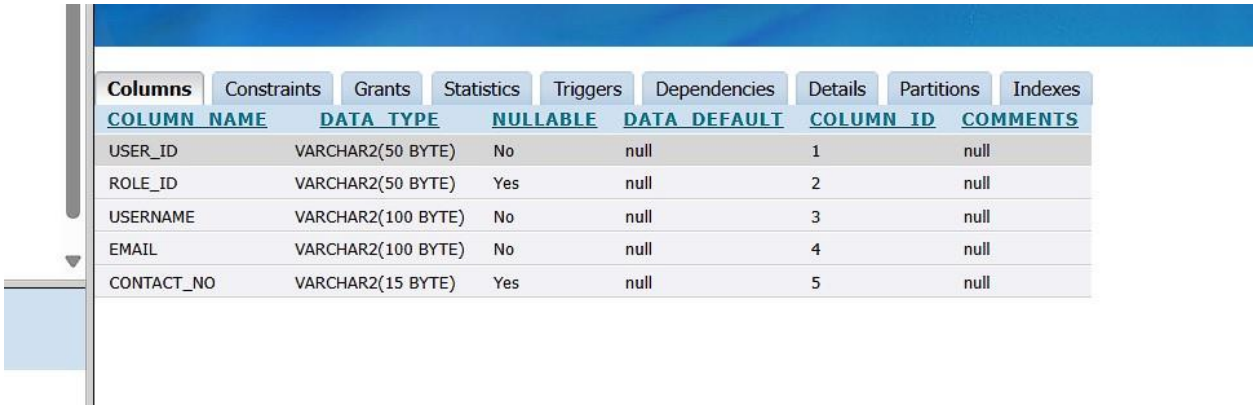


Figure 22: Final ERD

6. Data Dictionary:

A data dictionary is a centralized repository that provides detailed information about the data within a database or information system. It includes names, definitions, and attributes of data elements, such as data types, sizes, formats, and relationships between data elements. This documentation helps users and developers understand the structure, content, and meaning of the data, ensuring consistency and clarity in data management and usage (Chai, 2019).

1. Users:

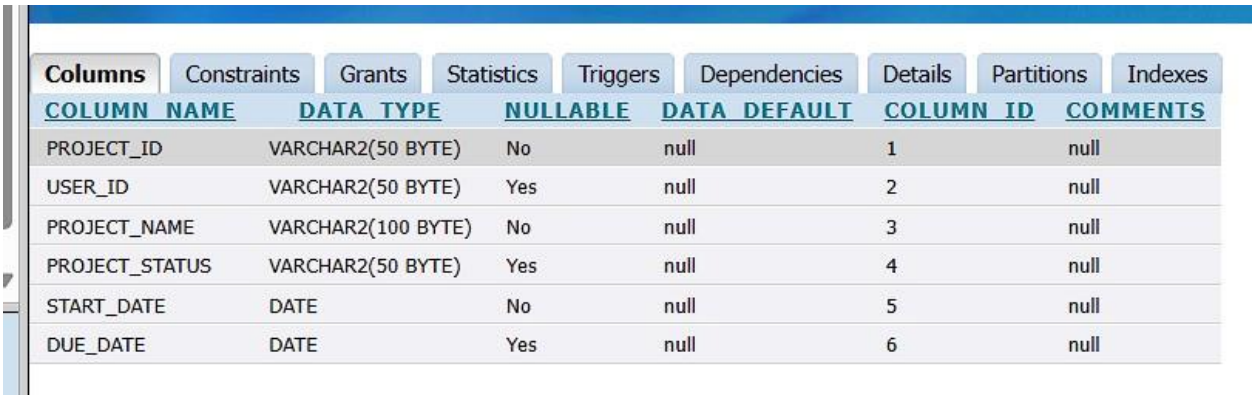


Columns | Constraints | Grants | Statistics | Triggers | Dependencies | Details | Partitions | Indexes

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
USER_ID	VARCHAR2(50 BYTE)	No	null	1	null
ROLE_ID	VARCHAR2(50 BYTE)	Yes	null	2	null
USERNAME	VARCHAR2(100 BYTE)	No	null	3	null
EMAIL	VARCHAR2(100 BYTE)	No	null	4	null
CONTACT_NO	VARCHAR2(15 BYTE)	Yes	null	5	null

Figure 23: Data Dictionary of Users table

2. Project:

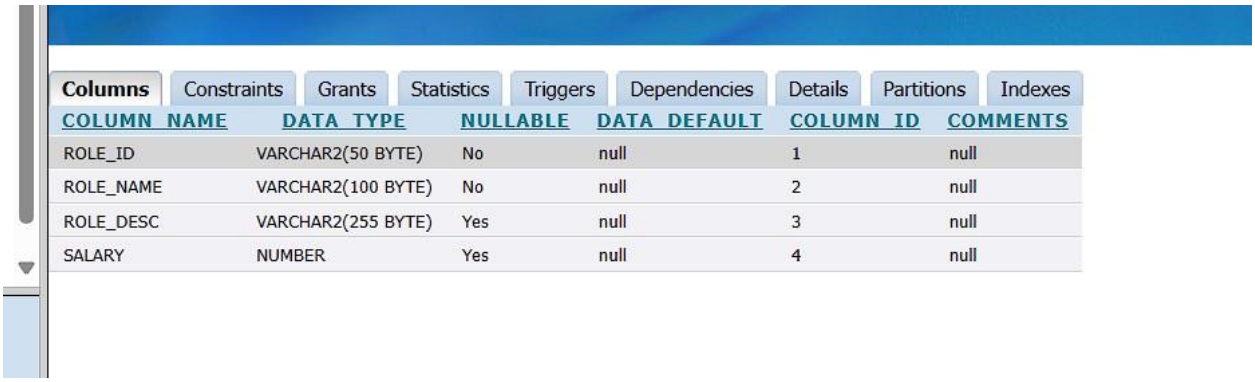


Columns | Constraints | Grants | Statistics | Triggers | Dependencies | Details | Partitions | Indexes

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
PROJECT_ID	VARCHAR2(50 BYTE)	No	null	1	null
USER_ID	VARCHAR2(50 BYTE)	Yes	null	2	null
PROJECT_NAME	VARCHAR2(100 BYTE)	No	null	3	null
PROJECT_STATUS	VARCHAR2(50 BYTE)	Yes	null	4	null
START_DATE	DATE	No	null	5	null
DUE_DATE	DATE	Yes	null	6	null

Figure 24: Data Dictionary of Project table

3. Roles:

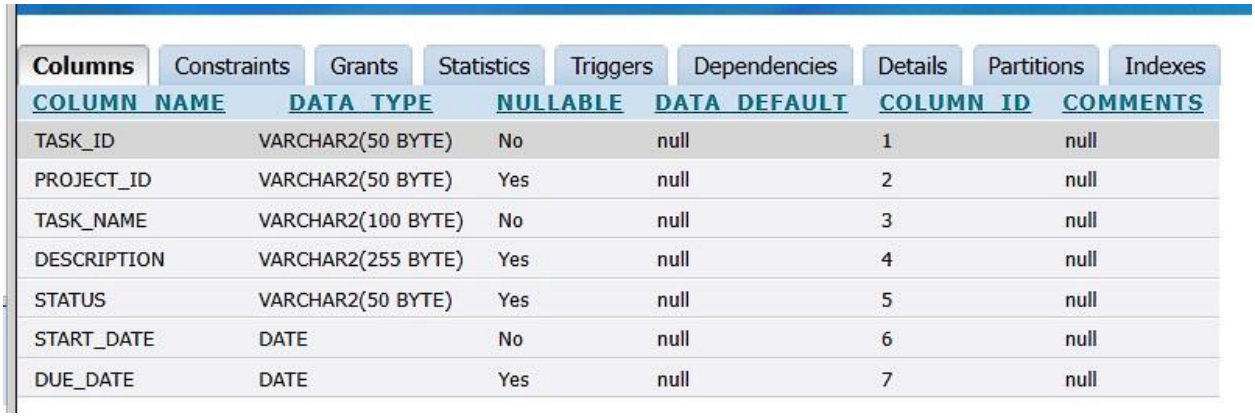


Columns | Constraints | Grants | Statistics | Triggers | Dependencies | Details | Partitions | Indexes

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
ROLE_ID	VARCHAR2(50 BYTE)	No	null	1	null
ROLE_NAME	VARCHAR2(100 BYTE)	No	null	2	null
ROLE_DESC	VARCHAR2(255 BYTE)	Yes	null	3	null
SALARY	NUMBER	Yes	null	4	null

Figure 25: Data Dictionary of roles

4. Tasks:

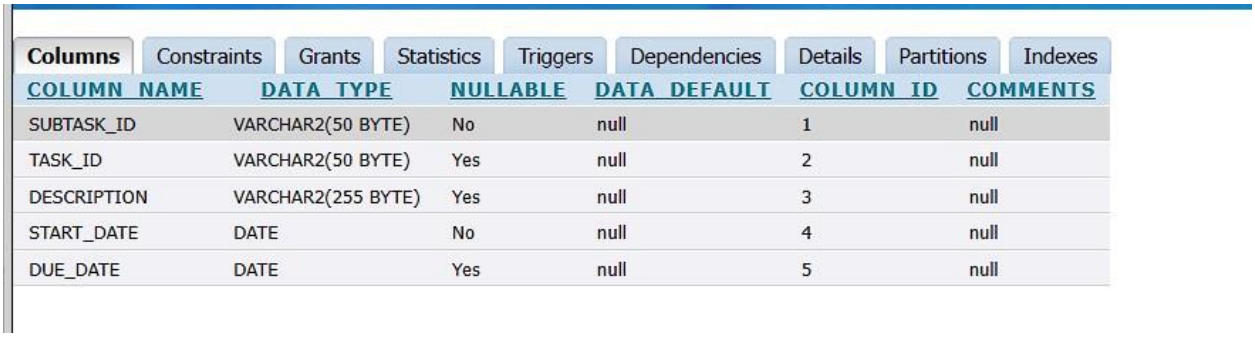


Columns | Constraints | Grants | Statistics | Triggers | Dependencies | Details | Partitions | Indexes

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
TASK_ID	VARCHAR2(50 BYTE)	No	null	1	null
PROJECT_ID	VARCHAR2(50 BYTE)	Yes	null	2	null
TASK_NAME	VARCHAR2(100 BYTE)	No	null	3	null
DESCRIPTION	VARCHAR2(255 BYTE)	Yes	null	4	null
STATUS	VARCHAR2(50 BYTE)	Yes	null	5	null
START_DATE	DATE	No	null	6	null
DUE_DATE	DATE	Yes	null	7	null

