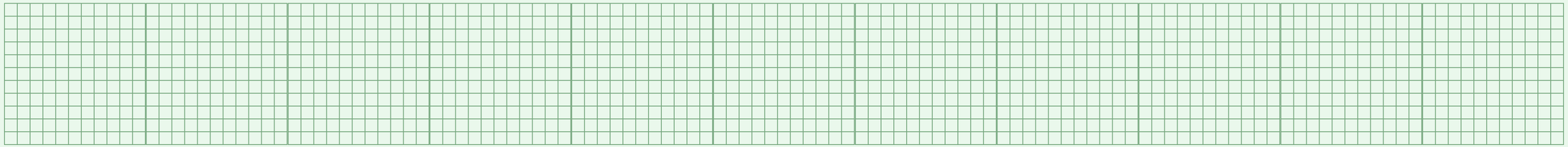
Table of Contents



SCRIBE UNNORMALIZED DATABASE

User_ID	Username	Email	Password	Note_ID	Note_Title	Note_Content	Task_ID	Task_Description	Task_Due_Date	Folder_ID	Folder_Name	Category_ID	Category_Name	Archived	Trash	Theme
1	john1	john1@gmail.com	John.123	101	Note 1	Content 1	201	Task 1	2024-05-24	301	Folder 1	401	Category 1	False	False	Light
1	john1	john1@gmail.com	John.123	102	Note 2	Content 2	202	Task 2	2024-05-25	302	Folder 1	402	Category 2	True	True	Dark

SCRIBE DATABASE NORMALIZATION

1NF

User_ID	Username	Email	Password	Firstname	Lastname	Created_At	Updated_At
1	john1	john@gmail.com	John.123	John	Doe	2024-05-24	2024-05-25

Note_ID	User_ID	Note_Title	Note_Content	Created_At	Updated_At
101	1	Note 1	Content 1	2024-05-25	2024-05-25

Task_ID	User_ID	Task_Description	Task_Due_Date	Created_At	Updated_At
201	1	Task 1	2024-05-30	2024-05-25	2024-05-25

Folder_ID	User_ID	Folder_Name	Created_At	Updated_At
301	1	Folder 1	2024-05-25	2024-05-25

SCRIBE DATABASE NORMALIZATION

1NF

Category_ID	User_ID	Category_Name	Created_At	Updated_At
401	1	Category 1	2024-05-25	2024-05-25

Archive_ID	User_ID	Note_ID	Archived_At
501	1	101	2024-05-25

Trash_ID	User_ID	Note_ID	Trashed_At
601	1	101	2024-05-25

SCRIBE DATABASE NORMALIZATION

2NF

DOES NOT SATISFY 2NF

Archive_ID	User_ID	Note_ID	Archived_At
501	1	101	2024-05-25

Trash_ID	User_ID	Note_ID	Trashed_At
601	1	101	2024-05-25

Archive_ID	User_ID	Archived_Type	Type_ID	Archived_At
501	1	Note	101	2024-05-25

Trash_ID	User_ID	Trashed_Type	Type_ID	Trashed_At
601	1	Folder	301	2024-05-25

SCRIBE DATABASE NORMALIZATION

BCNF 3NF 4NF

TABLES ARE ALREADY AND SATISFIED 3NF.

TABLES ARE ALREADY IN BCNF SINCE NO DEPENDENCIES VIOLATE THE BCNF.

NOTE TO FOLDER TABLE

Note_ID	Moved_to_Folder_ID	Updated_At
101	301	2024-05-25

TASK TO FOLDER TABLE

Task_ID	Moved_to_Folder_ID	Updated_At
201	301	2024-05-25

NOTE TO CATEGORY TABLE

Note_ID	Set_to_Category_ID	Updated_At
101	401	2024-05-25

TASK TO FOLDER TABLE

Task_ID	Set_to_Category_ID	Updated_At
201	401	2024-05-25

SCRIBE DATABASE NORMALIZATION

5NF

users_table

User_ID	Username	Email	Password	Firstname	Lastname	Created_At	Updated_At
1	john1	john@gmail.com	John.123	John	Doe	2024-05-24	2024-05-25

notes_table

Note_ID	User_ID	Note_Title	Note_Content	Created_At	Updated_At
101	1	Note 1	Content 1	2024-05-25	2024-05-25

tasks_table

Task_ID	User_ID	Task_Description	Task_Due_Date	Created_At	Updated_At
201	1	Task 1	2024-05-30	2024-05-25	2024-05-25

folders_table

Folder_ID	User_ID	Folder_Name	Created_At	Updated_At
301	1	Folder 1	2024-05-25	2024-05-25

