

Task Manager

Table of Contents

Table of Contents	2
Introduction	3
System Requirements	4
Context Diagram	4
Interface Requirements	5
Functional Requirements	9
Nonfunctional Requirements	16
Conceptual design of the database	17
ER Diagram	17
Business Rules and Integrity Constraints	17
Logical Database Schema	18
Database Schema	18
SQL Operations	18
Functional Dependencies and Normalization	20
The Database System	20
Creating Tables (MySQL CLI)	20
Inserting Value Into The Tables	21
Suggestions on Database Tuning	23
Additional queries and views	24
User Application Interface	26
Index.php	26
Dashboard Page	29
Associated SQL	30
Board Page	31
Conclusions and Future Work	34
Conclusions	34
Future Work	35
References	35

Introduction

Task manager is an application that lets users manage and track their tasks online in web-based software. Users will be able to create tasks and mark them as completed or

not completed similar to a to-do list. Each user will have several “boards” they can post their tasks to. These “boards” and any changes to them will be persisted in the database. Our goal for this project is to create an application that makes it easier to see and track your tasks.

The following report outlines several details about the implementation and design of our system to create the application described above.

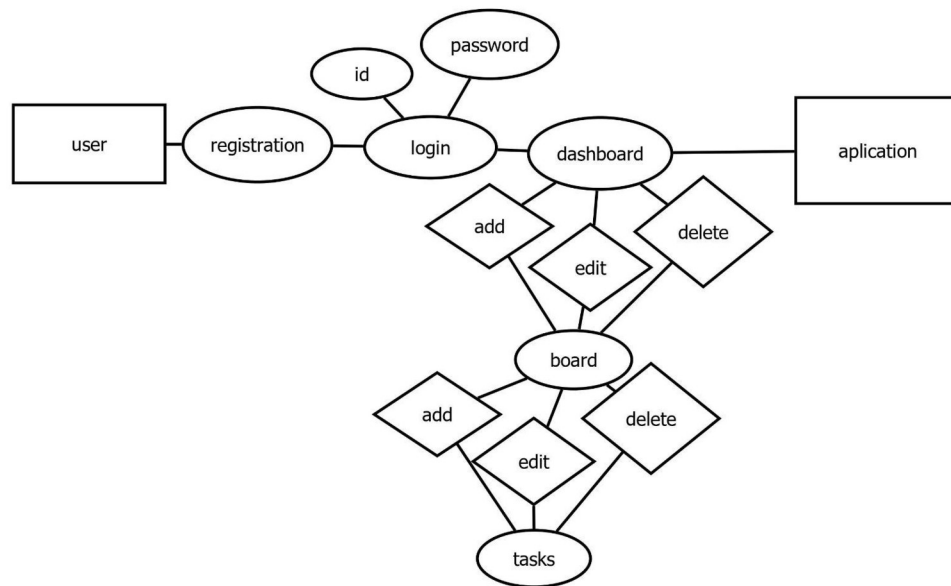
We start by going over the system requirements by providing a context diagram as well as a list of functional and non-functional requirements of the database system.

From there, we provide an overview of the design of the database, starting with its conceptual design all the way to the complete implementation of the database along with additional queries and views.

Finally, we go over the front-end application of our system and how it implements the SQL statements that were prepared in the previous sections as well give feedback on the project and a brief description describing future work.

System Requirements

Context Diagram



Interface Requirements

The images do not reflect our final project.

Login Page

Login

Register

When you first open the page, 2 boxes show up and you choose whether you have an account or not.

Dashboard Page

Account ▾

Dashboard

Board Name - Feb 2022 Edit

Task Title In Progress

Description: info
Date added: 02/06/2022

Task Title done

Description: info
Date added: 02/07/2022

Task Title done

Description: info
Date added: 02/08/2022

Task Title done

Description: info
Date added: 02/09/2022

Board Name - Mar 2022 Edit

Task Title In Progress

Description: info
Date added: 03/06/2022

Task Title done

Description: info
Date added: 03/07/2022

Task Title done

Description: info
Date added: 03/08/2022

Task Title done

Description: info
Date added: 03/09/2022

Board Name - Apr 2022 Edit

Task Title In Progress

Description: info
Date added: 04/06/2022

Task Title done

Description: info
Date added: 04/07/2022

Task Title done

Description: info
Date added: 04/08/2022

Board Name - May 2022 Edit

Task Title In Progress

Description: info
Date added: 05/06/2022

Task Title done

Description: info
Date added: 05/07/2022

Add Board

Add Board

Add Board (Modal)

Board Name

User Input

Create

Cancel

Board Page

Board Name - Feb 2022

Edit Board

Task Title

Description: info

Date added: 12/04/2021

Edit

mark as complete

Task Title

Description: info

Date added: 12/04/2021

Edit

done

Task Title

Description: info

Date added: 12/04/2021

Edit

done

Task Title

Description: info

Date added: 12/04/2021

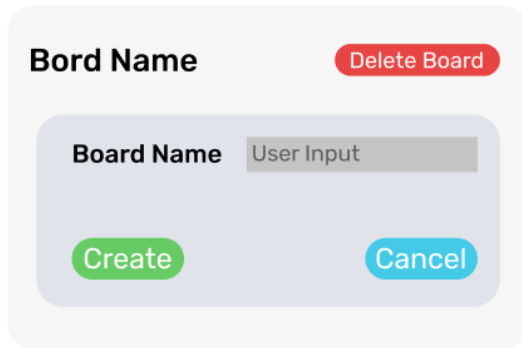
Edit

done

Add Task

Edit Board

Edit Board (Modal)



The modal has a light gray background. At the top left, the text "Bord Name" is displayed. At the top right, there is a red button labeled "Delete Board". Below this, a light blue rounded rectangle contains the form fields. Inside this rectangle, the text "Board Name" is followed by a text input field containing the placeholder "User Input". At the bottom of the light blue rectangle, there are two buttons: a green "Create" button on the left and a blue "Cancel" button on the right.