Figure 26: Data Dictionary for tasks

5. Subtask:

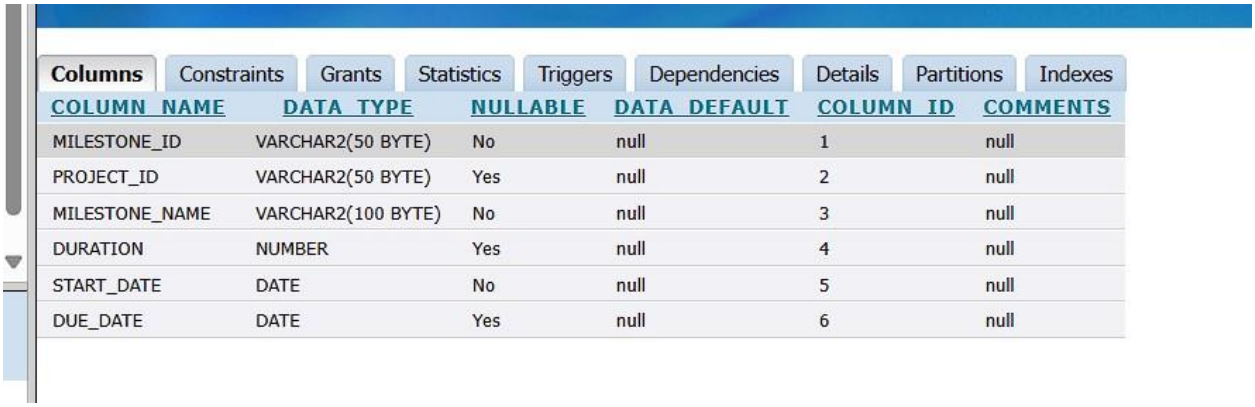


Columns | Constraints | Grants | Statistics | Triggers | Dependencies | Details | Partitions | Indexes

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
SUBTASK_ID	VARCHAR2(50 BYTE)	No	null	1	null
TASK_ID	VARCHAR2(50 BYTE)	Yes	null	2	null
DESCRIPTION	VARCHAR2(255 BYTE)	Yes	null	3	null
START_DATE	DATE	No	null	4	null
DUE_DATE	DATE	Yes	null	5	null

Figure 27:Data Dictionary of Subtask table

6. Milestone:

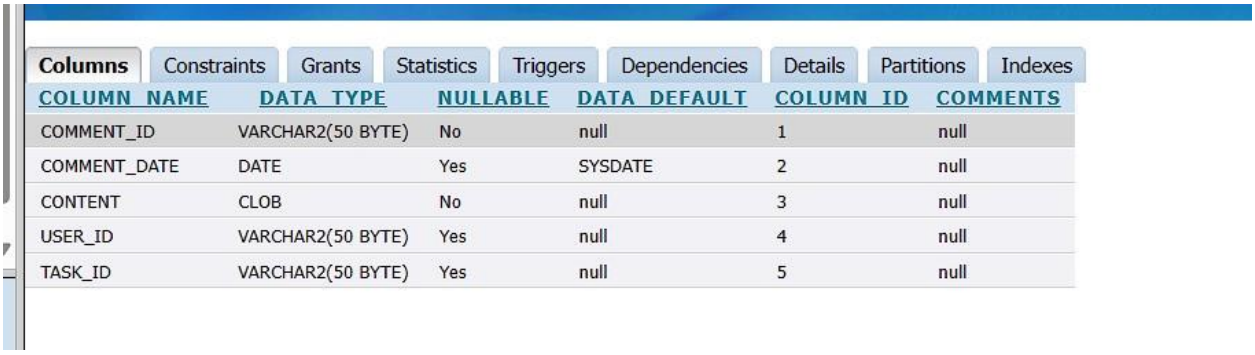


Columns | Constraints | Grants | Statistics | Triggers | Dependencies | Details | Partitions | Indexes

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
MILESTONE_ID	VARCHAR2(50 BYTE)	No	null	1	null
PROJECT_ID	VARCHAR2(50 BYTE)	Yes	null	2	null
MILESTONE_NAME	VARCHAR2(100 BYTE)	No	null	3	null
DURATION	NUMBER	Yes	null	4	null
START_DATE	DATE	No	null	5	null
DUE_DATE	DATE	Yes	null	6	null

Figure 28: Data Dictionary of Milestone

7. Comments:

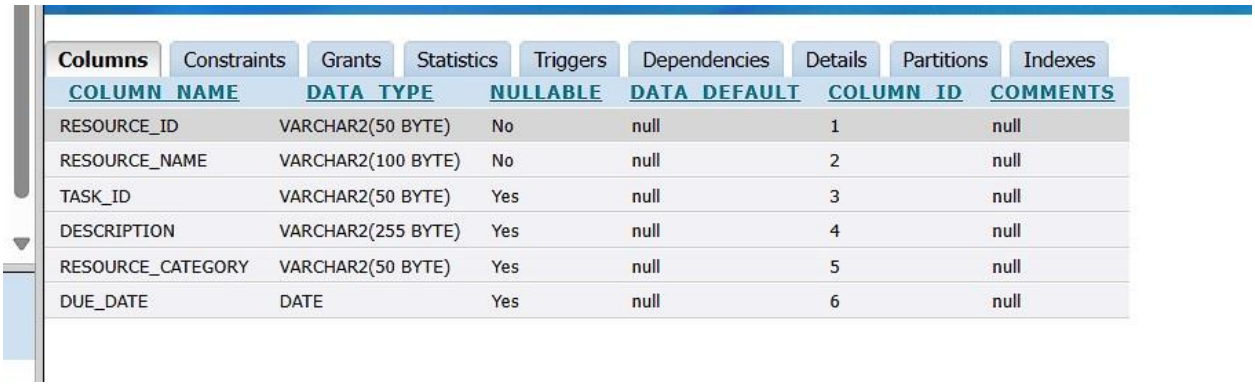


Columns | Constraints | Grants | Statistics | Triggers | Dependencies | Details | Partitions | Indexes

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
COMMENT_ID	VARCHAR2(50 BYTE)	No	null	1	null
COMMENT_DATE	DATE	Yes	SYSDATE	2	null
CONTENT	CLOB	No	null	3	null
USER_ID	VARCHAR2(50 BYTE)	Yes	null	4	null
TASK_ID	VARCHAR2(50 BYTE)	Yes	null	5	null

Figure 29:Data Dictionary of comments

8. Resources:



Columns | Constraints | Grants | Statistics | Triggers | Dependencies | Details | Partitions | Indexes

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
RESOURCE_ID	VARCHAR2(50 BYTE)	No	null	1	null
RESOURCE_NAME	VARCHAR2(100 BYTE)	No	null	2	null
TASK_ID	VARCHAR2(50 BYTE)	Yes	null	3	null
DESCRIPTION	VARCHAR2(255 BYTE)	Yes	null	4	null
RESOURCE_CATEGORY	VARCHAR2(50 BYTE)	Yes	null	5	null
DUE_DATE	DATE	Yes	null	6	null

