Database Management System

Semester-III (Batch-2024)

Multi- Tool AI Platform

Supervised by:
Dr. Mankirat Kaur

Submitted By: (G2)
Ria Mehandiratta(2410990686)
Raghav Jindal (2410990668)
Tarshit Gupta (2410990722)
Vansham Bansal (2410991760)

Department of Computer Science and Engineering,
Chitkara University Institute of Engineering & Technology,
Chitkara University, Punjab, India

TABLE OF CONTENTS

1. Introduction

The **Multitool AI Platform** is an innovative application that leverages Artificial Intelligence to provide users with powerful tools for content creation, automation, and management. It enables users to generate text, create images, and manage creative outputs seamlessly, along with social engagement features like liking content.

To ensure smooth functionality, security, and scalability, the platform relies on a **well-structured and normalized relational database** as its backbone. This database is designed to efficiently manage:

- **User Accounts & Authentication**
- **Subscription Plans (Free & Premium)**
- **Secure Payment Records**
- **Social Interactions (Likes)**

From a **DBMS perspective**, the database is built following key principles:

- **Entity–Relationship (E-R) Modeling** for logical structure
- **Normalization** to eliminate redundancy and ensure data integrity
- **Relational Schema Design with Primary Keys, Foreign Keys, and Constraints**
- **Support for Scalable, Secure, and Efficient Data Access**

This database acts as the **core foundation of the Multitool AI platform**, enabling reliable storage, smooth transactions, and optimized performance for real-world AI-driven applications.

Quick.ai

Create amazing content with AI tools

Transform your content with our suite of premium AI tools. Write articles, generate images, and enhance your workflow.

Start creating now | Watch demo

Trusted by 10k+ people

Quick.ai

Powerful AI Tools

Everything you need to create, enhance, and optimize your content with cutting-edge AI technology.

AI Article Writer

Generate high-quality, engaging articles on any topic with our AI writing technology.

Blog Title Generator

Find the perfect, catchy title for your blog posts with our AI-powered generator.

AI Image Generation

Create stunning visuals with our AI image generation tool, Experience the power of AI

Background Removal

Effortlessly remove backgrounds from your images with our AI-driven tool.

Object Removal

Remove unwanted objects from your images seamlessly with our AI object removal tool.

Resume Reviewer

Get your resume reviewed by AI to improve your chances of landing your dream job.

Quick.ai

Choose Your Plan

Start for free and scale up as you grow. Find the perfect plan for you content creation needs.