Add Task

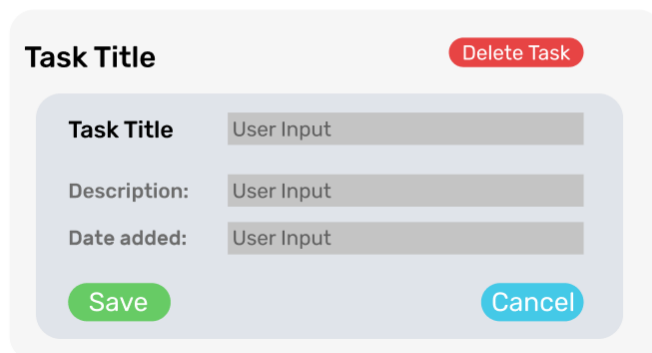
Add Task



The modal has a light gray background. It contains a light blue rounded rectangle with form fields. Inside, the text "Task Title" is followed by a text input field with the placeholder "User Input". Below this, the text "Description:" is followed by another text input field with the placeholder "User Input". Then, the text "Date added:" is followed by a third text input field with the placeholder "User Input". At the bottom of the light blue rectangle, there are two buttons: a green "Create" button on the left and a blue "Cancel" button on the right.

Edit Task

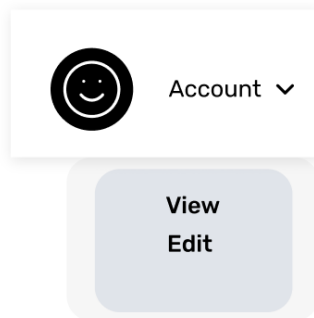
Edit Task



The modal has a light gray background. At the top left, the text "Task Title" is displayed. At the top right, there is a red button labeled "Delete Task". Below this, a light blue rounded rectangle contains the form fields. Inside this rectangle, the text "Task Title" is followed by a text input field with the placeholder "User Input". Below this, the text "Description:" is followed by another text input field with the placeholder "User Input". Then, the text "Date added:" is followed by a third text input field with the placeholder "User Input". At the bottom of the light blue rectangle, there are two buttons: a green "Save" button on the left and a blue "Cancel" button on the right.

My Account

My Account (Modal)



My Account View Page

My Account

FirstName LastName

First Name

Last Name

Email Address

My Account Edit Page

My Account

FirstName LastName [Delete Account](#)

First Name (User Input)

Last Name (User Input)

Email (User Input)

Password (User Input) [Verify](#)

[Save](#) [Cancel](#)

Functional Requirements

Registration

- **Positive User Flow**
 - When a user clicks on the “Registration” link on the home page, the registration form page is displayed. On this screen, the user will enter information related to their account.
 - Once the user submits the registration form, our system will create an entry for that user in our database.
 - Once the user is created in our database, the user should be redirected to their dashboard page.
- **Exceptions**
 - E001> The user inputs an invalid email address
 - Respond by telling the user the email address is not valid and highlight the invalid field on the form.
 - E002> The user inputs a password that does not meet our criteria
 - Respond by telling the user the password is invalid and list the rules they must follow to generate a valid password
 - E003> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **Post**
 - email
 - password

Login

- **Positive User Flow**
 - When a user clicks the “Login” link on the home page, a login form page will be displayed. On that screen, the user will be asked to enter their email and password.
 - Once the user enters their information and clicks the “Login” button, they will be redirected to their dashboard.
- **Exceptions**
 - E001> The user inputs the wrong password or email
 - Inform the user the information they’ve entered is not found on our database.
 - E002> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code

- **Data Dictionary**
 - **Post**
 - email
 - password

View Dashboard

- **Positive User Flow**
 - The user navigates to their dashboard and is presented with a list of cards of boards they've created in the past, with each card element representing one board.
 - These boards should be sorted by the last date they were interacted with, with the latest interactions first in the list.
 - A button should be shown to the user to add a new board and there should also be a button on each card to delete that board as well.
- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user "Something went wrong, please try again" with a 500 status code
- **Data Dictionary**
 - **Get**
 - User's boards

Forgot Password

- **Positive User Flow**
 - The user clicks on the "Forgot Password" link on the login page.
 - The user is navigated to a form where they can input their email address.
 - The user will receive an email with a reset link.
 - Upon clicking the link, the user will be directed to a page with a form where they can input their new password and confirm it.
 - When the user clicks "Submit" on that form, their password will be updated in the database and they will be shown a page that tells them their update was successful.
 - On that same page, there will be a link to go back to the login page.
- **Exceptions**
 - E001> The email does not exist
 - Tell the user no such email exists in our database
 - E002> The token generated for the password reset link has expired
 - Tell the user their token has expired and they should try again
 - E003> An error occurs on the backend of our system
 - Respond by telling the user "Something went wrong, please try again" with a 500 status code

- **Data Dictionary**
 - **Post**
 - User email
 - New password

Edit Board

- **Positive User Flow**
 - When the user is on the dashboard page after logging in, they can click on the edit button associated with a board on their dashboard.
 - Once that button is clicked, a modal will appear for the user to edit information.
 - This modal will contain a form with an input field for the user to modify the name of the board.
 - The modal will also have two buttons, one to close the modal and one to submit the form.
 - Once the form is submitted, the board is modified in the database and the page is reloaded.
- **Exceptions**
 - E001> The user inputs nothing for the name of the board
 - Replace the empty field with “Untitled” when processing the update
 - E002> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **Post**
 - User’s boards

Delete Board

- **Positive User Flow**
 - When the user has the “Edit Board” modal open, there should be a button to delete the board they are editing.
 - When the button is clicked, the board will be removed from the database and the dashboard page should reload to reflect the change in data.
- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **DELETE**
 - Current Board

View Board

- **Positive User Flow**

- When a user clicks on a board in their dashboard, they will be directed to another page to view their board.
- On their board, the user should be able to see the name of the board at the top, as well as all tasks associated with that board in the main portion of the screen.
- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **GET**
 - Current Board
 - Current board’s tasks

Add Board

- **Positive User Flow**
 - When the user is on the dashboard page, they will be able to click the “Add Board” card to create a new board.
 - When this card is clicked, a modal will pop up for them to input the information of the board. This modal will have a cancel and create button as well.
 - When the create button is clicked, a new board should be created and the user should be redirected to that board’s page.
 - If the cancel button is clicked, remove the modal and return the user to the dashboard page.
- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **POST**
 - Board Name

Add Task

- **Positive User Flow**
 - When the user is on the board page, they will be able to click the “Add Task” button to create a new task.
 - When this button is clicked, a new container will create on the board page for them to enter the information of the task.
 - When the create button is clicked, a new task should be created and the user should be enter information about that task.

- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **POST**
 - Board Name
 - Task Name

View Task

- **Positive User Flow**
 - When the user is on the board page, they will be able to see the “Task” cards.
 - The tasks should be visible on Board page for the user, and they can check whether that task is completed or not.
- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **GET**
 - Board’s Tasks

Edit Task

- **Positive User Flow**
 - When the user is on the board page, they will be able to click the “Task card” to edit the task.
 - When the task is clicked, they are able to edit the task’s detail.
 - After changing the information, there is a button to save the new data.
- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **POST**
 - Task Name

Mark task as complete/not complete

- **Positive User Flow**
 - When the user is on the board page, they will be able to click the “Mark as complete” button to mark the task as completed.
 - The default setting for each task is marked as not completed.
 - That “mark as complete” button should be visible on the task card.

- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **POST**
 - Task’s completion status

Delete Task

- **Positive User Flow**
 - When the user is on the board page, they will be able to click the “Delete Task” button to delete the task.
- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **DELETE**
 - Current Task

View Account

- **Positive User Flow**
 - There is a button at top of the page called “My Account”.
 - When the user clicks on the button, a menu should be open and there are multiple button is under the menu such as View, Edit, etc.
 - The account information page will open whenever the View button is clicked.
- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **GET**
 - User Email

Edit Account

- **Positive User Flow**
 - There is a button called Edit under My account menu.
 - The user will redirect to a new page whenever the View or Edit buttons are clicked.
- **Exceptions**
 - E001> An error occurs on the backend of our system

- Respond by telling the user “Something went wrong, please try again” with a 500 status code
 - E002> User email is already taken
 - Respond by telling user the email is taken
 - E003> User password doesn’t match criteria for valid passwords
 - Respond by telling user how to generate a valid password
- **Data Dictionary**
 - **POST**
 - User Email
 - User Password

Delete Account

- **Positive User Flow**
 - The Delete Account button would be visible to the user after they click on the Edit button under My account menu.
 - There is a confirmation text captcha whenever the user needs to delete the account.
 - All tasks and boards will be deleted after deleting the account.
 - The user is redirected to the login page after the deletion takes place.
- **Exceptions**
 - E001> An error occurs on the backend of our system
 - Respond by telling the user “Something went wrong, please try again” with a 500 status code
- **Data Dictionary**
 - **DELETE**
 - User

Nonfunctional Requirements

Organized

The view should be easy to look at and could easily track what needs to be done for a certain project or just a day in a life.

Accessible

The final product could be used anywhere in the world whether you’re on the other side of the world or across the country.

Easy To Use

When a first user starts using our product, they should be able to do what they need without problems or confusion.

Fast Response Time

The response time of our database/application should be fast enough to offer the end user a pleasant and non frustrating experience. If the response time is too slow, then the user will have trouble getting the work they wish to accomplish.

Maintainability

Keeping high maintainability is crucial as long or unexpected maintenance periods could frustrate our users and could drastically increase the cost of maintaining the application.

Scalability

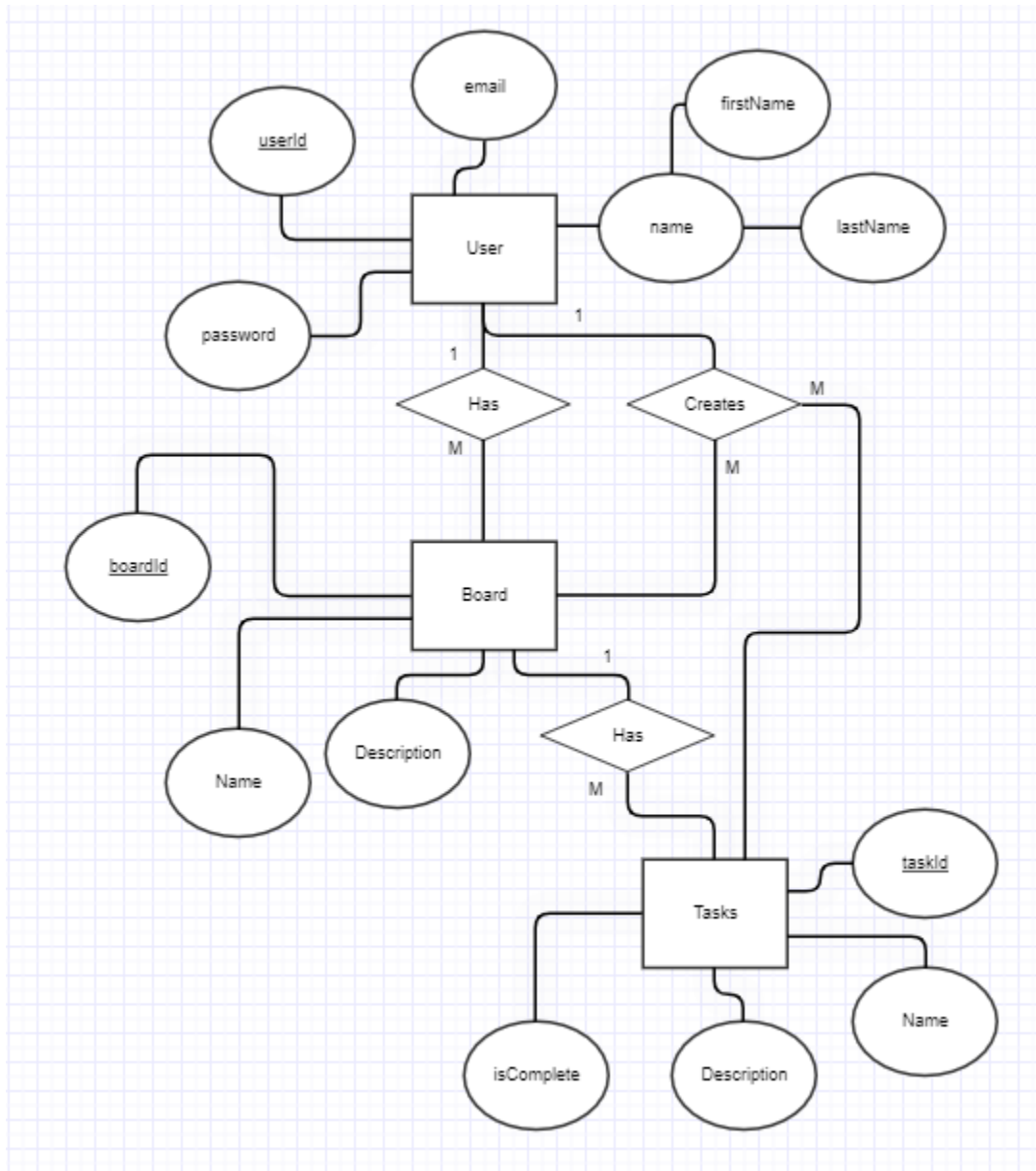
In order to support the continuous increase in our user base, we need to ensure that our database is able to keep up with the increase in workload.