Figure 30: Data Dictionary for resources

7. Webforms:

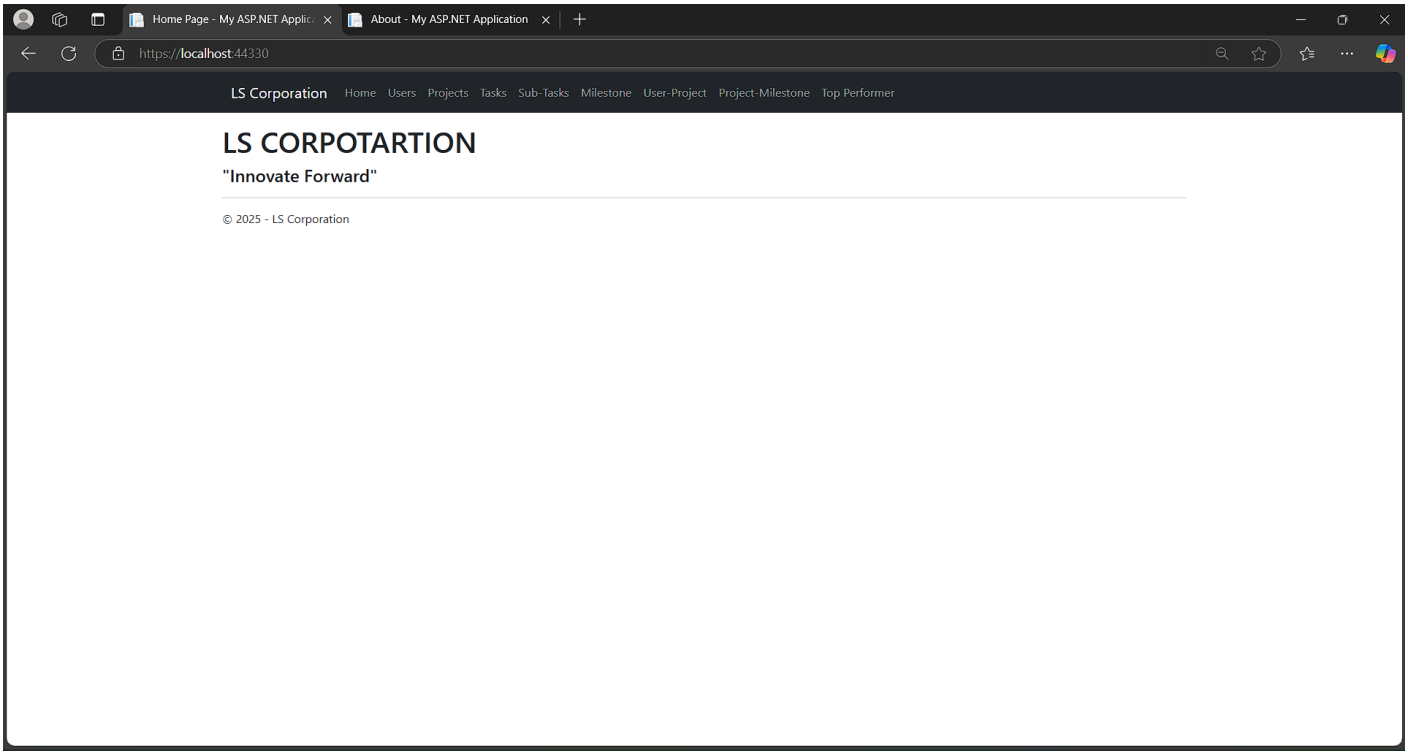


Figure 31:Home Page

1. User Details:

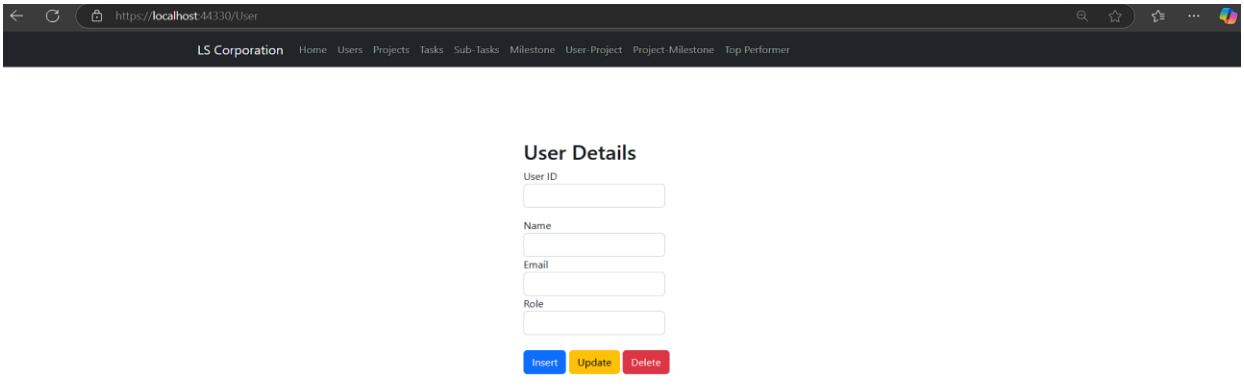


Figure 32: user details

2. Project Details:

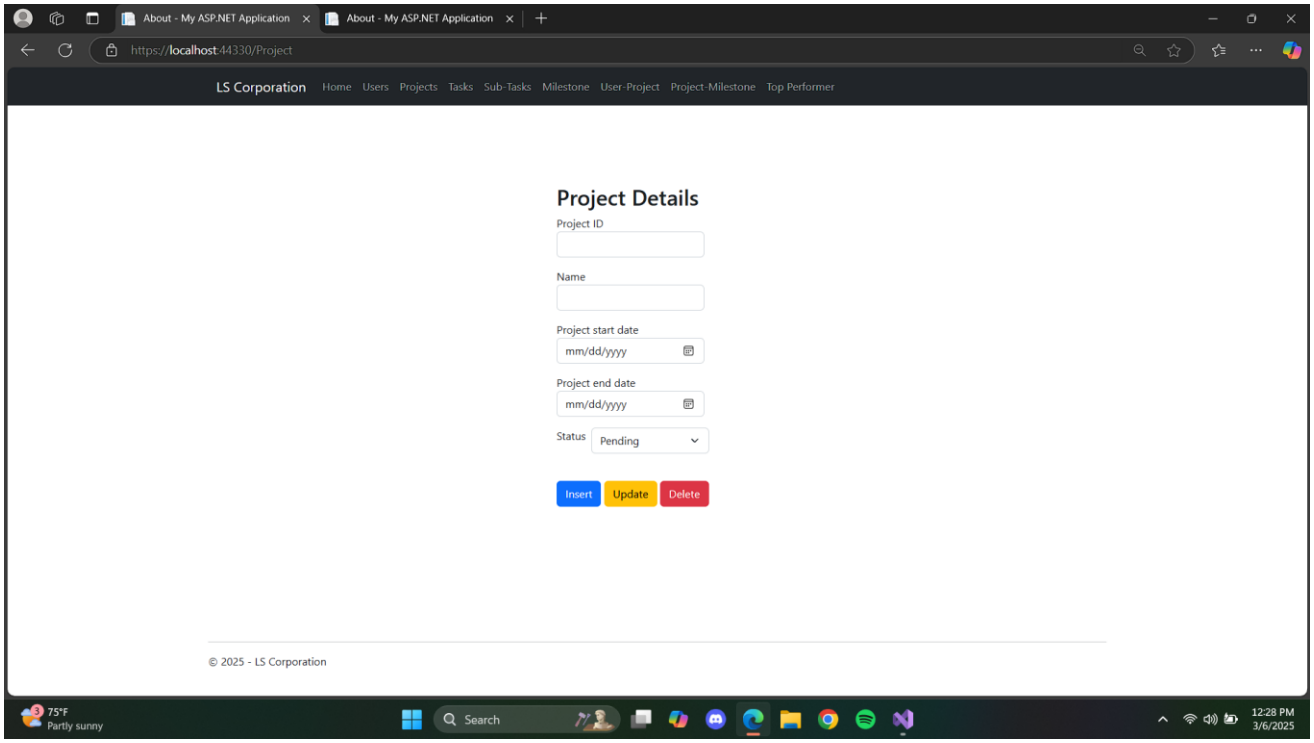


Figure 33: Project Details

3. Task Details:

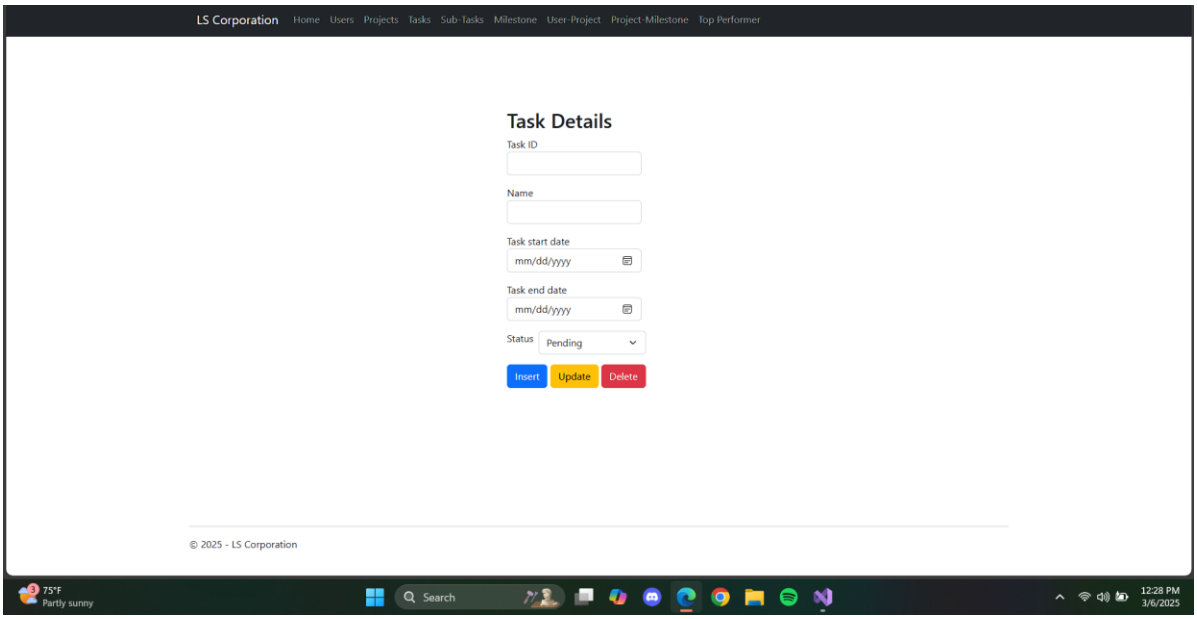


Figure 34: Task Deatils

4. SubTask Details:

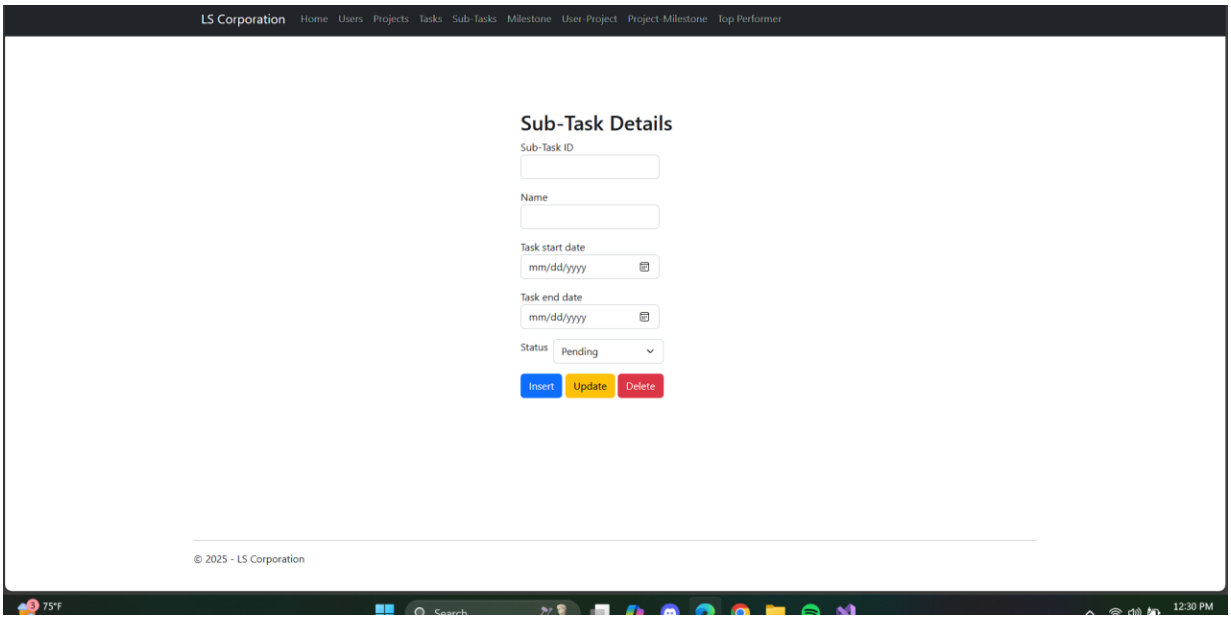


Figure 35: Sub-Task Details

5. Milestone Details:

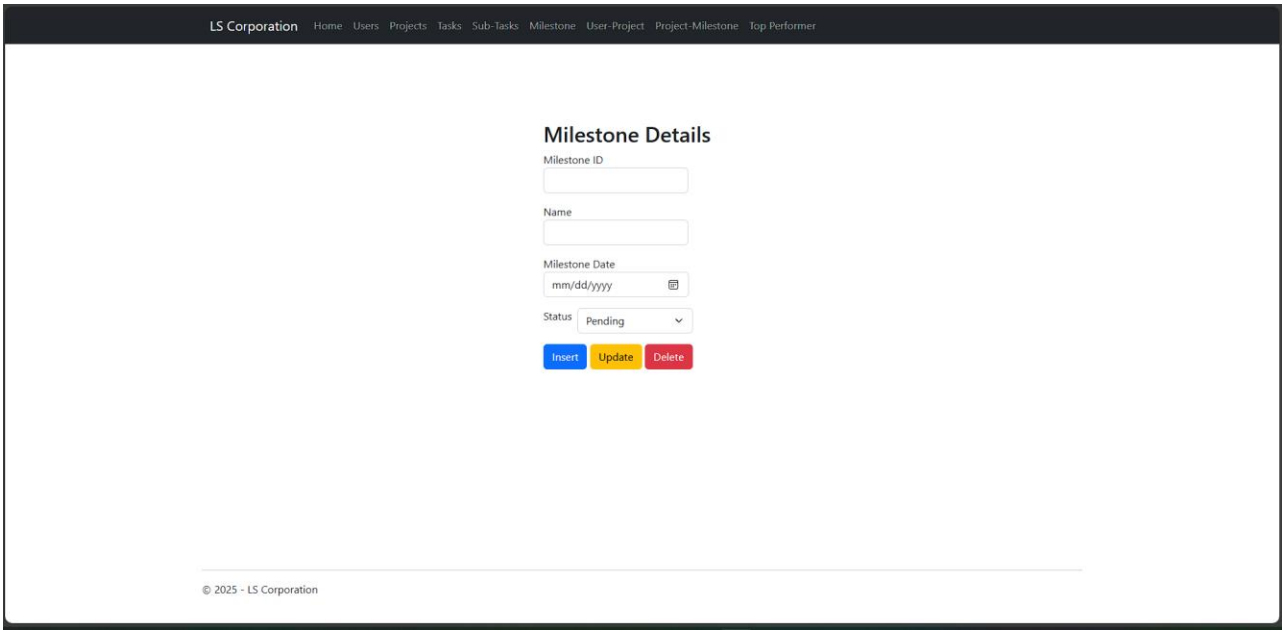


Figure 36: Milestone details

8. Testing of web form:

8.1 Users:

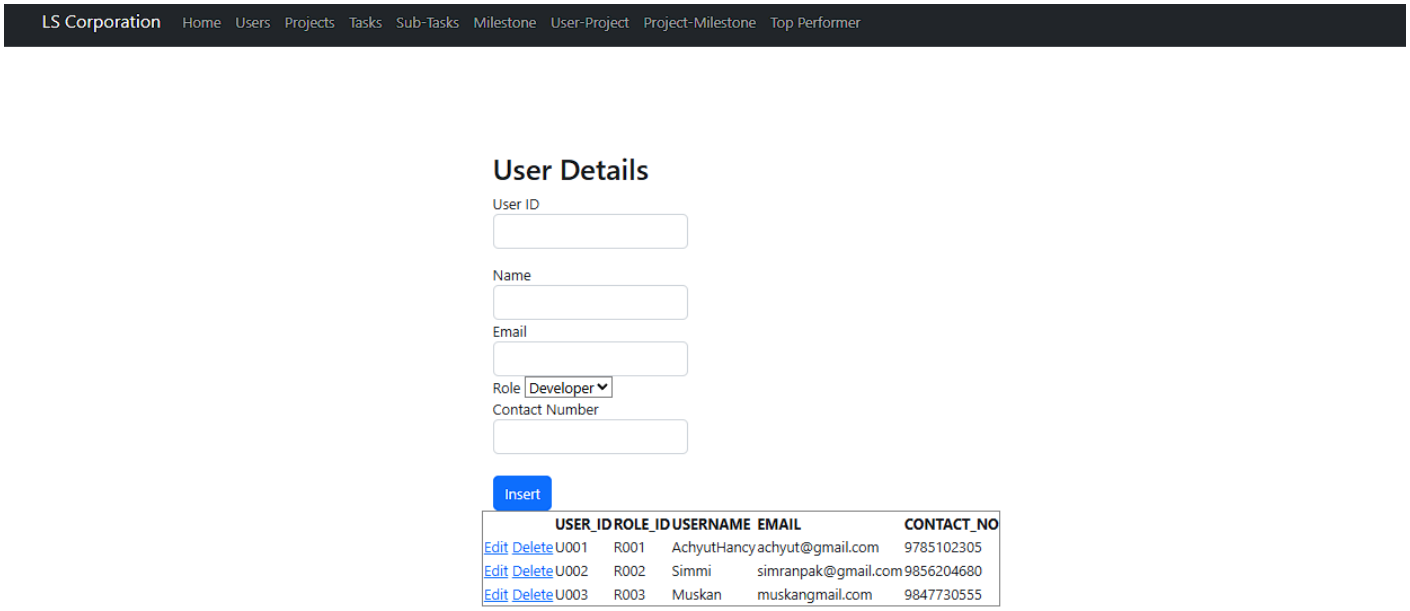


Figure 37: Basic Webform of User details with CRUD

Adding User Details:

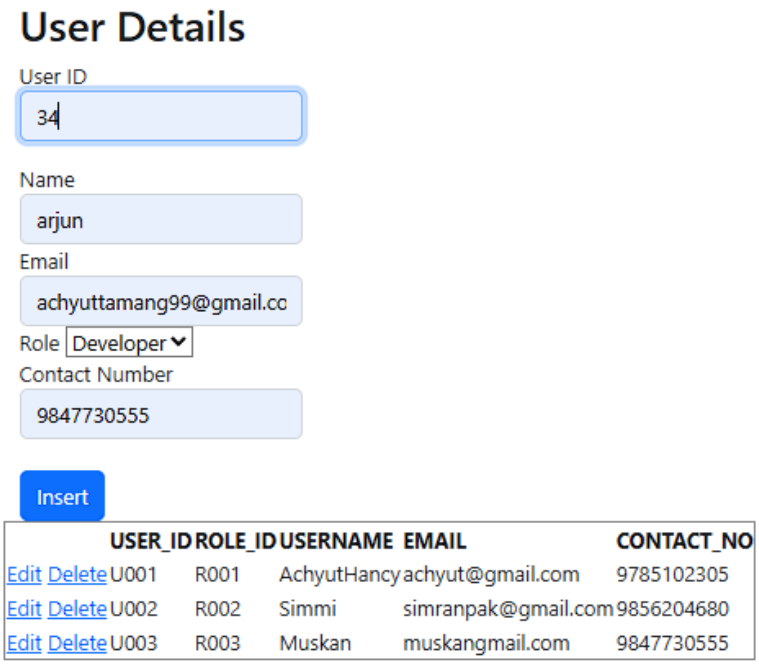


Figure 38: Inserting values in user details

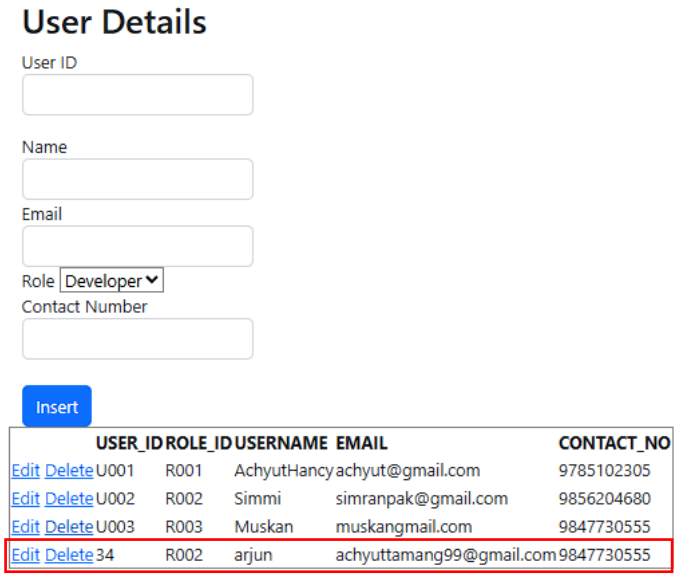


Figure 39: User details successfully added

Table 10: Add a new user testcase

Objective	To test inserting a new user into the system
Action	Enter all the required user details and click insert
Expected Result	System should add a new user successfully and display it on the user list