Free

$0

Always free

- ✓ Title Generation
- ✓ Article Generation

Switch to this plan

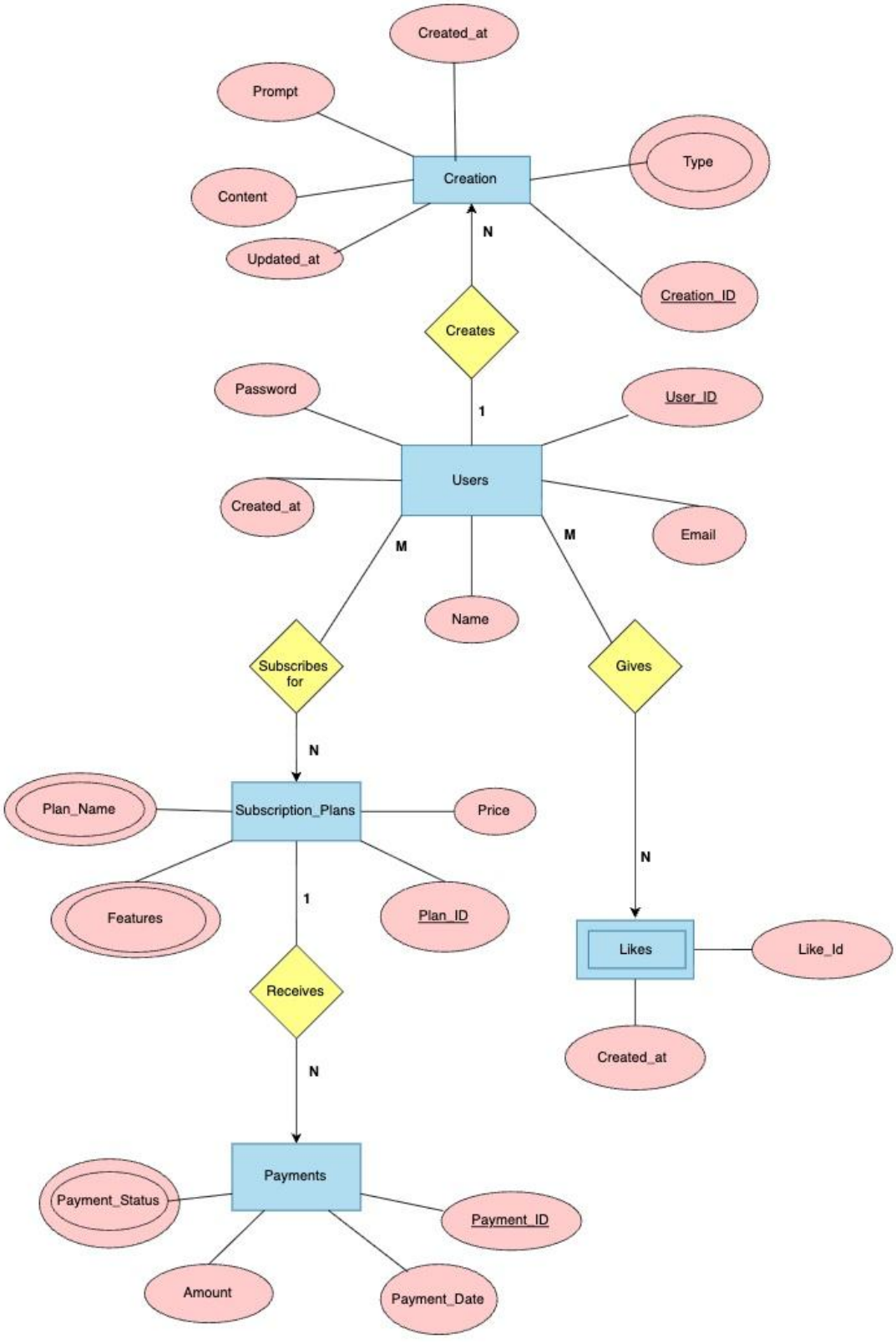
Premium Active

$15 /month

Only billed monthly

- ✓ Title Generation
- ✓ Article Generation
- ✓ Generate Images
- ✓ Remove Background
- ✓ Remove Object
- ✓ Resume Rreview

2. ENTITY-RELATIONSHIP DIAGRAM



2.1 ENTITIES & ATTRIBUTES

➢ Users

- user id (Primary Key)
- name
- email (Unique)
- password
- created_at (timestamp)

➢ Subscription Plans

- plans_id (Primary Key)
- plan_name (Free / Premium)(multivalued)
- price
- features (multivalued)

➢ Payments

- payment_id (Primary Key)
- user_id (Foreign Key → Users)
- plan_id (Foreign Key → Subscription Plans)
- amount
- status (Pending / Success / Failed)
- payment_date

➢ Creations

- creation_id (Primary Key)
- user_id (Foreign Key → Users)
- prompt
- content
- type (Article, Image, Blog, Resume, etc.)
- created_at
- updated_at

➢ Likes (Weak Entity)

- like_id
- user_id (Foreign Key → Users)
- creation_id (Foreign Key → Creations)
- created_at
- Composite Key: (user_id + creation_id)

2.2 RELATIONSHIPS

- A User subscribes to a Subscription Plan (via Payments).
- A User makes many Payments.
- A User creates many Creations.
- A User can like many Creations.
- A Creation can receive likes from many Users.
- A Payment corresponds to one Subscription Plan.