Security

Security needs to be the number one concern as any security breach/ flaw is not only a massive blunder, but an issue that could cause massive harm to our user base.

Conceptual design of the database

ER Diagram



Business Rules and Integrity Constraints

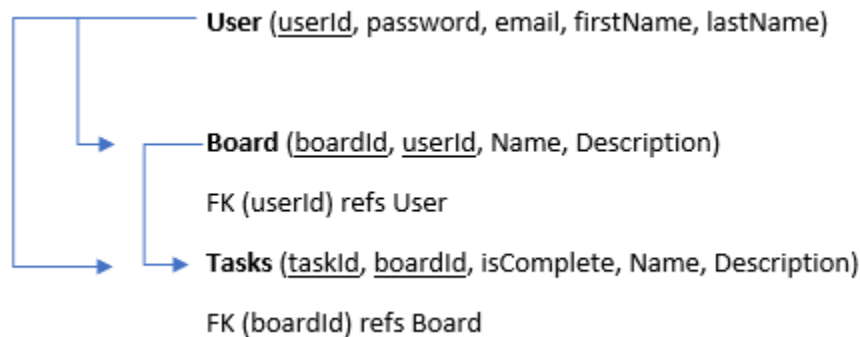
- All Primary keys are unique: To ensure that all primary keys serve their purpose in identifying relations, they must be unique to avoid one key being linked to multiple tuples inside a table. This includes `userid`, `boardId`, and `taskId`.
- Email/password verification: We need to ensure that the user gives us a real email and a password that satisfies our requirements. On the user side, the password would likely have to include characters from multiple sources (lowercase letters, uppercase letters, numbers, etc...) and that would then be hashed and sent to our database to be stored securely.
- Foreign keys must have an associated primary key: In order for our tables to be properly linked, we need to have each foreign key be linked to a primary key from another table.

In this case, a user tuple should have 1 or more boardIds and taskIds so that the user in that tuple can be properly linked to those boards/tasks that it has access to.

- Values cannot be Null: certain values in our table cannot be null, this includes all primary keys, user's name, email, and password to name a few that cannot be null.
- One to Many: The number of relationships each tuple in a table has needs to be maintained according to the ER diagram. For example, users can have multiple boards/tasks, but a board or task can only have one user associated with it.

Logical Database Schema

Database Schema



SQL Operations

Creating User Table

```

CREATE TABLE task_manager_db.user(
  UserId INT NOT NULL AUTO_INCREMENT,
  Email VARCHAR(255) NOT NULL,
  Password VARCHAR(255) NOT NULL,
  FirstName VARCHAR(255) NOT NULL,
  LastName VARCHAR(255) NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (UserId));
  
```

Inserting Into User Table

```

INSERT INTO User( Password, Email, FirstName, LastName) VALUES ("password",
"test@email.com", "John", "Smith");
  
```

Creating Board Table

```

CREATE TABLE task_manager_db.board(
  BoardId INT NOT NULL AUTO_INCREMENT,
  
```

```

UserId INT NOT NULL,
Name VARCHAR(255) NOT NULL,
Description VARCHAR(1000),
created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY (BoardId),
INDEX UserId_idx (UserId ASC) VISIBLE,
CONSTRAINT UserId FOREIGN KEY (UserId) REFERENCES
task_manager_db.user(UserId) ON DELETE CASCADE ON UPDATE CASCADE);

```

Inserting Into Board Table

```

INSERT INTO Board(UserId, Name, Description) VALUES ( "1", "Test", "Test Description");

```

Creating Task Table

```

CREATE TABLE task_manager_db.task(
TaskId INT NOT NULL AUTO_INCREMENT,
BoardId INT NOT NULL,
Name VARCHAR(255) NOT NULL,
Description VARCHAR(1000),
IsComplete BOOLEAN,
created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY (TaskId),
INDEX BoardId_idx (BoardId ASC) VISIBLE,
CONSTRAINT BoardId FOREIGN KEY (BoardId) REFERENCES
task_manager_db.board(BoardId) ON DELETE CASCADE ON UPDATE CASCADE);

```

Inserting Into Task Table

```

INSERT INTO Task(BoardId, Name, Description, IsComplete) VALUES ("1", "Test", "Test
Description", false);

```

Functional Dependencies and Normalization

1. User table:
 - a. {userID}->{password, email, firstname, lastname}
 - b. This table is fairly simple and is already in 3NF. It would violate 1NF if name was a composite value, but we already took that into account when designing our schema.
2. Board table:
 - a. {boardID}->{name, description}

- b. The table here is rather simple and only has one FD which determines the other attributes in the table. No normalization is needed for this table as it is already in 3NF.
 - c. Note: Redundant data would be present if multiple users had access to the same board. This would make boardID non unique and redundant tuples would be created for the same board for different users. However, as a part of our ER diagram, only one user has access to a board making boardID unique and preventing this problem.
3. Task table:
- a. {taskID}->{name, description, iscomplete}
 - b. The table here is rather simple and only has one FD which determines the other attributes in the table. No normalization is needed for this table as it is already in 3NF.
 - c. Note: same issue as above, just with boards and tasks.