Actual Result	The users details were successfully added and displayed in the user list
Conclusion	Successfully executed.

Updating User Details:

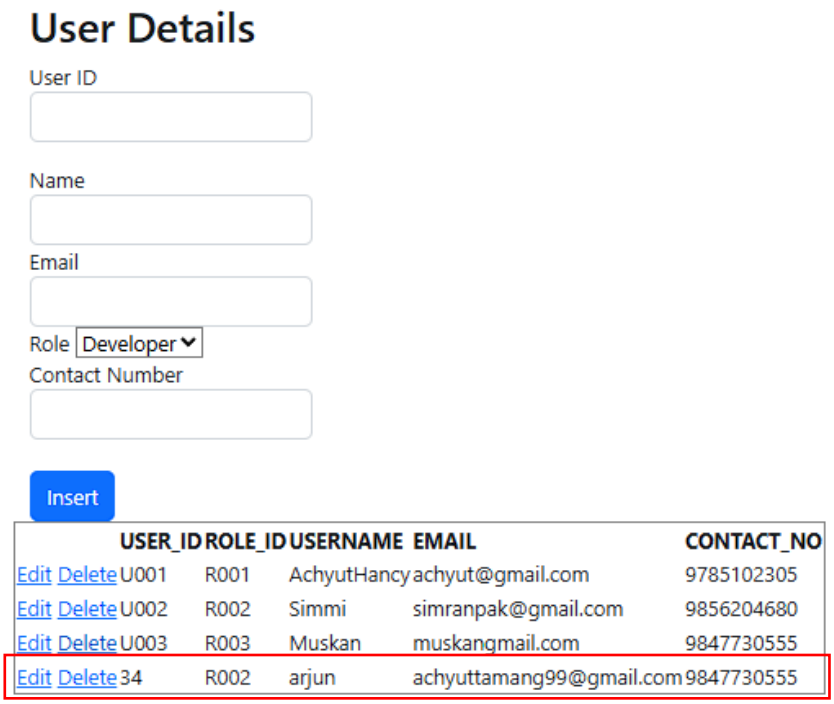


Figure 40: before updating user details

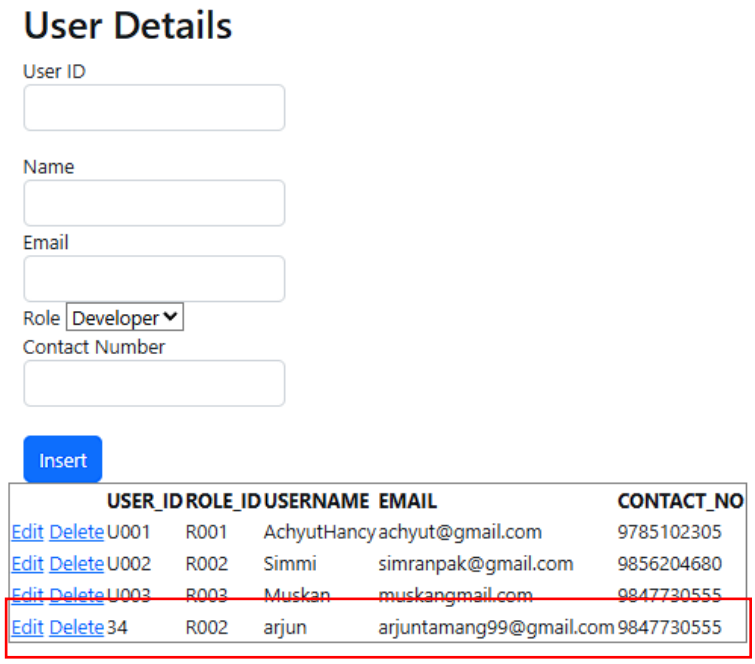


Figure 41: After updating user details

Table 11: Update User Details

Objective	To update user details into the system
Action	Edit and update the user information
Expected Result	System should update the field where user have updated
Actual Result	The users mail was updated
Conclusion	Successfully executed.

Deleting User from the list:

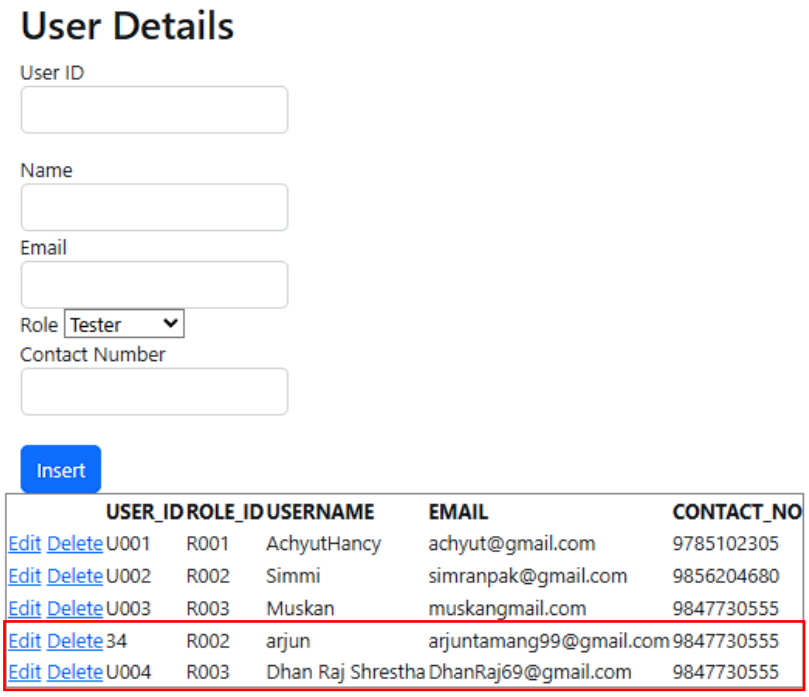


Figure 42: Before Deleting User details

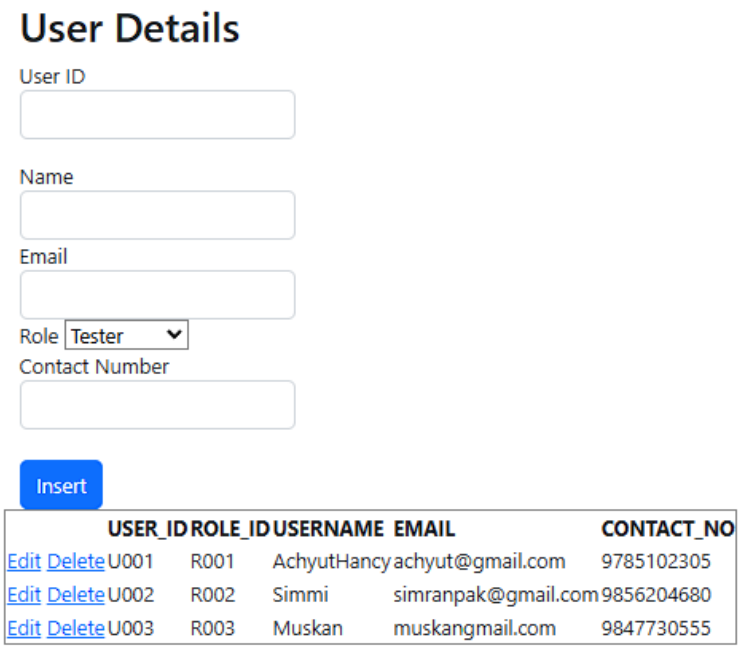


Figure 43: After Deleting user details

Table 12: Deleting User from the user list

Objective	To delete a user from the user list
Action	Clicking on the delete option and deleting user from the user list.
Expected Result	Users should be deleted from the user list
Actual Result	Two newly added users were deleted as shown in the figure above.
Conclusion	Successfully executed.

