UniTask

Bases de Dados
Report

Francisco Cardita 97640

Pedro Ferreira 98620

2022/2023

Introduction **3**
Requirements analysis **3**
Features 3
Entities & Attributes 4
Relations 5
ERD 6
ER Model 7
Implementation **8**
Tables 9
Inserts 9
Stored Procedures 9
User-Defined Functions 12
Triggers 13
Indexes 13
Security 14
Abstract Interface 14
Input validation 14
Password hashing 14
Conclusion **15**
Appendix **15**
Appendix 1: Creation of Tables 15
Appendix 2: Inserts 18
Appendix 3: Stored Procedures 35

Introduction

The UniTask platform is designed to help users organize and manage tasks related to their courses. It offers features such as creating, editing, and updating tasks, defining attributes and visibility, aggregating tasks into groups, searching for users, following other users, viewing public tasks, and collaborating work.

This report tackles the requirements analysis and implementation process of the database. For information about the infrastructure and implementation of the server, please refer to `secondary-report.pdf`.

Requirements analysis

Features

- Create tasks by class and edit/update them
- Add various attributes to each task
- Define task visibility
- Balance of finished/unfinished tasks of each class (not implemented)
- Aggregate tasks by groups
- Search for users
- Follow other users
- View other users' public tasks
- See who the user follows and who is following them
- Shared tasks
- Teacher publishes grades of an assignment (not implemented)

Entities & Attributes

Primary keys are underlined

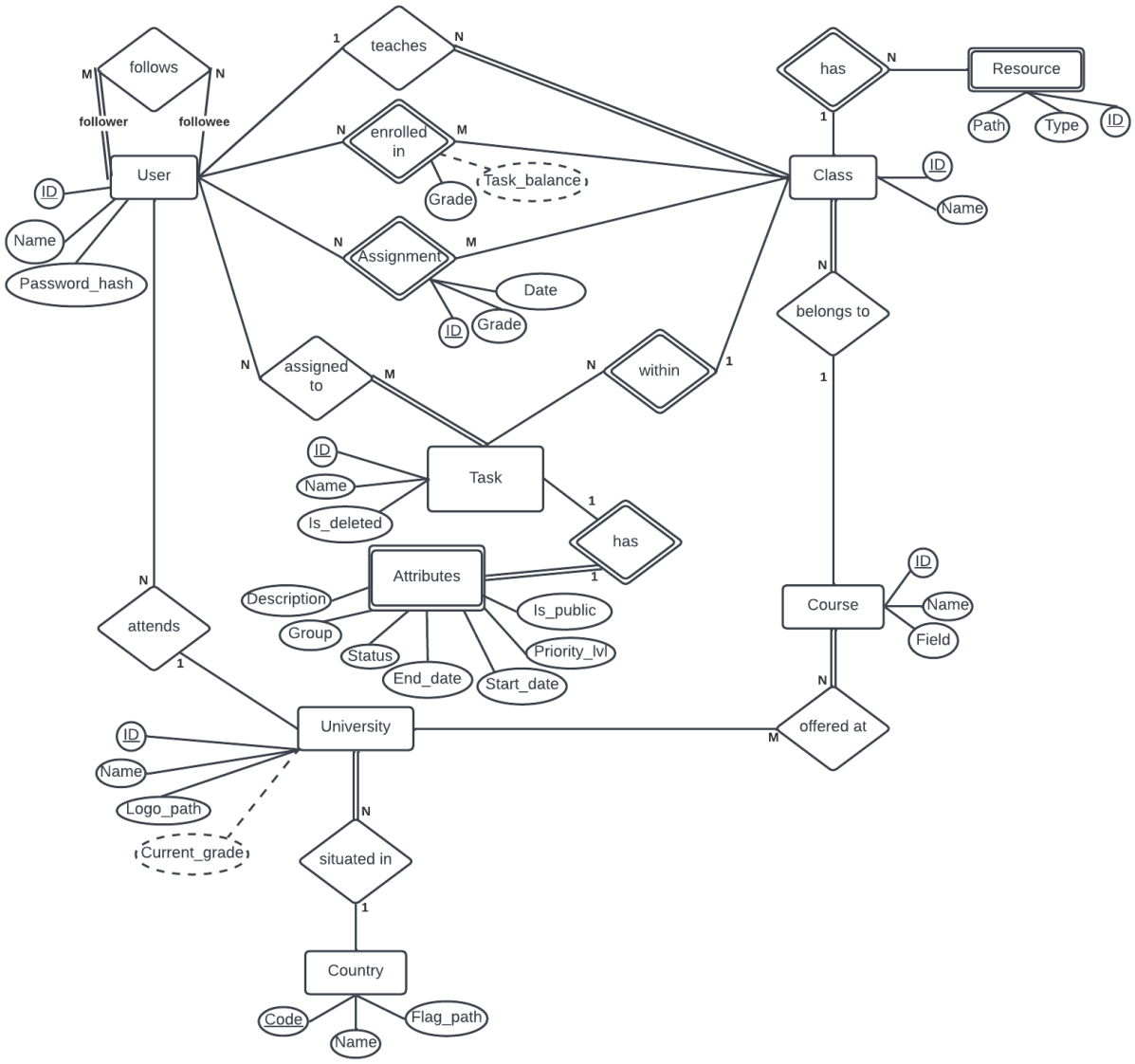
- User
  - ID
  - Name
  - Password_hash
- Task
  - ID
  - Name
  - Is_deleted
- Attributes (weak entity)
  - Description
  - Group
  - Status ('Completed', 'In Progress', 'Pending')
  - Start_date
  - End_date
  - Priority_lvl
  - Is_public
- Enrolled in (relation)
  - Grade
- Assignment (associative entity)
  - Grade
  - Date
  - ID
- Class
  - ID
  - Name
- Course
  - ID
  - Name
  - Field
- Resource (weak entity)
  - ID
  - Path
  - Type (video, img, pdf, etc)
- University
  - ID
  - Name
  - Logo_path
- Country