The Database System

Creating Tables (MySQL CLI)

User Table

```
mysql> CREATE TABLE task_manager_db.user(
-> UserId INT NOT NULL AUTO_INCREMENT,
-> Email VARCHAR(255) NOT NULL,
-> Password VARCHAR(255) NOT NULL,
-> FirstName VARCHAR(255) NOT NULL,
-> LastName VARCHAR(255) NOT NULL,
-> created_at DATETIME DEFAULT CURRENT_TIMESTAMP, PRIMARY KEY (UserId));
Query OK, 0 rows affected (0.05 sec)

mysql> █
```

Board Table

```
mysql> CREATE TABLE task_manager_db.board(
-> BoardId INT NOT NULL AUTO_INCREMENT,
-> UserId INT NOT NULL,
-> Name VARCHAR(255) NOT NULL,
-> Description VARCHAR(1000),
-> created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
-> PRIMARY KEY (BoardId),
-> INDEX UserId idx (UserId ASC) VISIBLE,
-> CONSTRAINT UserId FOREIGN KEY (UserId) REFERENCES task_manager_db.user(UserId) ON DELETE CASCADE ON UPDATE CASCADE);
Query OK, 0 rows affected (0.07 sec)

mysql> █
```

Task Table

```
mysql> CREATE TABLE task_manager_db.task(
-> TaskId INT NOT NULL AUTO_INCREMENT,
-> BoardId INT NOT NULL,
-> Name VARCHAR(255) NOT NULL,
-> Description VARCHAR(1000),
-> IsComplete BOOLEAN,
-> created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
-> PRIMARY KEY (TaskId),
-> INDEX BoardId_idx (BoardId ASC) VISIBLE,
-> CONSTRAINT BoardId FOREIGN KEY (BoardId) REFERENCES task_manager_db.board(BoardId) ON DELETE CASCADE ON UPDATE CASCADE);
Query OK, 0 rows affected (0.06 sec)

mysql>
```

Inserting Value Into The Tables

User Table

```
mysql> INSERT INTO user(Email, Password, FirstName, LastName) VALUES
-> ("wsw190000@utdallas.edu", "123456", "Warren", "Wu"),
-> ("glm200000@utdallas.edu", "123456", "Gavin", "Moreland"),
-> ("thh190001@utdallas.edu", "123456", "Thach", "Huynh"),
-> ("axz172330@utdallas.edu", "123456", "Armin", "Ziaei"),
-> ("jgg190001@utdallas.edu", "123456", "Janet", "Gutierrez");
Query OK, 5 rows affected (0.01 sec)
Records: 5 Duplicates: 0 Warnings: 0
```

```
mysql> SELECT * FROM user;
```

UserId	Email	Password	FirstName	LastName	created_at
1	wsw190000@utdallas.edu	123456	Warren	Wu	2022-05-02 20:18:59
2	glm200000@utdallas.edu	123456	Gavin	Moreland	2022-05-02 20:18:59
3	thh190001@utdallas.edu	123456	Thach	Huynh	2022-05-02 20:18:59
4	axz172330@utdallas.edu	123456	Armin	Ziaei	2022-05-02 20:18:59
5	jgg190001@utdallas.edu	123456	Janet	Gutierrez	2022-05-02 20:18:59

5 rows in set (0.00 sec)

Board Table

```
mysql> INSERT INTO board(UserId, Name, Description) VALUES
-> ("1", "Proj_Phase_1", "Requirements Analysis"),
-> ("2", "Proj_Phase_1", "Requirements Analysis"),
-> ("3", "Proj_Phase_1", "Requirements Analysis"),
-> ("4", "Proj_Phase_1", "Requirements Analysis"),
-> ("5", "Proj_Phase_1", "Requirements Analysis"),
-> ("1", "Proj_Phase_2", "Conceptual and Logical Database Design"),
-> ("2", "Proj_Phase_2", "Conceptual and Logical Database Design"),
-> ("3", "Proj_Phase_2", "Conceptual and Logical Database Design"),
-> ("4", "Proj_Phase_2", "Conceptual and Logical Database Design"),
-> ("5", "Proj_Phase_2", "Conceptual and Logical Database Design"),
-> ("1", "Proj_Phase_3", "Normalization and Database Implementation and Testing"),
-> ("2", "Proj_Phase_3", "Normalization and Database Implementation and Testing"),
-> ("3", "Proj_Phase_3", "Normalization and Database Implementation and Testing"),
-> ("4", "Proj_Phase_3", "Normalization and Database Implementation and Testing"),
-> ("5", "Proj_Phase_3", "Normalization and Database Implementation and Testing");
Query OK, 15 rows affected (0.01 sec)
Records: 15  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM board;
```

BoardId	UserId	Name	Description	created_at
1	1	Proj_Phase_1	Requirements Analysis	2022-05-02 20:20:18
2	2	Proj_Phase_1	Requirements Analysis	2022-05-02 20:20:18
3	3	Proj_Phase_1	Requirements Analysis	2022-05-02 20:20:18
4	4	Proj_Phase_1	Requirements Analysis	2022-05-02 20:20:18
5	5	Proj_Phase_1	Requirements Analysis	2022-05-02 20:20:18
6	1	Proj_Phase_2	Conceptual and Logical Database Design	2022-05-02 20:20:18
7	2	Proj_Phase_2	Conceptual and Logical Database Design	2022-05-02 20:20:18
8	3	Proj_Phase_2	Conceptual and Logical Database Design	2022-05-02 20:20:18
9	4	Proj_Phase_2	Conceptual and Logical Database Design	2022-05-02 20:20:18
10	5	Proj_Phase_2	Conceptual and Logical Database Design	2022-05-02 20:20:18
11	1	Proj_Phase_3	Normalization and Database Implementation and Testing	2022-05-02 20:20:18
12	2	Proj_Phase_3	Normalization and Database Implementation and Testing	2022-05-02 20:20:18
13	3	Proj_Phase_3	Normalization and Database Implementation and Testing	2022-05-02 20:20:18
14	4	Proj_Phase_3	Normalization and Database Implementation and Testing	2022-05-02 20:20:18
15	5	Proj_Phase_3	Normalization and Database Implementation and Testing	2022-05-02 20:20:18

```
15 rows in set (0.00 sec)
```