8.2 Project:

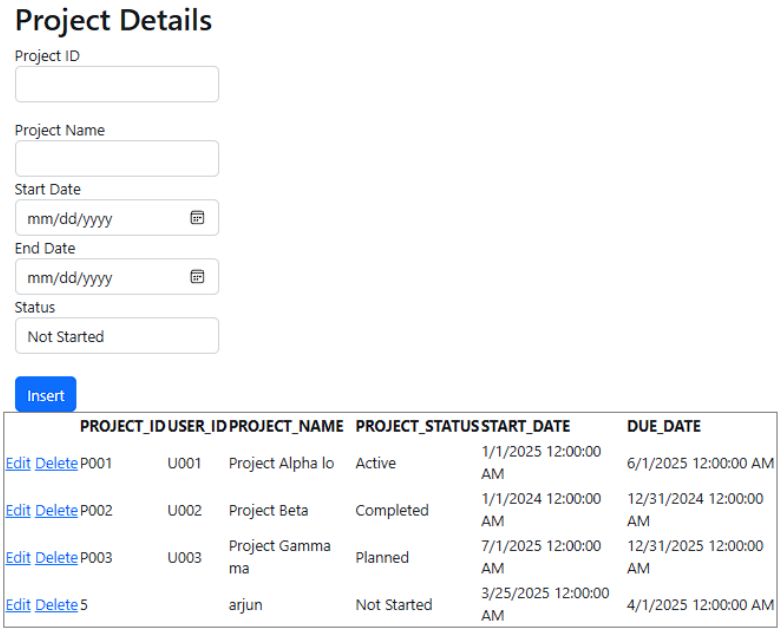
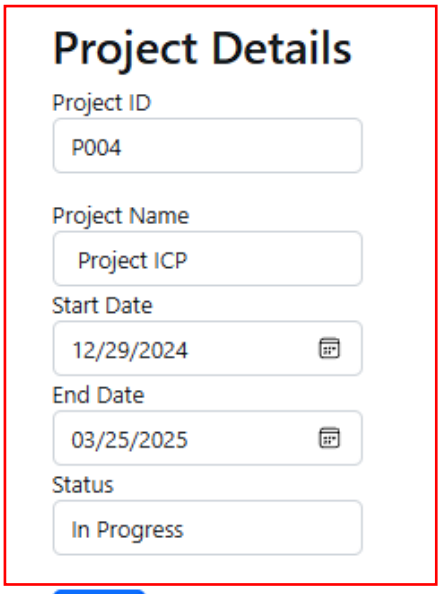


Figure 44: Updated Project Page

Adding New Project:



Insert

	PROJECT_ID	USER_ID	PROJECT_NAME	PROJECT_STATUS	START_DATE	DUE_DATE
Edit Delete	P001	U001	Project Alpha lo	Active	1/1/2025 12:00:00 AM	6/1/2025 12:00:00 AM
Edit Delete	P002	U002	Project Beta	Completed	1/1/2024 12:00:00 AM	12/31/2024 12:00:00 AM
Edit Delete	P003	U003	Project Gamma ma	Planned	7/1/2025 12:00:00 AM	12/31/2025 12:00:00 AM
Edit Delete	5		arjun	Not Started	3/25/2025 12:00:00 AM	4/1/2025 12:00:00 AM

Figure 45: Adding new Project

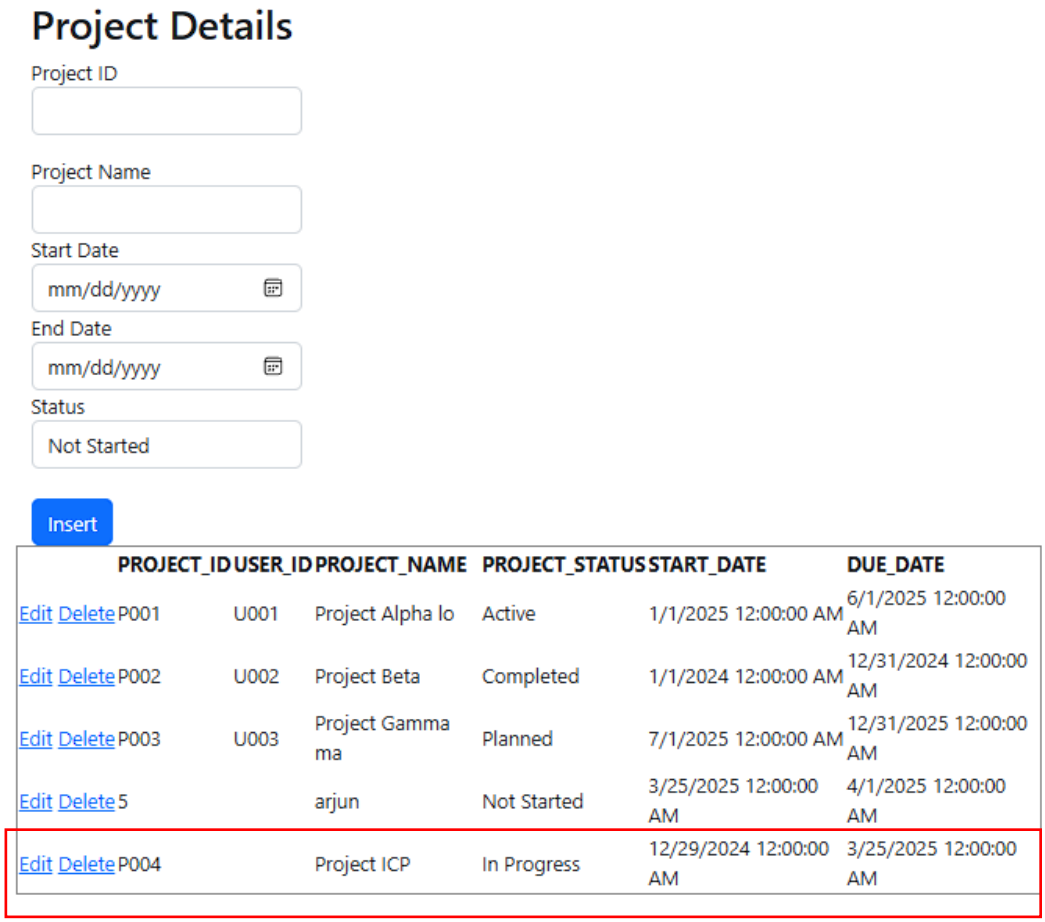


Figure 46: New Project Successfully Added

Table 13: Testcase Adding new project into the system

Objective	To inserting a new project into the system byt entering project details.
Action	Enter all the required project details and click insert
Expected Result	System should add a new project successfully and display it on the project list
Actual Result	The project details were successfully added and displayed in the Project list
Conclusion	Adding New Project was Successfully executed.

Updating the Project:

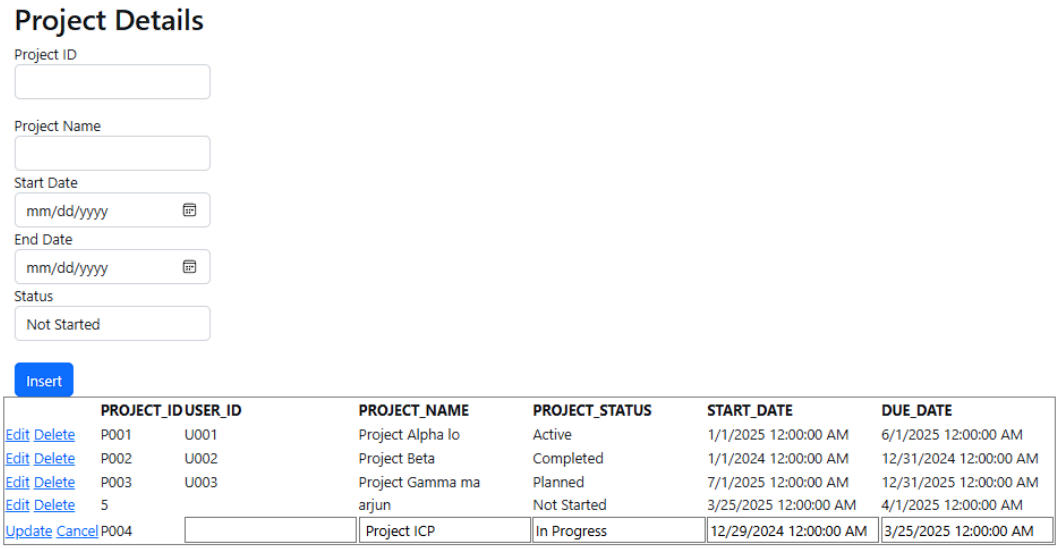


Figure 47: Updating the Project List

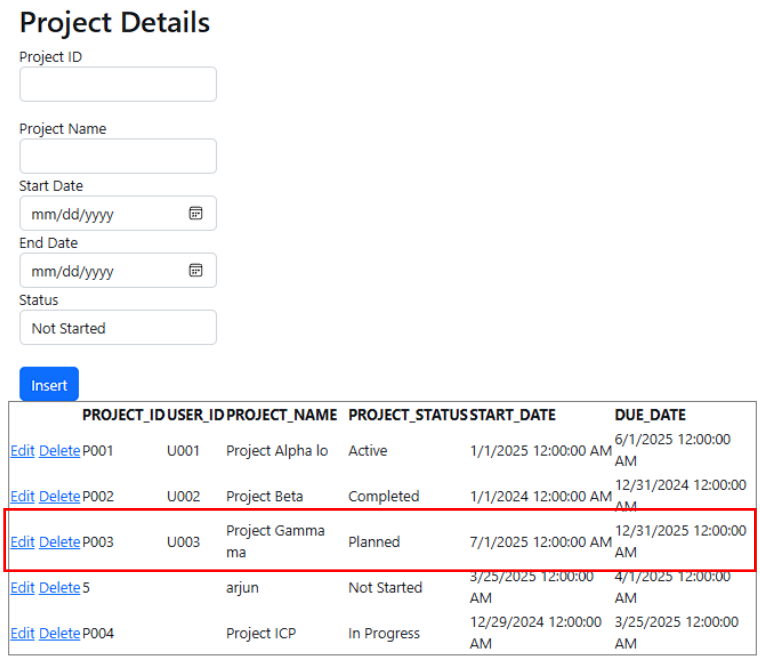


Figure 48: before Updating the Project Details

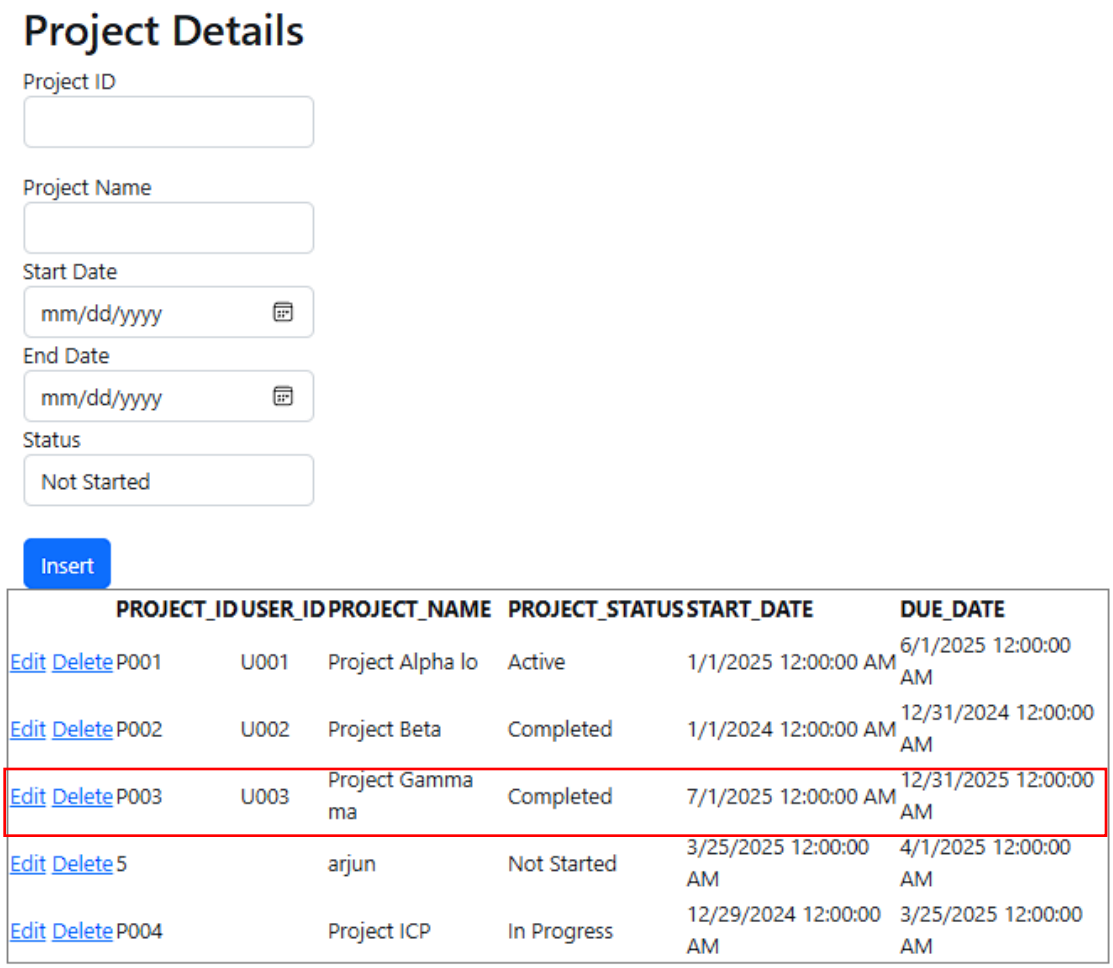


Figure 49: After Updating the Project details

Table 14: Testcase of Updating Project

Objective	To update project details into the system
Action	Edit and update the project information
Expected Result	System should update the field where user have updated
Actual Result	The project status was updated as shown in the figure above
Conclusion	Successfully executed.

Deleting the Project:

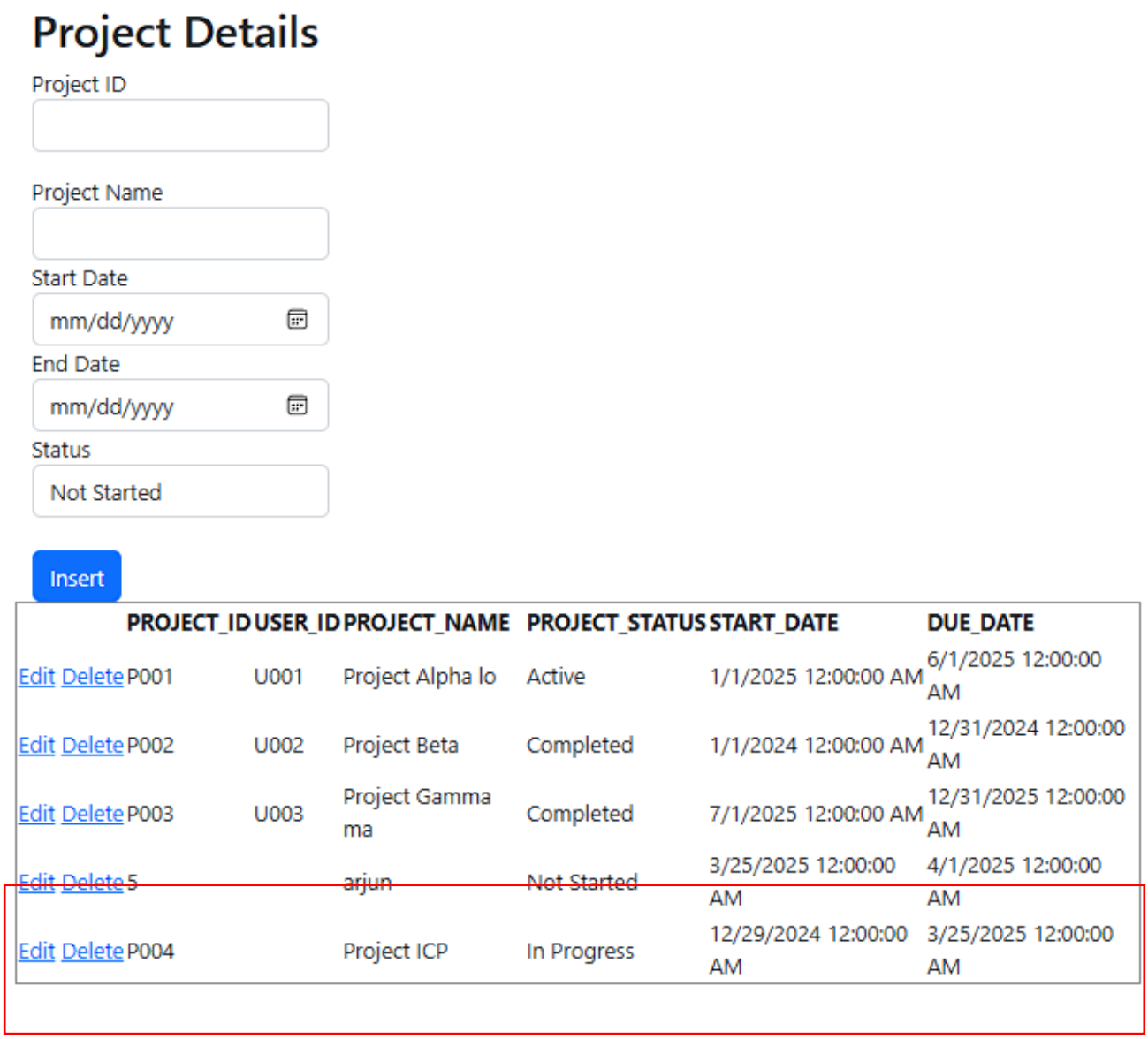


Figure 50: Before deleting the project:

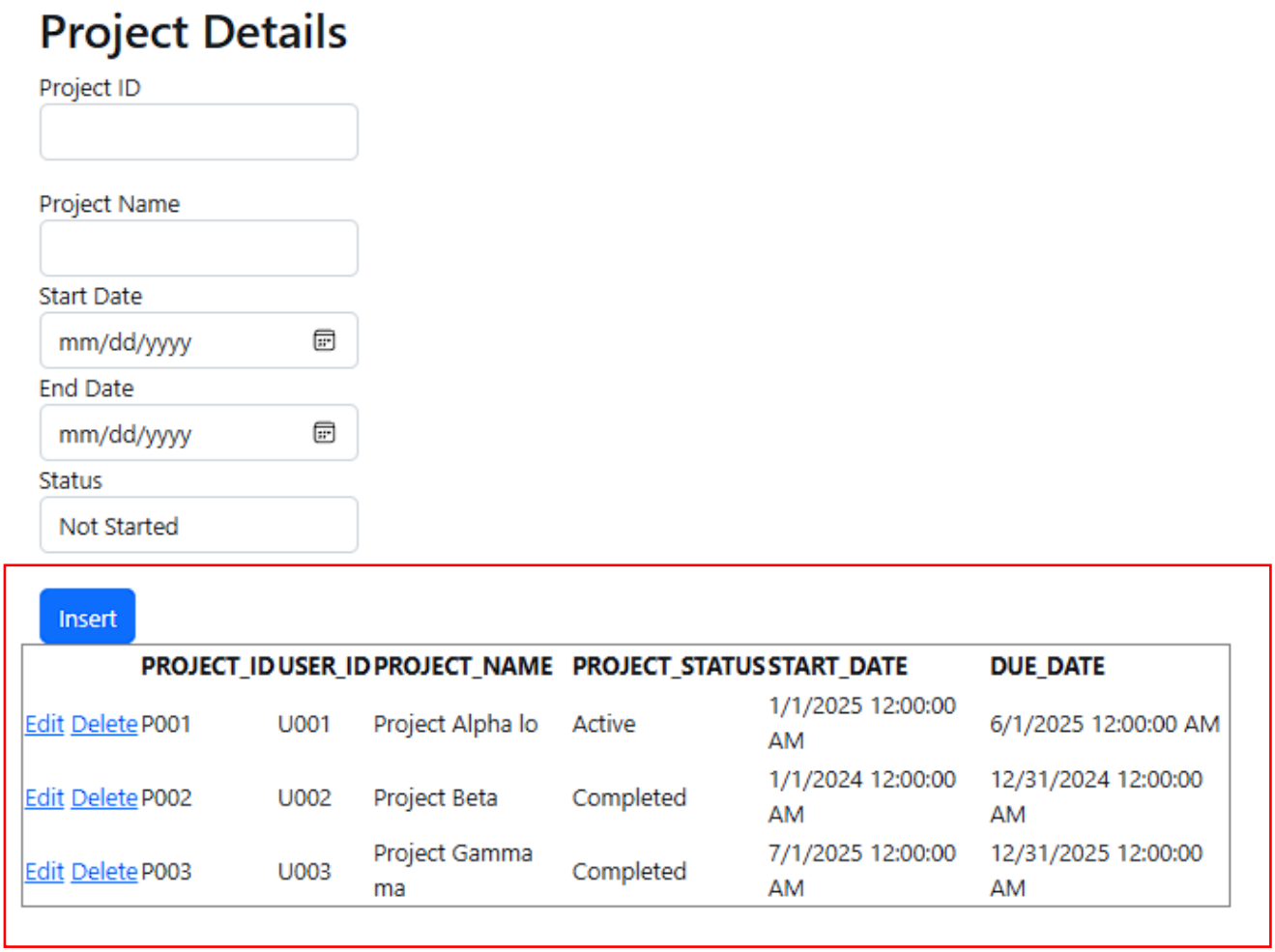


Figure 51: After Deleting the Project

Table 15: Testcasees of deleting the project

Objective	To delete a project from the project list
Action	Clicking on the delete option and deleting project from the project list.
Expected Result	Project should be deleted from the user list
Actual Result	Two newly added Projects were deleted as shown in the figure above.
Conclusion	Successfully executed.