SCRIBE DATABASE NORMALIZATION

5NF

categories_table

Category_ID	User_ID	Category_Name	Created_At	Updated_At
401	1	Category 1	2024-05-25	2024-05-25

archives_table

Archive_ID	User_ID	Archived_Type	Type_ID	Archived_At
501	1	Note	101	2024-05-25

trashes_table

Trash_ID	User_ID	Trashed_Type	Type_ID	Trashed_At
601	1	Folder	301	2024-05-25

SCRIBE DATABASE NORMALIZATION

5NF

notes_to_folders_table

Note_ID	Moved_to_Folder_ID	Updated_At
101	301	2024-05-25

notes_to_categories_table

Note_ID	Set_to_Category_ID	Updated_At
101	401	2024-05-25

tasks_to_folders_table

Task_ID	Moved_to_Folder_ID	Updated_At
201	301	2024-05-25

tasks_to_categories_table

Task_ID	Set_to_Category_ID	Updated_At
201	401	2024-05-25

SCRIBE ERD

Users(user_id PK, first_name, last_name, username, email, password, created_at, updated_at)

Notes(note_id PK, user_id FK, note_title, note_content, created_at, updated_at)

Tasks(task_id PK, user_id FK, task_description, task_due_date, created_at, updated_at)

Folders(folder_id PK, user_id FK, folder_name, created_at, updated_at)

Categories(category_id PK, user_id FK, category_name, created_at, updated_at)

Archives(archive_id PK, user_id FK, archive_type, type_id FK, archived_at)

Trashes(trash_id PK, user_id FK, trashed_type, type_id FK, trashed_at)

Note_to_Folder(note_id FK, moved_to_folder_id FK, updated_at, PK(note_id, moved_to_folder_id))

Note_to_Category(note_id FK, set_to_category_id FK, updated_at, PK(note_id, set_to_category_id))

Task_to_Folder(task_id FK, moved_to_folder_id FK, updated_at, PK(task_id, moved_to_folder_id))

Task_to_Category(task_id FK, set_to_category_id FK, updated_at, PK(task_id, set_to_category_id))