Task Table

```
mysql> INSERT INTO task(BoardId, Name, Description, IsComplete) VALUES
-> ("1", "Proj Phase 1", "System description", true),
-> ("5", "Proj Phase 1", "Context diagram", true),
-> ("1", "Proj Phase 1", "Functional requirements", true),
-> ("4", "Proj Phase 1", "Functional requirements", true),
-> ("2", "Proj Phase 1", "Non-functional requirements", true),
-> ("3", "Proj Phase 1", "Non-functional requirements", true),
-> ("10", "Proj Phase 2", "ER Diagram", true),
-> ("6", "Proj Phase 2", "Database Schema", true),
-> ("7", "Proj Phase 2", "Business rules and integrity constraints", true),
-> ("8", "Proj Phase 2", "Business rules and integrity constraints", true),
-> ("9", "Proj Phase 2", "Interface requirements", true),
-> ("6", "Proj Phase 2", "Interface requirements", true),
-> ("12", "Proj Phase 3", "Normalization", true),
-> ("11", "Proj Phase 3", "Database Implementation", true),
-> ("14", "Proj Phase 3", "Testing", true),
-> ("13", "Proj Phase 3", "Suggestions", true),
-> ("15", "Proj Phase 3", "Additional queries", true);
Query OK, 17 rows affected (0.02 sec)
Records: 17 Duplicates: 0 Warnings: 0

mysql> SELECT * FROM task;
+-----+-----+-----+-----+-----+-----+
| TaskId | BoardId | Name | Description | IsComplete | created_at |
+-----+-----+-----+-----+-----+-----+
| 1 | 1 | Proj Phase 1 | System description | 1 | 2022-05-02 20:21:35 |
| 2 | 5 | Proj Phase 1 | Context diagram | 1 | 2022-05-02 20:21:35 |
| 3 | 1 | Proj Phase 1 | Functional requirements | 1 | 2022-05-02 20:21:35 |
| 4 | 4 | Proj Phase 1 | Functional requirements | 1 | 2022-05-02 20:21:35 |
| 5 | 2 | Proj Phase 1 | Non-functional requirements | 1 | 2022-05-02 20:21:35 |
| 6 | 3 | Proj Phase 1 | Non-functional requirements | 1 | 2022-05-02 20:21:35 |
| 7 | 10 | Proj Phase 2 | ER Diagram | 1 | 2022-05-02 20:21:35 |
| 8 | 6 | Proj Phase 2 | Database Schema | 1 | 2022-05-02 20:21:35 |
| 9 | 7 | Proj Phase 2 | Business rules and integrity constraints | 1 | 2022-05-02 20:21:35 |
| 10 | 8 | Proj Phase 2 | Business rules and integrity constraints | 1 | 2022-05-02 20:21:35 |
| 11 | 9 | Proj Phase 2 | Interface requirements | 1 | 2022-05-02 20:21:35 |
| 12 | 6 | Proj Phase 2 | Interface requirements | 1 | 2022-05-02 20:21:35 |
| 13 | 12 | Proj Phase 3 | Normalization | 1 | 2022-05-02 20:21:35 |
| 14 | 11 | Proj Phase 3 | Database Implementation | 1 | 2022-05-02 20:21:35 |
| 15 | 14 | Proj Phase 3 | Testing | 1 | 2022-05-02 20:21:35 |
| 16 | 13 | Proj Phase 3 | Suggestions | 1 | 2022-05-02 20:21:35 |
| 17 | 15 | Proj Phase 3 | Additional queries | 1 | 2022-05-02 20:21:35 |
+-----+-----+-----+-----+-----+-----+
17 rows in set (0.00 sec)
```

Suggestions on Database Tuning

Since the database is very simple, we do not have any suggestions that would make the database more effective or even simpler.

When it comes to potential queries and commands, the asterisk should be avoided as much as possible so the system would not have to load as often and we only get the necessary data that we are looking for.

For indexing, possibly we can look at each board being a bucket corresponding to the userID which is indexed using hashes so that after finding the user when logging in, it would go to the index with the board(s) corresponding to the user then that board hash or bucket contains all the information needed for that specific board.

Additional queries and views

USER

userId	password	email	firstName	lastName
User1	Password1	User1@email.com	Amber	Cox
User2	Password2	User2@email.com	Daniel	Ruiz
User3	Password3	User3@email.com	Zach	Duffy

BOARD

boardId	BuserId	boardName	Description
Board1	User1	Workout	Exercises to complete
Board2	User2	School	Assignments
Board3	User3	Errands	House chores, shopping for groceries, etc.

