3NF Schema with Attributes and Justification

Project: Freelance Marketplace System

This document contains normalized table structures (in 3NF) with attributes, data types, and explanations of normalization decisions.

## Users Table (3NF)

Schema:

CREATE TABLE Users (  
 UserID INT PRIMARY KEY,  
 Username VARCHAR(50),  
 Email VARCHAR(100) UNIQUE,  
 PasswordHash VARCHAR(255),  
 UserType VARCHAR(20) CHECK (UserType IN ('Client', 'Freelancer')),  
 RegistrationDate DATE,  
 LastLoginDate DATE,  
 ProfilePictureURL VARCHAR(255),  
 Bio TEXT  
);

Justification:

All fields depend directly on UserID. Separated Client/Freelancer specifics into other tables to avoid nulls and partial dependencies.

## Freelancers Table (3NF)

Schema:

CREATE TABLE Freelancers (  
 FreelancerID INT PRIMARY KEY REFERENCES Users(UserID),  
 HourlyRate DECIMAL(10,2)  
);

Justification:

Stores freelancer-specific data. Maintains dependency on FreelancerID (which equals UserID).

## Clients Table (3NF)

Schema:

CREATE TABLE Clients (  
 ClientID INT PRIMARY KEY REFERENCES Users(UserID),  
 CompanyName VARCHAR(100)  
);

Justification:

Stores client-specific data separately to avoid mixing roles in the Users table.

## Skills Table (3NF)

Schema:

CREATE TABLE Skills (  
 SkillID INT PRIMARY KEY,  
 SkillName VARCHAR(50),  
 Description TEXT  
);

Justification:

Atomic table of skills. Independent of users, fully normalized.

## FreelancerSkills Table (3NF)

Schema:

CREATE TABLE FreelancerSkills (  
 FreelancerID INT REFERENCES Freelancers(FreelancerID),  
 SkillID INT REFERENCES Skills(SkillID),  
 PRIMARY KEY (FreelancerID, SkillID)  
);

Justification:

Resolves many-to-many between Freelancers and Skills. Prevents redundant data storage.

## Projects Table (3NF)

Schema:

CREATE TABLE Projects (  
 ProjectID INT PRIMARY KEY,  
 ClientID INT REFERENCES Clients(ClientID),  
 Title VARCHAR(100),  
 Description TEXT,  
 CategoryID INT REFERENCES Categories(CategoryID),  
 Budget DECIMAL(10,2),  
 Deadline DATE,  
 PostingDate DATE,  
 Status VARCHAR(20),  
 Visibility VARCHAR(20)  
);

Justification:

Client and category info are stored using foreign keys to avoid duplication. No derived data.

## Categories Table (3NF)

Schema:

CREATE TABLE Categories (  
 CategoryID INT PRIMARY KEY,  
 CategoryName VARCHAR(50),  
 Description TEXT  
);

Justification:

Stores distinct project categories. Avoids repetition across Projects.

## Bids Table (3NF)

Schema:

CREATE TABLE Bids (  
 BidID INT PRIMARY KEY,  
 ProjectID INT REFERENCES Projects(ProjectID),  
 FreelancerID INT REFERENCES Freelancers(FreelancerID),  
 BidAmount DECIMAL(10,2),  
 DeliveryTime INT,  
 BidDate DATE,  
 Status VARCHAR(20)  
);

Justification:

All columns are fully functionally dependent on BidID. No transitive dependencies.

## Contracts Table (3NF)

Schema:

CREATE TABLE Contracts (  
 ContractID INT PRIMARY KEY,  
 BidID INT REFERENCES Bids(BidID),  
 StartDate DATE,  
 EndDate DATE,  
 ContractTerms TEXT,  
 Status VARCHAR(20),  
 PaymentSchedule VARCHAR(100)  
);

Justification:

Data depends on the ContractID and relates directly to one bid.

## Payments Table (3NF)

Schema:

CREATE TABLE Payments (  
 PaymentID INT PRIMARY KEY,  
 ContractID INT REFERENCES Contracts(ContractID),  
 Amount DECIMAL(10,2),  
 PaymentDate DATE,  
 PaymentMethod VARCHAR(50),  
 Status VARCHAR(20),  
 TransactionReference VARCHAR(100)  
);

Justification:

Stores payment information. All fields dependent only on PaymentID.

## Reviews Table (3NF)

Schema:

CREATE TABLE Reviews (  
 ReviewID INT PRIMARY KEY,  
 ContractID INT REFERENCES Contracts(ContractID),  
 ReviewerID INT REFERENCES Users(UserID),  
 RevieweeID INT REFERENCES Users(UserID),  
 Rating INT,  
 Comment TEXT,  
 ReviewDate DATE  
);

Justification:

Reviewer and reviewee are both linked by foreign keys. No redundant fields.

## Messages Table (3NF)

Schema:

CREATE TABLE Messages (  
 MessageID INT PRIMARY KEY,  
 SenderID INT REFERENCES Users(UserID),  
 ReceiverID INT REFERENCES Users(UserID),  
 Content TEXT,  
 SentDate DATETIME,  
 IsRead BIT  
);

Justification:

All attributes depend on MessageID. No derived data like sender name.

## Notifications Table (3NF)

Schema:

CREATE TABLE Notifications (  
 NotificationID INT PRIMARY KEY,  
 UserID INT REFERENCES Users(UserID),  
 Content TEXT,  
 NotificationDate DATE,  
 IsRead BIT,  
 NotificationType VARCHAR(50)  
);

Justification:

Stores only notification-specific data. Linked to Users table via FK.

## Portfolio Table (3NF)

Schema:

CREATE TABLE Portfolio (  
 PortfolioID INT PRIMARY KEY,  
 FreelancerID INT REFERENCES Freelancers(FreelancerID),  
 ProjectTitle VARCHAR(100),  
 ProjectDescription TEXT,  
 SkillsUsed TEXT,  
 CompletionDate DATE,  
 ClientFeedback TEXT,  
 ProjectURL VARCHAR(255)  
);

Justification:

All data specific to one freelancer's past work. Normalized to keep atomic data.

## TransactionHistory Table (3NF)

Schema:

CREATE TABLE TransactionHistory (  
 TransactionID INT PRIMARY KEY,  
 UserID INT REFERENCES Users(UserID),  
 TransactionType VARCHAR(50),  
 Amount DECIMAL(10,2),  
 TransactionDate DATE,  
 Description TEXT,  
 Status VARCHAR(20)  
);

Justification:

Each transaction is uniquely identified and associated with one user.