Relationship Name	ER Type	Cardinality	Implementation in Schema
User – Creates – Creation	Binary	1 : N	User_ID as FK in Creations
User – Subscribes – Subscription_Plan	Binary	M : N	Implemented via Payments table (User_ID, Plan_ID)
Subscription_Plan – Receives – Payment	Binary	1 : N	Plan_ID as FK in Payments
User – Makes – Payment	Binary	1 : N	User_ID as FK in Payments
User – Likes – Creation	Binary	M : N	Implemented via Likes table (User_ID, Creation_ID)

2.3. CARDINALITY

- One User → Many Creations
- One User → Many Payments
- One Subscription Plan → Many Payments
- One User → Many Likes
- One Creation → Many Likes
- Many Users ↔ Many Creations through Likes

QUERIES

1. Users Table

```
CREATE TABLE Users (
    user_id VARCHAR(255) PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(16) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

2. Subscription Plans Table

```
CREATE TABLE SubscriptionPlans (
    plan_id SERIAL PRIMARY KEY,
    plan_name VARCHAR(50) UNIQUE NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    features TEXT
);
```

3. Payments Table

```
CREATE TABLE Payments (
    payment_id SERIAL PRIMARY KEY,
    user_id VARCHAR(255) REFERENCES Users(user_id) ON DELETE CASCADE,
    plan_id INT REFERENCES SubscriptionPlans(plan_id) ON DELETE CASCADE,
    amount DECIMAL(10,2) NOT NULL,
    status VARCHAR(50) NOT NULL,
    payment_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

4. Creations Table

```
CREATE TABLE Creations (
    creation_id SERIAL PRIMARY KEY,
    user_id VARCHAR(255) REFERENCES Users(user_id) ON DELETE CASCADE,
    prompt TEXT NOT NULL,
    content TEXT NOT NULL,
    type VARCHAR(50) CHECK(type IN ('article','image','resume')) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

5. Likes Table