TASKS

taskId	TboardId	taskName	isComplete	Description
Task1	Board1	Running	True	Run 2 miles
Task2	Board2	Paper	False	Write 4 page paper
Task3	Board3	Groceries	False	Shop for groceries for the week

User Management

User logins	User Info
SELECT userId,password FROM USER;	SELECT email,firstName,lastName FROM USER;

Add User	Change password
INSERT INTO USER VALUES ('User4', 'Password4', 'User4@email.com', 'Maegan', 'Vang');	UPDATE USER SET password = newPassword WHERE userId = 'User1';

Task board and Task System

Current Tasks	Add task board/task
SELECT taskName FROM TASKS;	INSERT INTO BOARD VALUES ('Board4','User4','Vacation', 'Plan itinerary/book tickets etc.');
	INSERT INTO TASKS VALUES ('Task4','Board4','Flight ticket', 'False' ,'Book flight for trip');

Edit task board/task
UPDATE BOARD SET boardName = newBoardName WHERE boardId = 'Board1';
UPDATE TASK SET taskName = newTaskName WHERE taskId = 'Task2';

Views

```
CREATE VIEW COMPLETION_STATUS
AS SELECT      userId,boardName,taskName,isComplete
FROM          USER,BOARD,TASKS
WHERE         userId=BuserId AND boardId=TboardId;
```

userId	boardName	taskName	isComplete
User1	Workout	Running	True
User2	School	Paper	False
User3	Errands	Groceries	False

```
CREATE VIEW USERS_INCOMPLETE_TASKS
AS SELECT      firstName,lastName,boardName,taskName,isComplete
FROM          USER,BOARD,TASKS
WHERE         userId=BuserId AND boardId=TboardId AND isComplete=False;
```

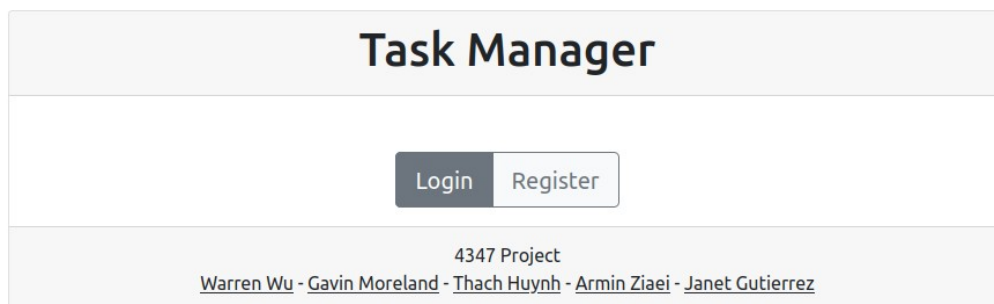
firstName	lastName	boardName	taskName	isComplete
Daniel	Ruiz	School	Paper	False
Zach	Duffy	Errands	Groceries	False

User Application Interface

The user interface was made using PHP. Each page is rendered server-side and injected with the information that's relevant to it using SQL. The styling is done using CSS with Bootstrap and the content of each page is just HTML. The normal user flow goes as follows:

Index.php

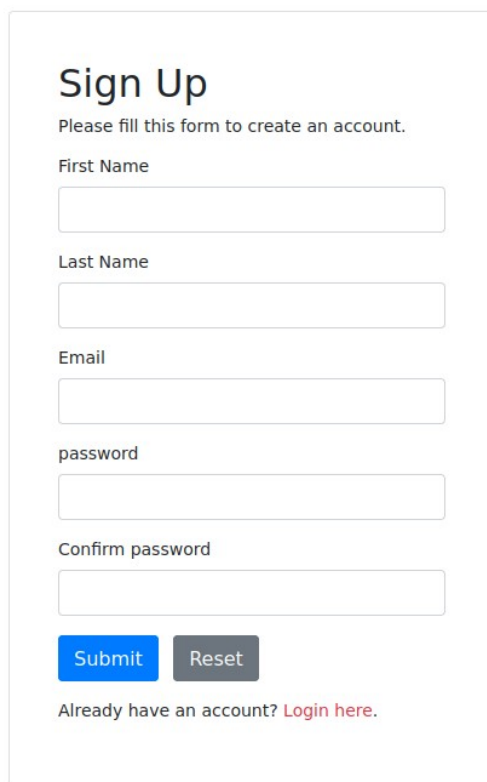
The program starts from the index.php page and gives the user two options to login or register.



The screenshot shows a web interface titled "Task Manager" in a large, bold, black font at the top. Below the title, there are two buttons: "Login" (dark gray) and "Register" (light gray). At the bottom of the page, there is a footer section containing the text "4347 Project" and a list of names: "Warren Wu - Gavin Moreland - Thach Huynh - Armin Ziaei - Janet Gutierrez".

Registration

The user will fill this form out and our PHP code will construct a SQL statement based on the information entered to create a new user. That new user's information is stored on a session object afterwards for future authentication purposes.



The screenshot shows a "Sign Up" registration form. The title "Sign Up" is in a large, bold, black font. Below it, a subtitle reads "Please fill this form to create an account." The form contains five input fields: "First Name", "Last Name", "Email", "password", and "Confirm password". At the bottom of the form, there are two buttons: "Submit" (blue) and "Reset" (gray). Below the buttons, there is a link that says "Already have an account? [Login here.](#)".

Registration Implementation

After the user has registered, they are pushed to the login.php page.

```
// Prepare an insert statement
$sql = "INSERT INTO user (Email, Password, FirstName, LastName) VALUES (?, ?, ?, ?)";

if($stmt = mysqli_prepare($link, $sql)){
    // Bind variables to the prepared statement as parameters
    mysqli_stmt_bind_param($stmt, "ssss", $param_email, $param_password, $param_fname, $param_lname);

    // Set parameters
    $param_fname = $first_name;
    $param_lname = $last_name;
    $param_email = $email;
    $param_password = password_hash($password, PASSWORD_DEFAULT); // Creates a password hash

    // Attempt to execute the prepared statement
    if(mysqli_stmt_execute($stmt)){
        // Redirect to login page
        header("location: login.php");
    } else{
        echo "Oops! Something went wrong. Please try again later.";
    }

    // Close statement
    mysqli_stmt_close($stmt);
}
```

Login

If a user already has an account, they can use this form to login to our application. If successful, the pertinent user information is returned to the front-end and stored on a session object for future authentication.

Login

Please fill in your credentials to login.