    - Code
    - Name
    - Flag_path

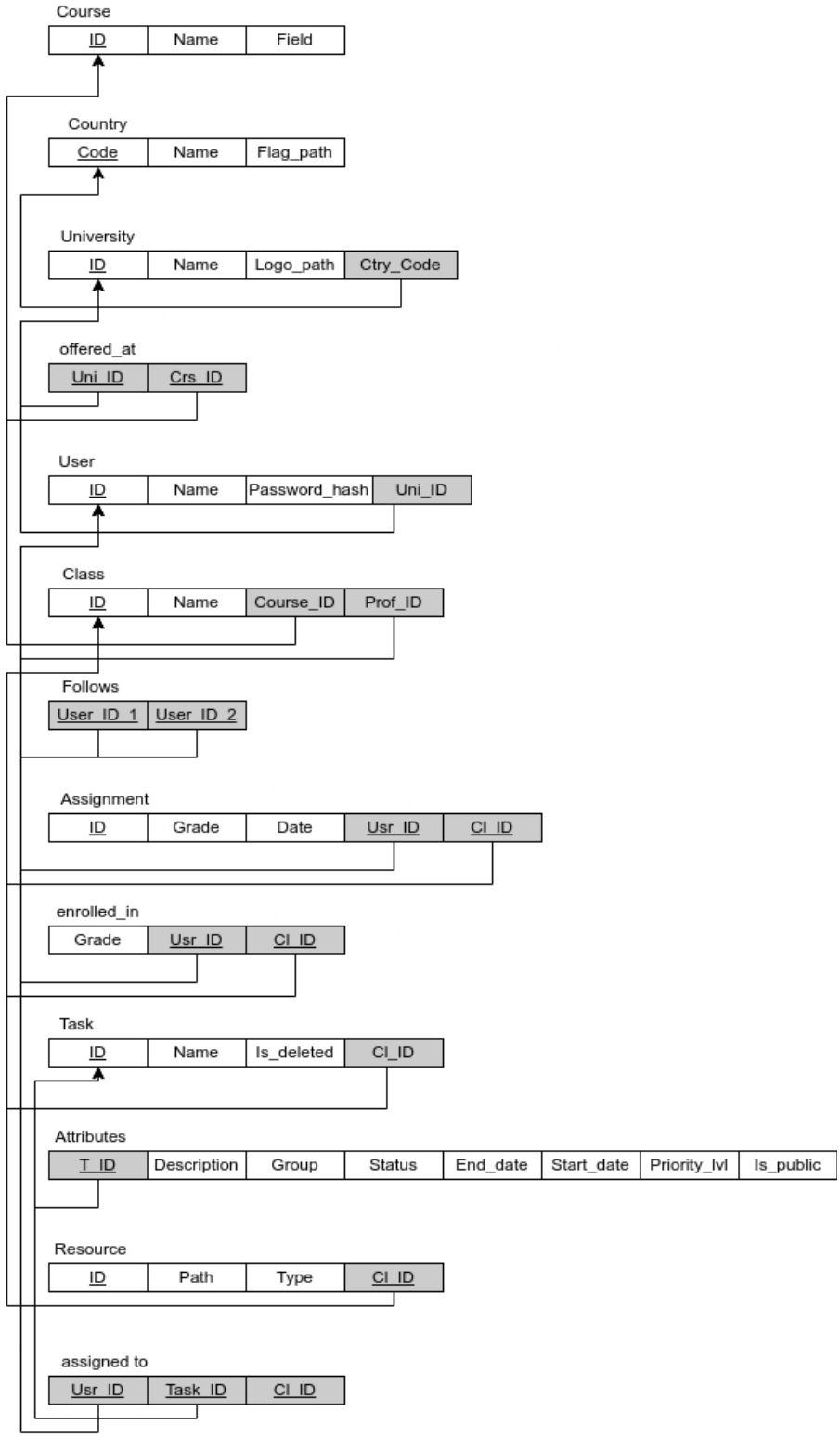
Relations

- User teaches Class (1:N)
- User follows User (N:M)
- User enrolled in Class (N:M)
- User attends University (N:1)
- Task assigned to Student (N:M)
- Task has Attributes (1:1)
- Task within Class (N:1)
- Class has Resource (1:N)
- Class belongs to Course (N:1)
- Course offered at University (N:M)
- University situated in Country (N:1)

ERD


follows
follower
followee
teaches
enrolled in
Grade
Task_balance
Assignment
Date
ID
Grade
User
ID
Name
Password_hash
has
Resource
Path
Type
ID
Class
ID
Name
belongs to
assigned to
within
Task
ID
Name
Is_deleted
has
Attributes
Description
Group
Status
End_date
Start_date
Priority_lvl
Is_public
attends
Course
ID
Name
Field
offered at
University
ID
Name
Logo_path
Current_grade
situated in
Country
Code
Name
Flag_path

ER Model


Course
ID
Name
Field
Country
Code
Name
Flag_path
University
ID
Name
Logo_path
Ctry_Code
offered_at
Uni_ID
Crs_ID
User
ID
Name
Password_hash
Uni_ID
Class
ID
Name
Course_ID
Prof_ID
Follows
User_ID_1
User_ID_2
Assignment
ID
Grade
Date
Usr_ID
Cl_ID
enrolled_in
Grade
Usr_ID
Cl_ID
Task
ID
Name
Is_deleted
Cl_ID
Attributes
T_ID
Description
Group
Status
End_date
Start_date
Priority_lvl
Is_public
Resource
ID
Path
Type
Cl_ID
assigned to
Usr_ID
Task_ID
Cl_ID

Implementation

*The .sql files can be found in **project/db**.*

Tables

Following the ER model above, we created the tables required for the project.

The SQL code for the creation of tables can be found in appendix 1.

Inserts