8.3 SubTask:

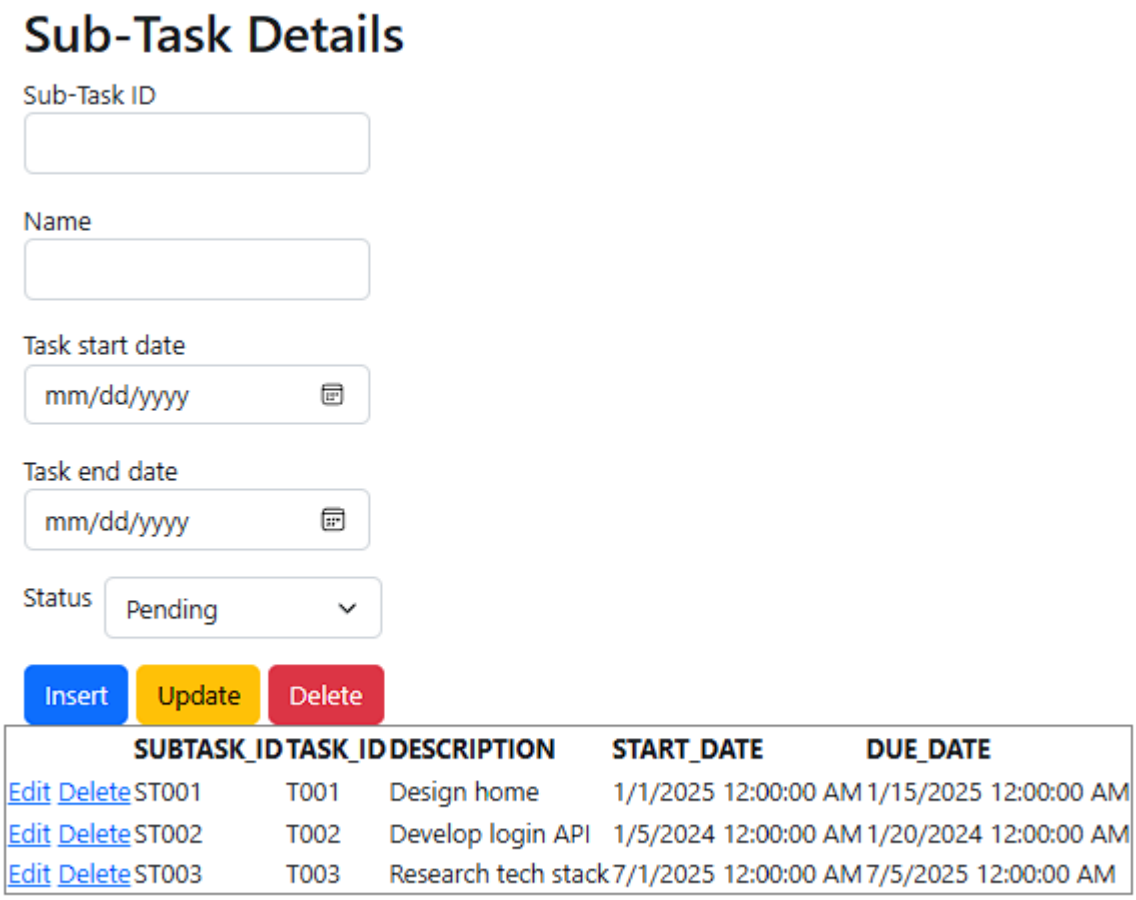


Figure 52 Sub Task Page:

Adding New SubTask:

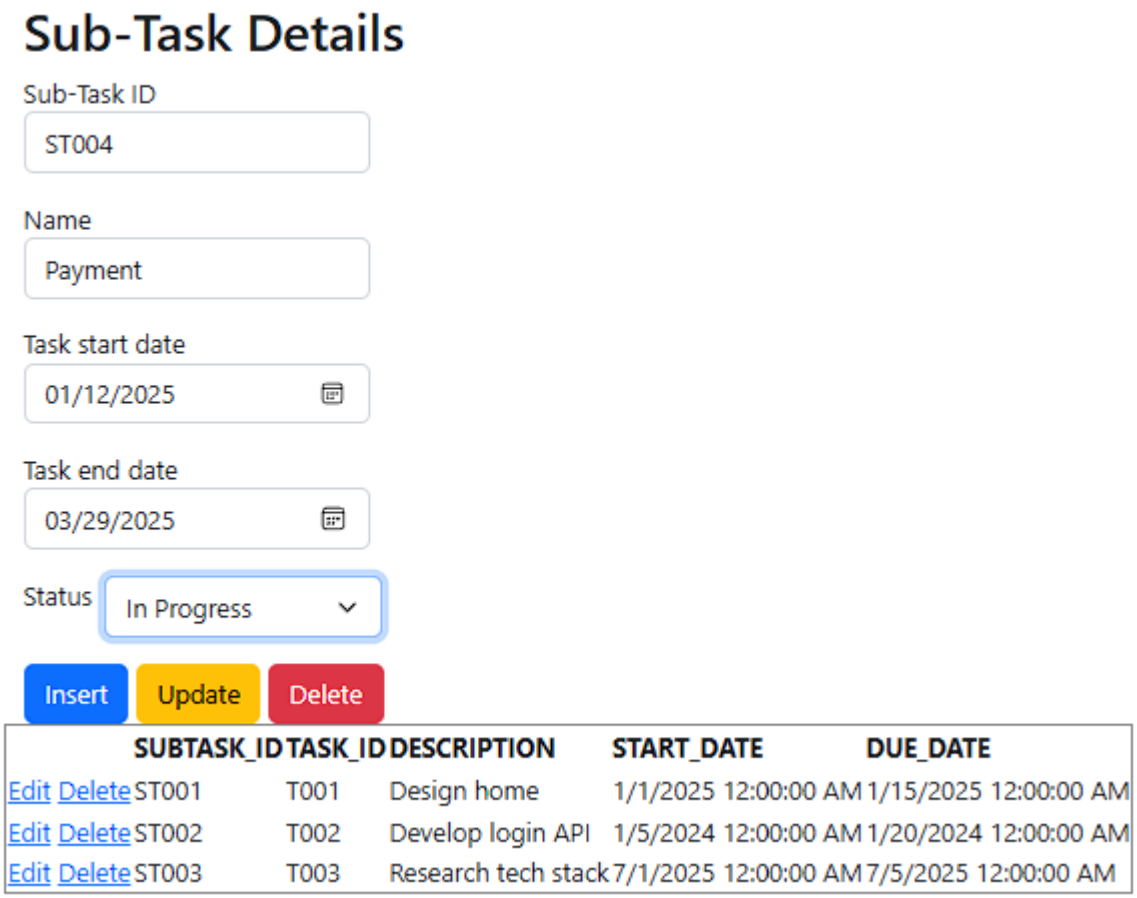


Figure 53: Inserting Values in sub task

Updating Sub Task:

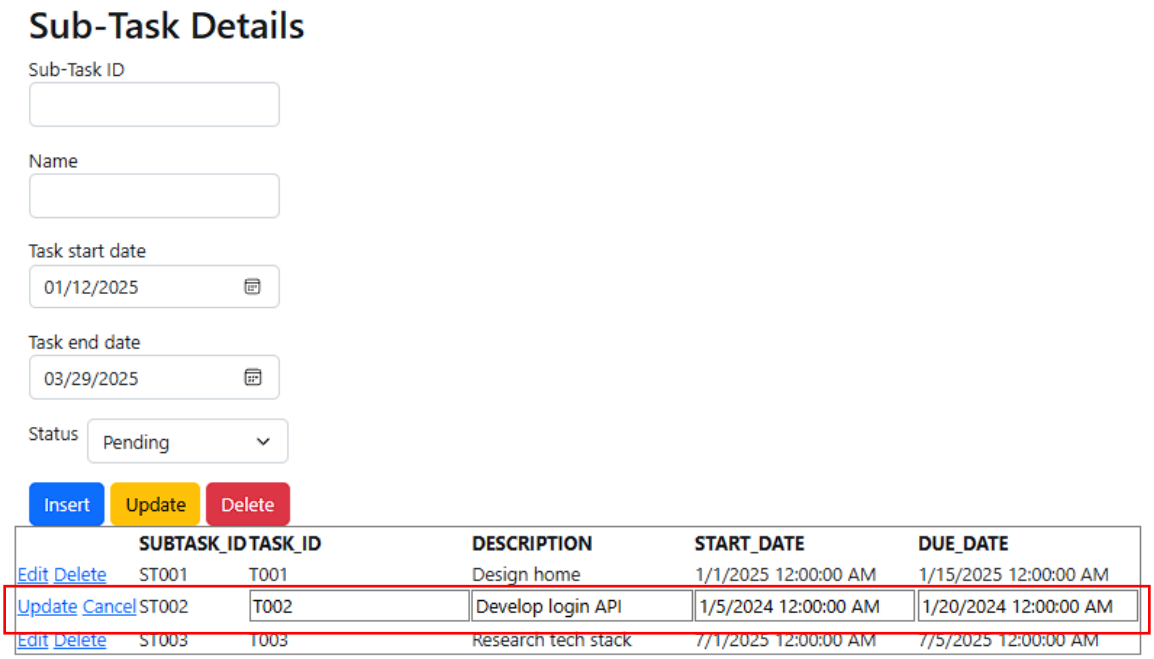


Figure 54: Before Updating the subtask details

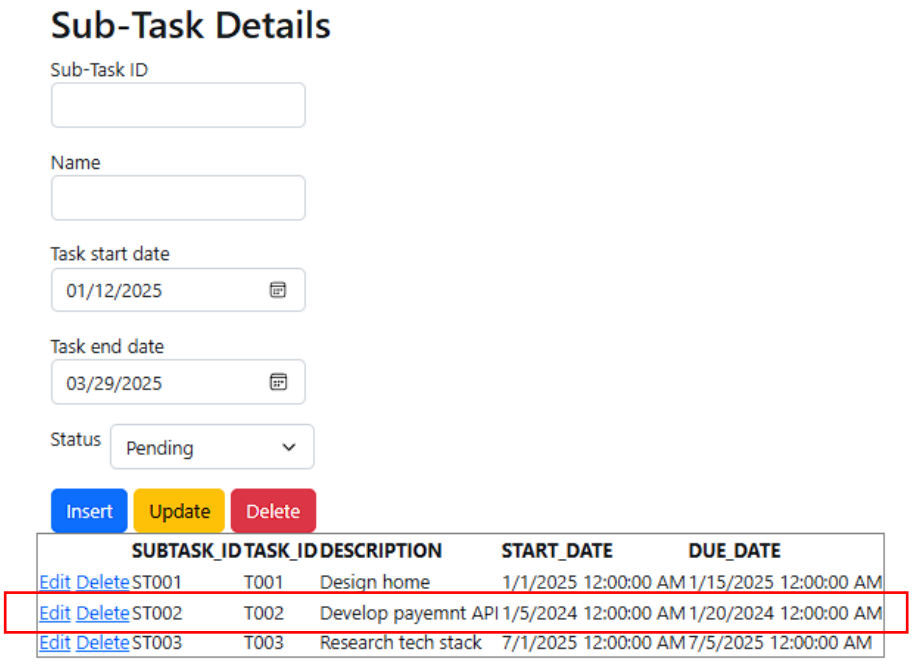


Figure 55: After Updating the sub task details

Table 16: Testcase Updating subtask

Objective	To update sub-task details into the system
Action	Edit and update the sub-task information
Expected Result	System should update the field where user have updated
Actual Result	The sub-task field was updated as shown in the figure above
Conclusion	Successfully executed.