Username

Password

Don't have an account? [Sign up now.](#)

Login Implementation

After the user has logged-in successfully, they are pushed to the dashboard.php page.

```
// Validate credentials
if(empty($email_err) && empty($password_err)){
    // Prepare a select statement
    $sql = "SELECT UserId, Email, password, FirstName, LastName FROM user WHERE Email = ?";

    if($stmt = mysqli_prepare($link, $sql)){
        // Bind variables to the prepared statement as parameters
        mysqli_stmt_bind_param($stmt, "s", $param_username);

        // Set parameters
        $param_username = $email;

        // Attempt to execute the prepared statement
        if(mysqli_stmt_execute($stmt)){
            // Store result
            mysqli_stmt_store_result($stmt);

            // Check if username exists, if yes then verify password
            if(mysqli_stmt_num_rows($stmt) == 1){
                // Bind result variables
                mysqli_stmt_bind_result($stmt, $id, $email, $hashed_password, $first_name, $last_name);
                if(mysqli_stmt_fetch($stmt)){
                    if(password_verify($password, $hashed_password)){
                        // Password is correct, so start a new session
                        session_start();

                        // Store data in session variables
                        $_SESSION["loggedin"] = true;
                        $_SESSION["UserId"] = $id;
                        $_SESSION["Email"] = $email;
                        $_SESSION["FirstName"] = $first_name;
                        $_SESSION["LastName"] = $last_name;

                        // Redirect user to welcome page
                        header("location: dashboard.php");
                    } else{
                        // Password is not valid, display a generic error message
                        $login_err = "Invalid username or password.";
                    }
                }
            } else{
                // Username doesn't exist, display a generic error message
                $login_err = "Invalid username or password.";
            }
        } else{
            echo "Oops! Something went wrong. Please try again later.";
        }
    }
}
```

Dashboard Page

On this page, users are presented with the boards that are associated with their account and are also given the option to edit existing boards and create new boards.

dashboard.php

Add New Board

Reset Your Password

Sign Out of Your Account

Hi, **admin**. Welcome.

Owner	Board Name	Board Description	Created at			
admin	Board_1	Board_1 Description	2022-05-02 17:42:46	Select	Edit	Delete
admin	Board_2	Board_2 Description	2022-05-02 17:43:19	Select	Edit	Delete
admin	Board_3	Board_3 Description	2022-05-02 17:51:45	Select	Edit	Delete
admin	Board_4	Board_4 Description	2022-05-02 17:51:56	Select	Edit	Delete

add_board.php

New Board

Board Name

Board Description

Add

[Cancel](#)

edit_board.php

Edit Board

Board Name

Board Description

Edit

[Cancel](#)

delete_board.php

Delete the Board!

Click

[Cancel](#)

Associated SQL

Adding a board

```
// Prepare an insert statement
$sql = "INSERT INTO board (UserId, Name, Description) VALUES (?, ?, ?)";
```

The information about the user is injected into the SQL statement using the session object from earlier.

Editing a board

```
// Prepare an update statement
$sql = "UPDATE board as b
SET b.Name = ?, b.Description = ?
WHERE b.BoardId = ?";
```

To edit a board, we have to have the boardId. To get this value, we use query parameters that are injected into the URL when the edit button is clicked for a board. This value is then available on the next page and can be fetched to inject into this statement. This same process is repeated for any statement where we need data to persist across pages.

Deleting a board

```
// Prepare an update statement  
$sql = "DELETE FROM board WHERE BoardId = ?;";
```

Board Page

On this page, users can view what tasks are associated with the board as well as create, update, and delete them. They can also mark them as complete or incomplete.

Show_board.php (show all tasks under an specific board)

[Add New Task](#)

[Home Page \(Dashboard\)](#)

[Sign Out of Your Account](#)

Owner: **admin**
Board Name: **Board_1**

Task Title	Task Description	Created at	Status	
task_1	task_1 Description	2022-05-02 17:57:16	0	Done Edit Delete
task_2	task_2 Description	2022-05-02 17:57:27	0	Done Edit Delete
task_3	task_3 Description	2022-05-02 17:57:39	0	Done Edit Delete

add_task.php

New Task

Task Name

Task Description

Add

[Cancel](#)

edit_task.php

Edit Task

Task Name

Task Description

Edit

[Cancel](#)

delete_task.php

Delete the Task!

Click

[Cancel](#)

Retrieve tasks for board

```
$sql = "SELECT TaskId, Name, Description, IsComplete, created_at FROM task WHERE BoardId = ?";
```


Add a task

```
// Prepare an insert statement
$sql = "INSERT INTO task (BoardId, Name, Description, IsComplete) VALUES (?, ?, ?, ?)";
```

Delete a task

```
// Prepare an update statement
$sql = "DELETE FROM task WHERE TaskId = ?";
```

Update if task is complete

```
// Prepare an update statement
$sql = "UPDATE task as t
SET t.IsComplete = '1' WHERE t.TaskId = ?";
```

Edit a task

```
// Prepare an update statement
$sql = "UPDATE task as t
SET t.Name = ?, t.Description = ?
WHERE t.TaskId = ?";
```

Conclusions and Future Work

Conclusions

We used technologies such as Mysql, PHP, etc., to build an independent project. The project aims to interact with the database and create a graphical interface for sending/receiving data to the database. Each stage of the project made it possible to learn and apply the concepts we had learned in class. We used MySQL for the database and PHP for the graphical environment to build this program. Regardless of the power and beauty of PHP, the use of technologies such as Node.js, React is essential to enhancing the security and integrity of the application. I have learned a lot during the project. Also, we had some challenges, such as routing the pages. So, we implemented a method for routing the PHP pages when we wanted to redirect a request to another. URL routing allowed us to configure the application to accept request URLs that did not map to physical files. Overall, this project was one of the most exciting projects I had during my studies.

Future Work

Overall, the database is feature complete. It's a nice and simple online way to keep track of tasks and managed them anywhere. We could work to give users QOL features such as merging boards or deleting multiple tasks, but these are niche features and could lead to poor UI or user experience. Instead, we believe that the next step would be to allow multiple users/organizations to have joint access over certain tasks. This was something that we considered early on in the project, but was not implemented as we worried about how it would

affect the complexity of our project and could lead to concurrency problems with one user having a different version of a task.

References

- Elmasri, Ramez, and Shamkant B Navathe. *Fundamentals of Database Systems*. 7th ed., Pearson, 2016.