Using ChatGPT, we inserted dummy values for each table, respecting the constraints declared in their creation.

The SQL code for the inserts can be found in `appendix 2`.

Stored Procedures

The SQL code for the following procedures can be found in `appendix 3`.

---

1. **AssociateTaskWithUser:**
    - Associates a task with a user by inserting a record in the **`assigned_to`** table.
    - Arguments:
        - @task_id (INT)
        - @usr_id (INT)
    - Output: None

2. **DeleteTask:**
    - Deletes a task from the `task` table based on the provided task ID.
    - Arguments:
        - @id (INT)
    - Output: None
3. **FollowUser:**
    - Creates a follower-followee relationship between two users by inserting a record in the `follows` table.
    - Arguments:
        - @follower_id (INT)
        - @followee_id (INT)
    - Output: None
4. **UnfollowUser:**
    - Removes the follower-followee relationship between two users by deleting the corresponding record from the `follows` table.
    - Arguments:
        - @follower_id (INT)
        - @followee_id (INT)
    - Output: None
5. **ListFollowees:**
    - Retrieves a list of users that the specified user is following.
    - Arguments:
        - @usr_id (INT)
    - Output: Table(id, name, uni_id)
6. **ListFollowers:**
    - Retrieves a list of users who are following the specified user.
    - Arguments:
        - @usr_id (INT)
    - Output: Table(id, name, uni_id)
7. **ListTasks:**
    - Retrieves a list of tasks based on the user ID and whether they are public or not. (ordered by **class name** and **task name**)
    - Arguments:
        - @usr_id (INT)