Deleting subtask:

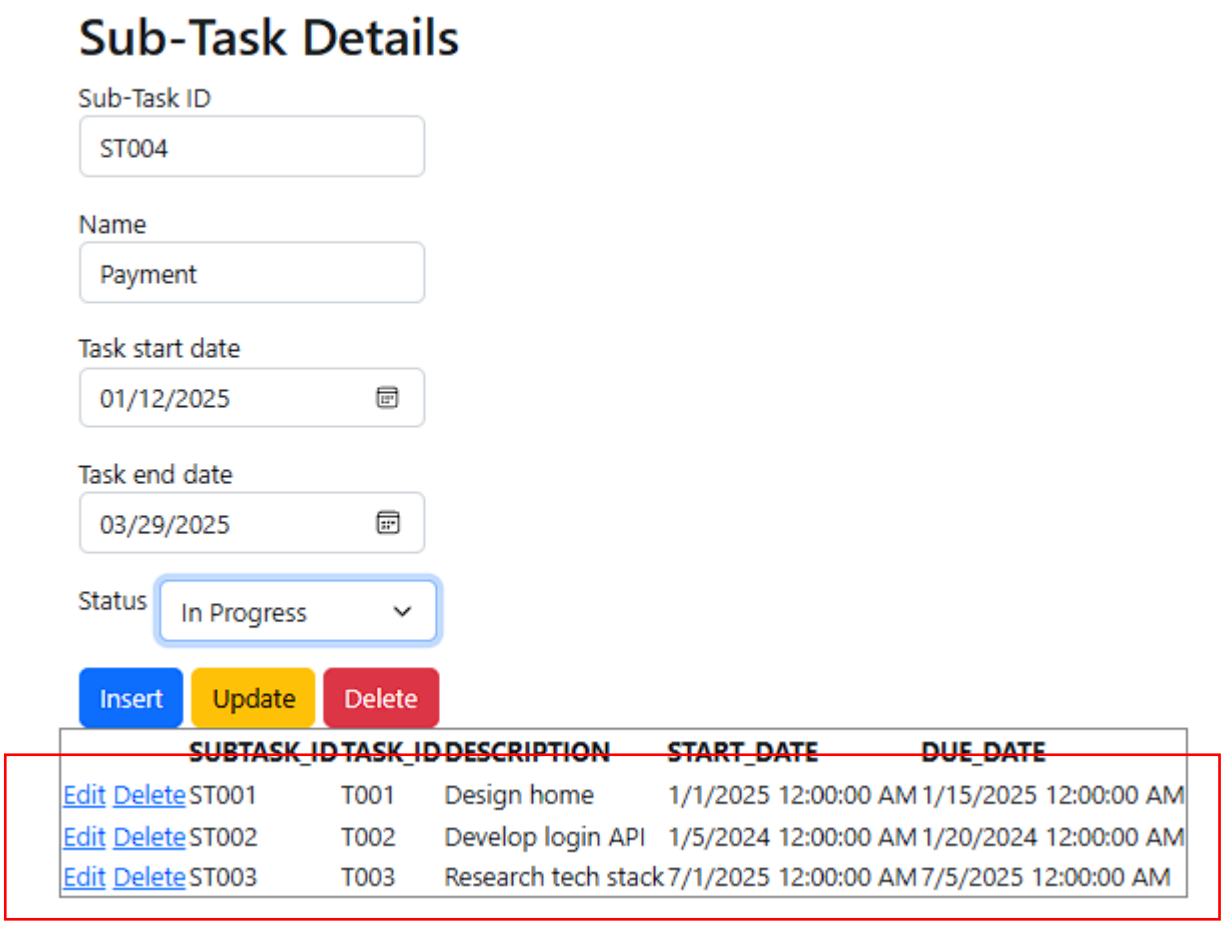


Figure 56: before Deleting subtask user

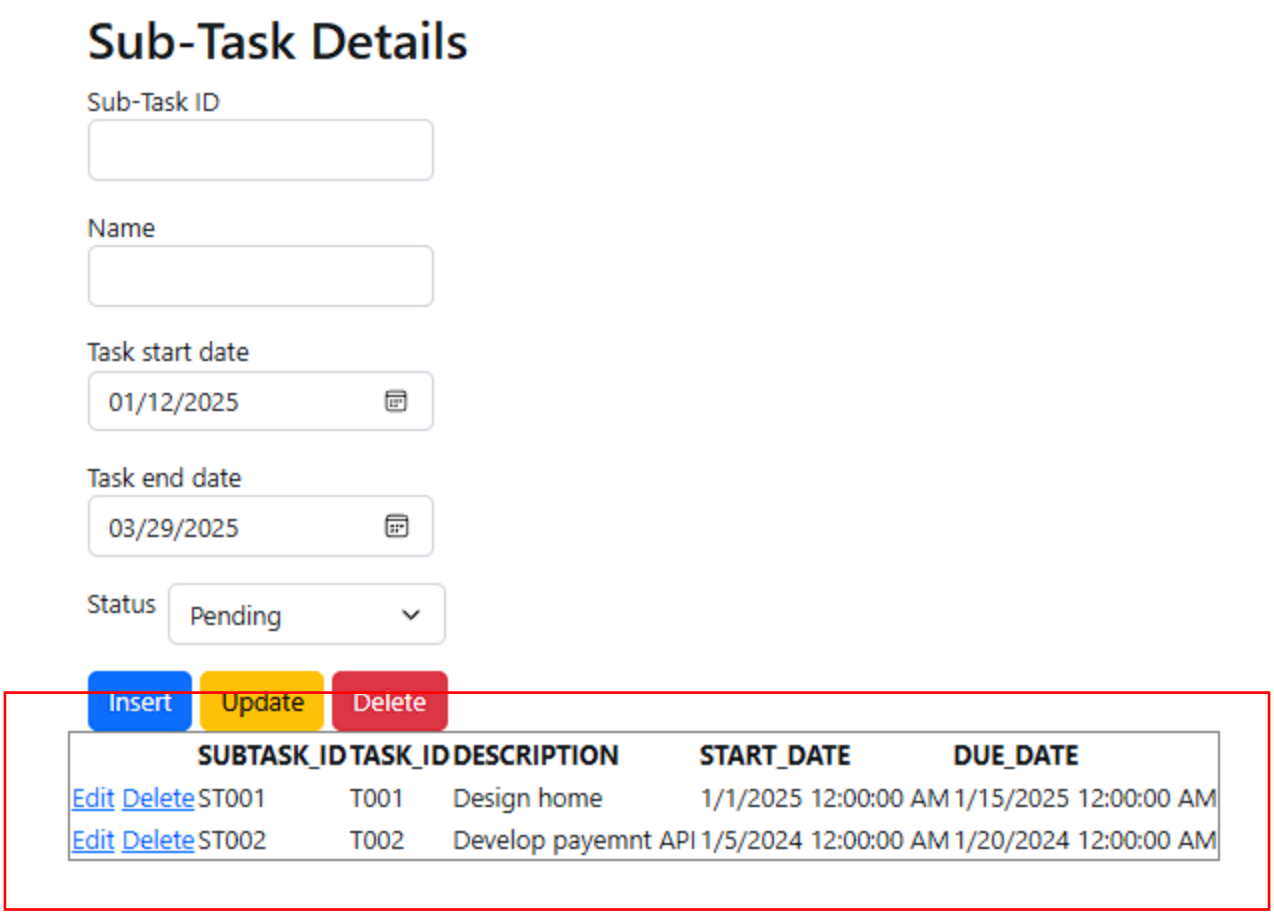


Figure 57: After deleting the subtask details

Table 17: Test cases of deleting subtasks

Objective	To delete a subtask from the subtask list
Action	Clicking on the delete option and deleting subtask from the subtask list.
Expected Result	Subtask should be deleted from the user list
Actual Result	Sub Task was deleted as shown in the figure above.
Conclusion	Successfully executed.

9. User Manual:

The Ls Cooperation website features a navigation bar that lets users easily access various pages like Users, Projects, Tasks, Subtasks, and more. Each page includes fields to fill out and an "Insert" button to add details to the list. Users have options to edit, update, insert, or delete information from the lists on these pages. Additionally, there's a more advanced web form that highlights the top performers.

10. Further discussion:

While developing the project management system, I faced several challenges that needed careful solutions. One of the biggest issues was improving the user experience with web forms, especially when users interacted with different types of data like projects, tasks, and subtasks. At first, the navigation and data handling were complicated, so I focused on improving form validation and making it easier to move between pages. By setting clear input rules and giving helpful feedback, I made the system more user-friendly and easier to use.

Another challenge was managing the complex relationships between users, projects, and tasks. This made database design difficult, especially when creating tables to connect data and keep it consistent. To solve this, I carefully organized tables to reduce unnecessary data and store everything correctly. I also added new tables like userProjects and projectRoles to better manage user assignments and roles within projects, which helped improve data handling.

I also had problems with using Oracle as my database management system. A major issue was the performance of queries, especially when dealing with large datasets or complex joins. Oracle's built-in optimizations didn’t always work well, which caused slow queries. To fix this, I optimized the database by adding the right indexes and changing how the queries were written. I also had issues with Oracle’s transaction and locking systems, which could lead to inconsistent data when multiple operations happened at the same time. I solved this by improving how transactions were managed and ensuring proper data handling.

Lastly, as the system grew, performance and scalability became concerns. The system started slowing down as more data was added. To fix this, I optimized database queries, added indexes,

and used techniques like pagination and lazy loading for large datasets. These changes helped improve performance and allowed the system to handle more data without slowing down.

References

Anon., 2024. [Online] Available at: https://coresignal.com/blog/data-normalization/ [Accessed 02 March 2025].

Belcic, I., 2024. [Online] Available at: https://www.ibm.com/think/topics/entity-relationship-diagram [Accessed 02 March 2025].

Chai, W., 2019. [Online] Available at: https://www.techtarget.com/searchapparchitecture/definition/data-dictionary [Accessed 12 March 2025].

Fayard, M., 2025. [Online] Available at: https://www.datacamp.com/tutorial/first-normal-form [Accessed 12 March 2025].

Kolli, D., 2024. [Online] Available at: https://medium.com/@dinesh.kolli2000/second-normal-form-2nf-85c2108f04e5 [Accessed 12 March 2024].

Kozubek, A., 2020. [Online] Available at: https://vertabelo.com/blog/normalization-1nf-2nf-3nf/ [Accessed 12 March 2025].