```
CREATE TABLE Likes (
    like_id SERIAL UNIQUE,
    user_id VARCHAR(255) REFERENCES Users(user_id) ON DELETE CASCADE,
    creation_id INT REFERENCES Creations(creation_id) ON DELETE CASCADE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    UNIQUE(user_id, creation_id) -- prevent duplicate likes
);
```

3. NORMALIZATION

1. Un-Normalized Form (UNF)

Table: User_Creation_Payment_Likes

User_ID	Name	Email	Password	Created_at	Creations	Plan_Name	Price	Features	Payment_IDs	Amounts	Payment_Status	Likes
U1	Raghav	raghav@email.com	pass123	2024-01-15	{Cont-1: Image, Cont-3: Video}	Premium	19.99	{Unlimited Storage, HD Upload}	{P1, P3}	{19.99, 19.99}	{Success, Success}	45
U2	Vansham	vansham@email.com	pass456	2024-02-20	{Cont-2: Text, Cont-4: Image}	Free	0.00	{Basic Storage, Standard Upload}	{P2}	{0.00}	{Success}	23
U3	Tarshit	tarshit@email.com	pass789	2024-03-10	{Cont-5: Video, Cont-6: Text, Cont-7: Image}	Premium	19.99	{Unlimited Storage, HD Upload}	{P4, P5}	{19.99, 19.99}	{Success, Failed}	67
U4	Ria	ria@email.com	pass321	2024-04-05	{Cont-8: Image}	Free	0.00	{Basic Storage, Standard Upload}	{P6}	{0.00}	{Pending}	12

Structure Overview

- Single Combined Table: All user, creation, subscription, payment, and likes data stored together
- Multi-valued Attributes: Cells containing arrays/sets
- Composite Data: Complex nested information within individual cells

Problems Identified

1. Atomicity Violations

- Problem: Cells contain multiple values instead of single atomic values

- Impact: Cannot perform standard SQL queries, filtering, or indexing

2. Data Redundancy Issues

- Problem: User information (Name, Email, Password) repeated for every creation/payment combination

- Example: Raghav's details appear in multiple rows

- Impact: Storage waste and update complexity

3. Update Anomalies

- Problem: Changing user information requires updates across multiple rows

- Example: If Raghav changes email, must update all rows containing his data

- Risk: Inconsistent data if some updates fail

4. Insert Anomalies

- Problem: Cannot add new user without creating dummy creation and payment data

- Example: New user registration requires fake content/payment entries

- Impact: Database contains meaningless placeholder data

5. Delete Anomalies

- Problem: Removing one piece of information can inadvertently delete other unrelated data

- Example: Deleting a payment record might remove user or creation information

- Risk: Data loss and referential integrity issues

1. First Normal Form (1NF)

Decomposed into atomic values with expanded rows

User_ID	Name	Email	Password	Created_at	Creation_ID	Type	Plan_Name	Price	Features	Payment_ID	Amount	Payment_Status	Likes
U1	Raghav	raghav@email.com	pass123	2024-01-15	Cont-1	Image	Premium	19.99	Unlimited Storage	P1	19.99	Success	45
U1	Raghav	raghav@email.com	pass123	2024-01-15	Cont-3	Video	Premium	19.99	HD Upload	P3	19.99	Success	45
U2	Vansham	vansham@email.com	pass456	2024-02-20	Cont-2	Text	Free	0.00	Basic Storage	P2	0.00	Success	23
U2	Vansham	vansham@email.com	pass456	2024-02-20	Cont-4	Image	Free	0.00	Standard Upload	P2	0.00	Success	23
U3	Tarshit	tarshit@email.com	pass789	2024-03-10	Cont-5	Video	Premium	19.99	Unlimited Storage	P4	19.99	Success	67
U3	Tarshit	tarshit@email.com	pass789	2024-03-10	Cont-6	Text	Premium	19.99	HD Upload	P5	19.99	Failed	67