SCRIBE ERD

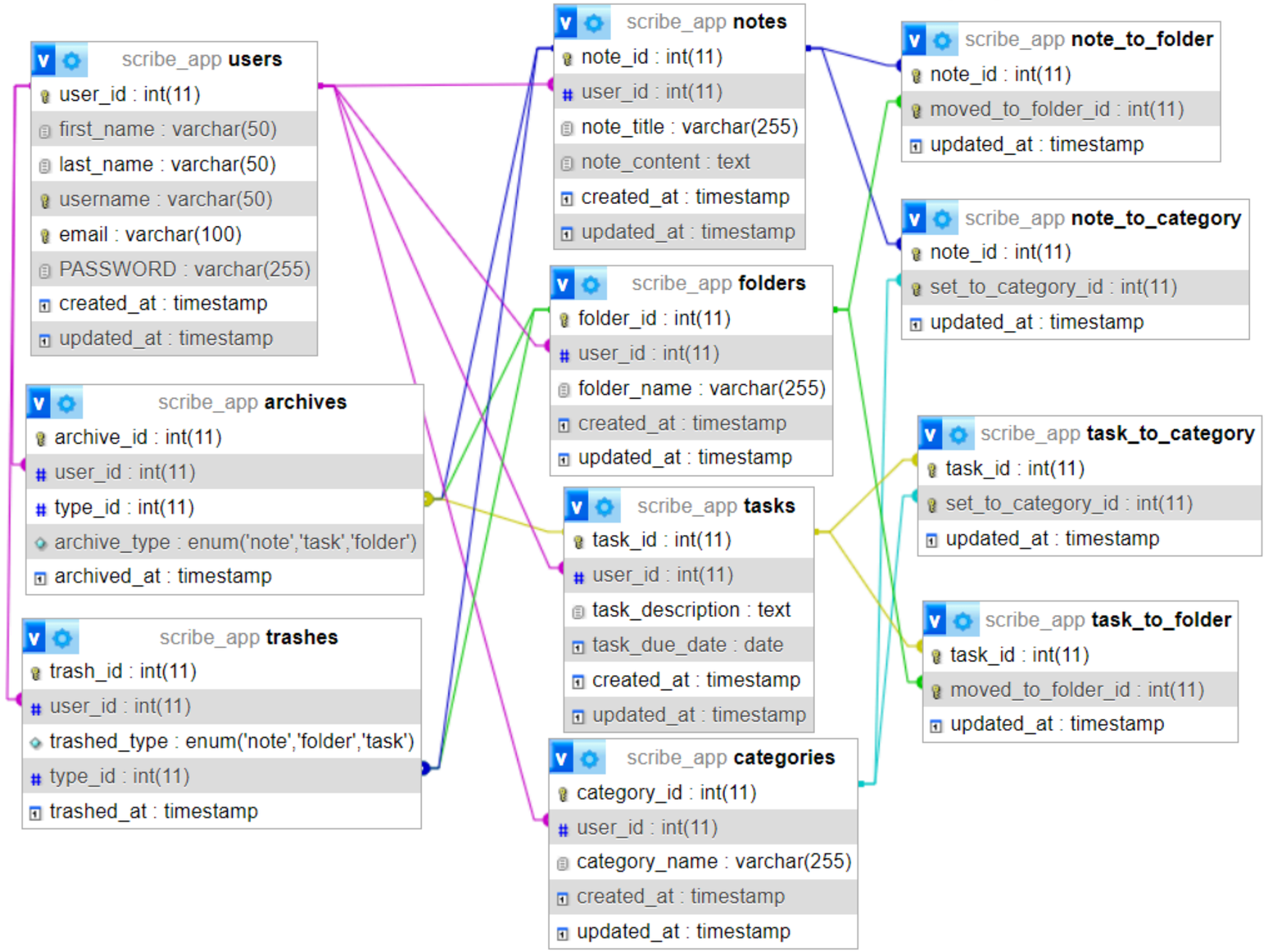
Entity Descriptions

1. **Users:** Represents users of the application.
2. **Notes:** Represents notes created by users.
3. **Tasks:** Represents tasks created by users.
4. **Folders:** Represents folders created by users to organize notes.
5. **Categories:** Represents categories created by users to categorize notes.
6. **Archives:** Represents archived items (notes or folders) by users.
7. **Trashes:** Represents trashed items (notes or folders) by users.

Relationship Descriptions

- **User - Notes**: One user can create multiple notes. (One-to-Many)
- **User - Tasks**: One user can create multiple tasks. (One-to-Many)
- **User - Folders**: One user can create multiple folders. (One-to-Many)
- **User - Categories**: One user can create multiple categories. (One-to-Many)
- **User - Archives:** One user can archive multiple items. (One-to-Many)
- **User - Trashes**: One user can trash multiple items. (One-to-Many)
- **Notes - Folders**: One note can belong to multiple folders, and one folder can contain multiple notes. (Many-to-Many)
- **Notes - Categories**: One note can belong to multiple categories, and one category can contain multiple notes. (Many-to-Many)
- **Tasks - Folders**: One task can belong to multiple folders, and one folder can contain multiple tasks. (Many-to-Many)
- **Tasks - Categories**: One task can belong to multiple categories, and one category can contain multiple tasks. (Many-to-Many)