        - @is_public (BIT)
    - Output: Table(task_id, task_name, class_name, description, group, status, start_date, end_date, priority_lvl)
8. **ListUniversities:**
    - Retrieves a list of universities.
    - Arguments: None
    - Output: Table(id, name)
9. **LoginUser:**
    - Validates user login credentials.
    - Arguments:
        - @username (VARCHAR(128))
        - @password (VARCHAR(128))
        - @login_result (BIT OUTPUT)
    - Output: @login_result (BIT)
10. **NewTask:**
    - Creates a new task with the provided details and associates it with a user.
    - Arguments:
        - @task_name (VARCHAR(32))
        - @class_id (INT)
        - @description (VARCHAR(256))
        - @group (VARCHAR(64))
        - @status (VARCHAR(16))
        - @start_date (DATE)
        - @end_date (DATE)
        - @priority_lvl (INT)
        - @is_public (BIT)
        - @usr_id (INT)
    - Output: None
11. **RegisterUser:**
    - Registers a new user by storing their username, password (hashed), and university ID.
    - Arguments:
        - @username (VARCHAR(128))
        - @password (VARCHAR(128))

        - @uni_id (INT)
    - Output: User ID
12. **SearchUser:**
    - Searches for users based on a given user name, and checks if users can follow @usr_id. (ordered by **user name**)
    - Arguments:
        - @user_name (VARCHAR(128))
        - @usr_id (INT)
    - Output: Table(id, name, uni_id, can_follow)
13. **UpdateTask:**
    - Updates a task and its attributes based on the given task ID.
    - Arguments:
        - @task_id (INT)
        - @task_name (VARCHAR(32))
        - @class_id (INT)
        - @description (VARCHAR(256))
        - @group (VARCHAR(64))
        - @status (VARCHAR(16))
        - @start_date (DATE)
        - @end_date (DATE)
        - @priority_lvl (INT)
        - @is_public (BIT)
    - Output: None
14. **ListClasses:**
    - Retrieves a list of classes based on the user ID. If the user has a university, it lists classes from that university; otherwise, it lists all classes.
    - Arguments:
        - @usr_id (INT)
    - Output: Table(id, name)
15. **GetUser:**
    - Retrieves the user's name and university name based on the user ID.
    - Arguments:
        - @usr_id (INT)
    - Output: Table(user_name, uni_name)

User-Defined Functions

- isLoginValid - given the username and password provided by the user when logging in, a variable of type BIT is returned to validate the login

```
CREATE FUNCTION uni_tasks.isLoginValid
(
    @username VARCHAR(128),
    @password VARCHAR(128)
)
RETURNS BIT
AS
BEGIN
    DECLARE @is_valid BIT;

    -- Hash the provided password
    DECLARE @passwordHash VARBINARY(256);
    SET @passwordHash = HASHBYTES('SHA2_256', @password);

    -- Check if the username and hashed password match
    IF EXISTS (SELECT 1 FROM uni_tasks._user WHERE [name] = @username AND
password_hash = @passwordHash)
        SET @is_valid = 1;
    ELSE
        SET @is_valid = 0;

    RETURN @is_valid;
END;
```

Views

Views enhance readability, simplify queries, and provide a logical representation of complex data relationships, making it easier to work with and understand the underlying data.

```
CREATE VIEW uni_tasks.TaskListView AS
SELECT
    t.id AS task_id,
    t.name AS task_name,
    c.name AS class_name,
    a.description,
    a.[group],
    a.[status],
    a.start_date,
    a.end_date,
    a.priority_lvl,
    a.is_public,
    assigned.usr_id
FROM
    uni_tasks.task t
    INNER JOIN uni_tasks.attributes a ON a.t_id = t.id
    INNER JOIN uni_tasks.class c ON t.cl_id = c.id
    INNER JOIN uni_tasks.assigned_to assigned ON t.id = assigned.t_id
WHERE
    t.is_deleted = 0;
```

The `TaskListView` view:

- Uses join conditions (INNER JOIN) to connect the tables based on their relationships.
- It includes a WHERE clause to **filter out deleted tasks** (t.is_deleted = 0).
- Provides a simplified and consolidated view of tasks, eliminating the need for complex joins and conditions when querying task-related information.
- Users can query the **TaskListView** view as if it were a regular table, retrieving the combined attributes of tasks along with their associated class, assigned user, and other relevant details.