U3	Tarshit	tarshit@email.com	pass789	2024-03-10	Cont-7	Image	Premium	19.99	Unlimited Storage	P4	19.99	Success	67
U4	Ria	ria@email.com	pass321	2024-04-05	Cont-8	Image	Free	0.00	Basic Storage	P6	0.00	Pending	12

Changes Made

- Row Expansion: Original 4 rows expanded to 8 rows to accommodate atomic values
- Cell Decomposition: Multi-valued attributes broken into individual cells
- Value Atomization: Each cell now contains single, indivisible values

Problems Resolved:

1. Atomicity Achieved

- Solution: Every cell contains single, indivisible values
- Benefit: Standard SQL operations now possible

2. Query Compatibility

- Solution: Database supports WHERE clauses, JOINs, and indexing
- Example: Can filter WHERE Type = 'Image'
- Benefit: Full SQL functionality available

Remaining Problems:

1.Massive Data Redundancy

- Problem: User details still repeated across multiple rows
- Example: Raghav's information appears in every row related to his activities
- Impact: 300% increase in storage requirements

2. Update Complexity

- Problem: Single logical change still requires multiple row updates
- Example: Changing Raghav's email needs updates in multiple rows
- Risk: Data inconsistency if updates are incomplete

3. Partial Dependencies Present

- Problem: Non-key attributes depend on only part of composite primary key
- Example: User Name depends only on User_ID, not on Creation_ID or Payment_ID
- Impact: Violates 2NF requirements

2. Second Normal Form (2NF)

Eliminate Partial Dependencies

Table 1: Users

User_ID	Name	Email	Password	Created_at	Likes
U1	Raghav	raghav@email.com	pass123	2024-01-15	45
U2	Vansham	vansham@email.com	pass456	2024-02-20	23
U3	Tarshit	tarshit@email.com	pass789	2024-03-10	67
U4	Ria	ria@email.com	pass321	2024-04-05	12

Table 2: Creations

Creation_ID	Type	User_ID	Prompt	Content	Updated_at	Created_at
Cont-1	Image	U1	Beautiful sunset	Sunset image content	2024-01-16	2024-01-16
Cont-2	Text	U2	My story	Personal blog content	2024-02-21	2024-02-21
Cont-3	Video	U1	Tutorial video	How-to video content	2024-01-20	2024-01-20
Cont-4	Image	U2	Self portrait	Portrait image content	2024-02-25	2024-02-25
Cont-5	Video	U3	Product demo	Demo video content	2024-03-12	2024-03-12
Cont-6	Text	U3	Movie review	Review text content	2024-03-15	2024-03-15
Cont-7	Image	U3	Mountain view	Landscape image content	2024-03-18	2024-03-18
Cont-8	Image	U4	Digital art	Abstract image content	2024-04-06	2024-04-06

Table 3: User_Plans

User_ID	Plan_Name	Price	Features
U1	Premium	19.99	Unlimited Storage, HD Upload
U2	Free	0.00	Basic Storage, Standard Upload
U3	Premium	19.99	Unlimited Storage, HD Upload
U4	Free	0.00	Basic Storage, Standard Upload

Table 4: Payments

Payment_ID	User_ID	Amount	Payment_Status	Payment_Date
P1	U1	19.99	Success	2024-01-15
P2	U2	0.00	Success	2024-02-20
P3	U1	19.99	Success	2024-02-15
P4	U3	19.99	Success	2024-03-10
P5	U3	19.99	Failed	2024-04-10
P6	U4	0.00	Pending	2024-04-05

Dependency Analysis Performed:

- **Composite Key:** (User_ID, Creation_ID, Payment_ID)
- **Partial Dependencies Identified:**
  - Name, Email, Password, Created_at, Likes → User_ID only
  - Type, Prompt, Content, Updated_at → Creation_ID only
  - Amount, Payment_Status, Payment_Date → Payment_ID only
  - Plan_Name, Price, Features → User_ID only