SCHEMA DIAGRAM



SQL QUERY

```
-- Create the database
CREATE DATABASE IF NOT EXISTS scribe_app;

-- Use the database
USE scribe_app;

-- Create the Users table
CREATE TABLE IF NOT EXISTS Users (
  user_id INT PRIMARY KEY AUTO_INCREMENT,
  first_name VARCHAR(50) NOT NULL,
  last_name VARCHAR(50) NOT NULL,
  username VARCHAR(50) NOT NULL UNIQUE,
  email VARCHAR(100) NOT NULL UNIQUE,
  password VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
);


-- Create the Notes table
CREATE TABLE IF NOT EXISTS Notes (
  note_id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT NOT NULL,
  note_title VARCHAR(255),
  note_content TEXT,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);
```

```
-- Create the Tasks table
CREATE TABLE IF NOT EXISTS Tasks (
  task_id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT NOT NULL,
  task_description TEXT NOT NULL,
  task_due_date DATE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);

-- Create the Folders table
CREATE TABLE IF NOT EXISTS Folders (
  folder_id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT NOT NULL,
  folder_name VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);

-- Create the Categories table
CREATE TABLE IF NOT EXISTS Categories (
  category_id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT NOT NULL,
  category_name VARCHAR(255) NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE
);
```

SQL QUERY

```
-- Create the Archives table
CREATE TABLE IF NOT EXISTS Archives (
  archive_id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT NOT NULL,
  archive_type ENUM('note', 'folder',’task’) NOT NULL,
  type_id INT NOT NULL,
  archived_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
  CONSTRAINT FK_Archive_Type_ID FOREIGN KEY (type_id) REFERENCES Notes(note_id) ON
DELETE CASCADE,
  CONSTRAINT FK_Archive_Folder_Type_ID FOREIGN KEY (type_id) REFERENCES
Folders(folder_id) ON DELETE CASCADE
);


-- Create the Trashes table
CREATE TABLE IF NOT EXISTS Trashes (
  trash_id INT PRIMARY KEY AUTO_INCREMENT,
  user_id INT NOT NULL,
  trashed_type ENUM('note', 'folder',’task’) NOT NULL,
  type_id INT NOT NULL,
  trashed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES Users(user_id) ON DELETE CASCADE,
  CONSTRAINT FK_Trash_Type_ID FOREIGN KEY (type_id) REFERENCES Notes(note_id) ON
DELETE CASCADE,
  CONSTRAINT FK_Trash_Folder_Type_ID FOREIGN KEY (type_id) REFERENCES
Folders(folder_id) ON DELETE CASCADE
);

-- Create the Notes to Folders table
CREATE TABLE IF NOT EXISTS Note_Folder (
 note_id INT NOT NULL,
 moved_to_folder_id INT NOT NULL,
 updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
 PRIMARY KEY (note_id, moved_to_folder_id),
 FOREIGN KEY (note_id) REFERENCES Notes(note_id) ON DELETE CASCADE,
 FOREIGN KEY (moved_to_folder_id) REFERENCES Folders(folder_id) ON DELETE CASCADE
);
```

```
-- Create the Notes to Categories table
CREATE TABLE IF NOT EXISTS Note_Category (
  note_id INT NOT NULL,
  set_to_category_id INT NOT NULL,
  updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (note_id, set_to_category_id),
  FOREIGN KEY (note_id) REFERENCES Notes(note_id) ON DELETE CASCADE,
  FOREIGN KEY (set_to_category_id) REFERENCES Categories(category_id) ON DELETE
CASCADE
);

-- Create the Task_To_Folder table
CREATE TABLE IF NOT EXISTS Task_To_Folder (
   task_id INT NOT NULL,
   moved_to_folder_id INT NOT NULL,
   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
   PRIMARY KEY (task_id, moved_to_folder_id),
   FOREIGN KEY (task_id) REFERENCES Tasks(task_id) ON DELETE CASCADE,
   FOREIGN KEY (moved_to_folder_id) REFERENCES Folders(folder_id) ON DELETE CASCADE
);

-- Create the Task_To_Category table
CREATE TABLE IF NOT EXISTS Task_To_Category (
   task_id INT NOT NULL,
   set_to_category_id INT NOT NULL,
   updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
   PRIMARY KEY (task_id, set_to_category_id),
   FOREIGN KEY (task_id) REFERENCES Tasks(task_id) ON DELETE CASCADE,
   FOREIGN KEY (set_to_category_id) REFERENCES Categories(category_id) ON DELETE
CASCADE
);
```