Triggers

Trigger for soft delete. Whenever the user wishes to delete a task, it is never completely removed from the database, as the trigger sets the `is_deleted` attribute from table *uni_tasks.task* to 1.

```
CREATE TRIGGER uni_tasks.SoftDeleteTask
ON uni_tasks.task
INSTEAD OF DELETE
AS
BEGIN
    UPDATE t
    SET t.is_deleted = 1
    FROM uni_tasks.task AS t
    INNER JOIN deleted AS d ON t.id = d.id;
END
```

Indexes

To improve query performance in the procedure `SearchUser`, we used a nonclustered index:

```
CREATE NONCLUSTERED INDEX idx_user ON uni_tasks._user([name]);
```

By creating the `idx_user` index on the **name** column, the database management system can efficiently locate rows in the `_user` table based on the user's name. This can speed up queries that involve searching for users whose names match a specific pattern, as indicated by the use of the "LIKE" operator in the `SearchUser` procedure.

Security

Abstract Interface

In our project, the DB is not accessed directly. Instead, we use procedures for every action needed.

This way, we create an abstraction between the server API endpoints and the DB, allowing for a more secure way to handle operations.

Input validation

To prevent SQL Injection attacks from happening, we need to sanitize input.

```
EXEC uni_tasks.GetUser @usr_id=%s, [user_id]
```

Password hashing

Storing passwords in plaintext poses a security risk. We resort to the `SHA2_256` hashing algorithm to hash the password when storing it in the database to mitigate this liability. Upon login, the password inserted by the user is also hashed to be compared to the one in the database, for authentication purposes.

In Stored Procedure `RegisterUser`:

```
SET @passwordHash = HASHBYTES('SHA2_256', @password);
-- Store hash
```

In UDF `isLoginValid`:

```
SET @passwordHash = HASHBYTES('SHA2_256', @password);
IF EXISTS (SELECT 1 FROM uni_tasks._user WHERE ... AND password_hash =
@passwordHash)
      -- Login successful
ELSE
      -- Login failed
```

Conclusion

In this project, we developed a task management platform for university classes. The system allows users to create and manage tasks, search for other users, follow them, and view their public tasks. We implemented security measures such as input validation and password hashing to enhance security. We also created a nonclustered index on the **name** column of the `_user` table to improve query performance in the `SearchUser` procedure. Overall, the project provides a robust and secure task management solution for university classes.