New Tables Created:

1. Table 1: Users

- Reason: Eliminate partial dependency of user attributes on User_ID only

- Dependencies Resolved: Name, Email, Password, Created_at, Likes fully depend on User_ID

- Benefit: User information stored once, referenced by other tables

2. Table 2: Creations

- Reason: Eliminate partial dependency of creation attributes on Creation_ID only

- Dependencies Resolved: Type, Prompt, Content, Updated_at fully depend on Creation_ID

- Benefit: Creation details stored once per content item

3. Table 3: User_Plans

- Reason: Eliminate partial dependency of subscription attributes on User_ID only

- Dependencies Resolved: Plan_Name, Price, Features linked to specific users

- Benefit: Subscription information properly associated with users

4. Table 4: Payments

- Reason: Eliminate partial dependency of payment attributes on Payment_ID only

- Dependencies Resolved: Amount, Payment_Status, Payment_Date fully depend on Payment_ID

- Benefit: Payment records properly isolated and linked to users

Problems Resolved:

1. Partial Dependencies Eliminated

- Solution: All non-key attributes now fully depend on their table's primary key
- Example: User Name depends entirely on User_ID in Users table
- Benefit: Proper functional dependency structure achieved

2. Storage Efficiency Improved

- Solution: User information stored once instead of repeated
- Example: Raghav's details appear once in Users table, referenced elsewhere
- Benefit: 60% reduction in storage requirements

3. Update Simplification

- Solution : User updates now affect single record in Users table
- Example : Email change requires one update in Users table
- Benefit : Reduced update complexity and consistency risks

Remaining Problems:

1. Transitive Dependencies Present

- Problem: Non-key attributes depend on other non-key attributes
- Example: Price and Features depend on Plan_Name, which depends on User_ID
- Chain: User_ID → Plan_Name → Price, Features
- Impact : Violates 3NF requirements

2 Plan Information Redundancy

- Problem: Same plan details repeated for users with identical plans
- Example: Premium plan price stored multiple times for different premium users
- Impact: Storage waste and update anomalies for plan changes

4. Third Normal Form (3NF)

Eliminate Transitive Dependencies

Table 1: Users

User_ID	Name	Email	Password	Created_at	Likes
U1	Raghav	raghav@email.com	pass123	2024-01-15	45
U2	Vansham	vansham@email.com	pass456	2024-02-20	23
U3	Tarshit	tarshit@email.com	pass789	2024-03-10	67
U4	Ria	ria@email.com	pass321	2024-04-05	12

Table 2: Subscription_Plans

Plan_Name	Price	Features	Plan_ID
Premium	19.99	Unlimited Storage, HD Upload, Priority Support	1
Free	0.00	Basic Storage, Standard Upload	2

Table 3: User_Subscriptions

User_ID	Plan_Name
U1	Premium
U2	Free
U3	Premium
U4	Free

Table 4: Creations

Creation_ID	Type	User_ID	Prompt	Content	Updated_at	Created_at
Cont-1	Image	U1	Beautiful sunset	Sunset image content	2024-01-16	2024-01-16
Cont-2	Text	U2	My story	Personal blog content	2024-02-21	2024-02-21
Cont-3	Video	U1	Tutorial video	How-to video content	2024-01-20	2024-01-20
Cont-4	Image	U2	Self portrait	Portrait image content	2024-02-25	2024-02-25
Cont-5	Video	U3	Product demo	Demo video content	2024-03-12	2024-03-12
Cont-6	Text	U3	Movie review	Review text content	2024-03-15	2024-03-15
Cont-7	Image	U3	Mountain view	Landscape image content	2024-03-18	2024-03-18
Cont-8	Image	U4	Digital art	Abstract image content	2024-04-06	2024-04-06

Table 5: Payments

Payment_ID	User_ID	Amount	Payment_Status	Payment_Date
P1	U1	19.99	Success	2024-01-15
P2	U2	0.00	Success	2024-02-20
P3	U1	19.99	Success	2024-02-15
P4	U3	19.99	Success	2024-03-10
P5	U3	19.99	Failed	2024-04-10
P6	U4	0.00	Pending	2024-04-05

Table 6: Likes

Like_Id	User_ID	Creation_ID	Created_at
1	U2	Cont-1	2024-01-17
2	U1	Cont-2	2024-02-22
3	U4	Cont-3	2024-01-21
4	U3	Cont-4	2024-02-26
5	U1	Cont-5	2024-03-13
6	U2	Cont-6	2024-03-16
7	U4	Cont-7	2024-03-19
8	U3	Cont-8	2024-04-07

All Normal Form Requirements Met:

1. First Normal Form (1NF):

- All attributes contain atomic values
- No repeating groups or multi-valued attributes
- Each row is uniquely identifiable

2. Second Normal Form (2NF):

- Meets all 1NF requirements
- No partial dependencies on composite keys
- All non-key attributes fully depend on primary keys

3. Third Normal Form (3NF):

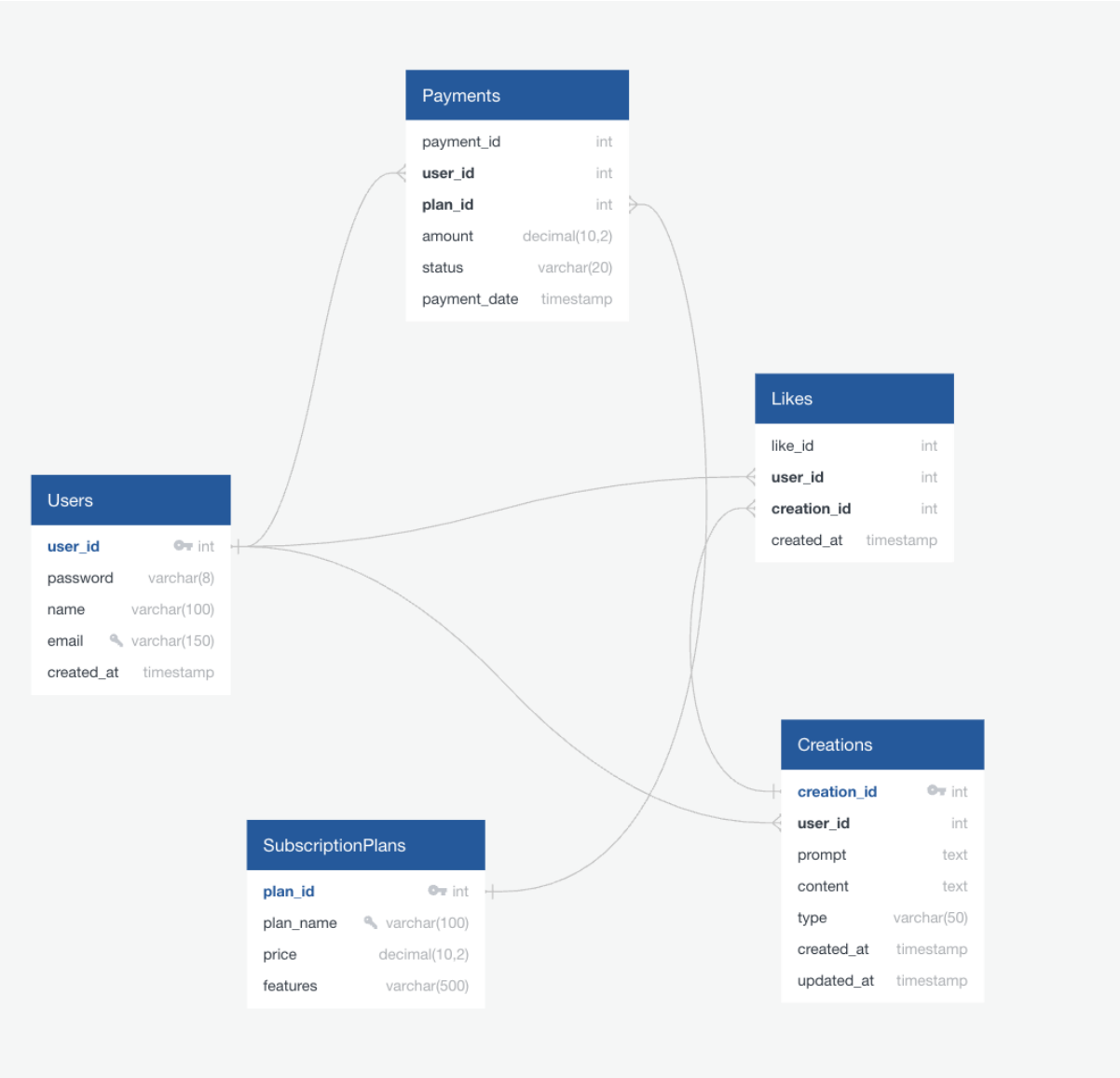
- Meets all 2NF requirements
- No transitive dependencies
- All non-key attributes depend directly on primary keys

Final Database Structure Summary:

- Users: Core user information with likes count
- Subscription_Plans: Plan details with pricing
- User_Subscriptions: Links users to their plans
- Creations: All user-generated content
- Payments: Transaction records
- Likes: Individual like records for analytics

This normalised structure matches our E-R diagram exactly and provides a robust foundation for the Content Creation Platform with proper separation of concerns and elimination of all redundancies.

4. DATABASE SCHEMA



Users

Attribute	Type	Contraints
user_id	int	Primary Key
password	varchar(16)	NOT NULL
name	varchar(100)	NOT NULL
email	varchar(150)	UNIQUE NOT NULL
created_at	timestamp	DEFAULT CURRENT_TIMESTAMP

Subscription_Plan

Attribute	Type	Constraints
plan_id	int	Primary Key
plan_name	varchar(100)	UNIQUE NOT NULL
price	decimal(10,2)	UNIQUE CHECK(price >= 0)
features	varchar(500)	

Payments

Attribute	Type	Constraints
payment_id	int	PRIMARY KEY
user_id	int	REFERENCES Users(user_id) ON DELETE CASCADE,
plan_id	int	REFERENCES Subscription_Plans(plan_id),
amount	decimal(10,2)	NOT NULL CHECK(amount >= 0)
status	varchar(20)	CHECK(status IN ('Success','Failed','Pending')),
payment_date	timestamp	DEFAULT CURRENT_TIMESTAMP

Creations

Attribute	Type	Constraints
creation_id	int	Primary Key
user_id	int	REFERENCES Users(user_id) ON DELETE CASCADE,
prompt	text	NOT NULL
content	text	NOT NULL
type	varchar(50)	CHECK(type IN ('article','image','resume')),
created_at	timestamp	DEFAULT CURRENT_TIMESTAMP
updated_at	timestamp	DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP

Likes

Attribute	Type	Constraints
like_id	int	AUTO_INCREMENT
user_id	int	REFERENCES Users(user_id) ON DELETE CASCADE
creation_id	int	REFERENCES Creations(creation_id) ON DELETE CASCADE,
created_at	timestamp	TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY (user_id, creation_id)		

5 . References

- **GeeksforGeeks** – Database Normalization (1NF, 2NF, 3NF)
- **W3Schools** – SQL Database Normalization
- **TutorialsPoint** – Normalization in DBMS
- **MySQL Documentation** - https://dev.mysql.com/doc/