Appendix

Appendix 1: Creation of Tables

```
CREATE TABLE uni_tasks.course(
      [id] [int] IDENTITY(1, 1) NOT NULL,
      [name] [varchar](64) NOT NULL,
      [field] [varchar](64),
      PRIMARY KEY ([id])
);

CREATE TABLE uni_tasks.country(
      [code] [varchar](8) NOT NULL,
      [name] [varchar](64) NOT NULL,
      [flag] [varchar](256),
      PRIMARY KEY ([code])
);

CREATE TABLE uni_tasks.university(
      [id] [int] IDENTITY(1, 1) NOT NULL,
      [name] [varchar](64) NOT NULL,
      [logo_path] [varchar](256),
      [ctry_code] [varchar](8) NOT NULL,
      PRIMARY KEY ([id]),
      FOREIGN KEY ([ctry_code]) REFERENCES uni_tasks.country([code])
);

CREATE TABLE uni_tasks.offered_at(
      [uni_id] [int] NOT NULL,
      [crs_id] [int] NOT NULL,
      PRIMARY KEY ([uni_id], [crs_id]),
      FOREIGN KEY ([uni_id]) REFERENCES uni_tasks.university([id]),
      FOREIGN KEY ([crs_id]) REFERENCES uni_tasks.course([id])
);

CREATE TABLE uni_tasks._user(
      [id] [int] IDENTITY(1, 1) NOT NULL,
      [name] [varchar](128) NOT NULL,
      [password_hash] [varbinary](256) NOT NULL,
      [uni_id] [int],
```

```
      PRIMARY KEY ([id]),
      FOREIGN KEY ([uni_id]) REFERENCES uni_tasks.university([id])
);

CREATE TABLE uni_tasks.class(
      [id] [int] IDENTITY(1, 1) NOT NULL,
      [name] [varchar](64) NOT NULL,
      [crs_id] [int] NOT NULL,
      [prof_id] [int],
      PRIMARY KEY ([id]),
      FOREIGN KEY ([crs_id]) REFERENCES uni_tasks.course([id]),
      FOREIGN KEY ([prof_id]) REFERENCES uni_tasks._user([id])
);

CREATE TABLE uni_tasks.follows(
      [usr_id_followee] [int] NOT NULL,
      [usr_id_follower] [int] NOT NULL,
      PRIMARY KEY ([usr_id_followee], [usr_id_follower]),
      FOREIGN KEY ([usr_id_followee]) REFERENCES uni_tasks._user([id]),
      FOREIGN KEY ([usr_id_follower]) REFERENCES uni_tasks._user([id])
);

CREATE TABLE uni_tasks.assignment(
      [id] [int] IDENTITY(1, 1) NOT NULL,
      [grade] [numeric],
      [date] [date],
      [usr_id] [int] NOT NULL,
      [cl_id] [int] NOT NULL,
      PRIMARY KEY ([id], [usr_id], [cl_id]),
      FOREIGN KEY ([usr_id]) REFERENCES uni_tasks._user([id]),
      FOREIGN KEY ([cl_id]) REFERENCES uni_tasks.class([id])
);

CREATE TABLE uni_tasks.enrolled_in(
      [grade] [int] CHECK ([grade] >= 0 AND [grade] <= 20),
      [usr_id] [int] NOT NULL,
      [cl_id] [int] NOT NULL,
      PRIMARY KEY ([usr_id], [cl_id]),
      FOREIGN KEY ([usr_id]) REFERENCES uni_tasks._user([id]),
      FOREIGN KEY ([cl_id]) REFERENCES uni_tasks.class([id])
);

CREATE TABLE uni_tasks.task(
```

```
	[id] [int] IDENTITY(1, 1) NOT NULL,
	[name] [varchar](64) NOT NULL,
	[cl_id] [int],
	[is_deleted] [bit] NOT NULL DEFAULT 0,
	PRIMARY KEY ([id]),
	FOREIGN KEY ([cl_id]) REFERENCES uni_tasks.class([id])
);

CREATE TABLE uni_tasks.attributes(
	[t_id] [int] NOT NULL,
	[description] [varchar](256),
	[group] [varchar](64),
	[status] [varchar](16) NOT NULL CHECK ([status] IN ('Completed', 'In
Progress', 'Pending')) DEFAULT 'Pending',
	[start_date] [date],
	[end_date] [date],
	[priority_lvl] [int] CHECK([priority_lvl]>=1 AND [priority_lvl]<=5),
	[is_public] [bit] NOT NULL,
	PRIMARY KEY ([t_id]),
	FOREIGN KEY ([t_id]) REFERENCES uni_tasks.task([id])
);

CREATE TABLE uni_tasks._resource(
	[id] [int] IDENTITY(1, 1) NOT NULL,
	[path] [varchar](256) NOT NULL,
	[type] [varchar](16),
	[cl_id] [int] NOT NULL,
	PRIMARY KEY ([id], [cl_id]),
	FOREIGN KEY ([cl_id]) REFERENCES uni_tasks.class([id])
);

CREATE TABLE uni_tasks.assigned_to(
	[usr_id] [int] NOT NULL,
	[t_id] [int] NOT NULL,
	PRIMARY KEY ([usr_id], [t_id]),
	FOREIGN KEY ([usr_id]) REFERENCES uni_tasks._user([id]),
	FOREIGN KEY ([t_id]) REFERENCES uni_tasks.task([id])
);
```

Appendix 2: Inserts

```sql
-- Dummy inserts for uni_tasks.course
INSERT INTO uni_tasks.course (name, field) VALUES
    ('Mathematics', 'Science'),
    ('History', 'Humanities'),
    ('Computer Science', 'Engineering'),
    ('Physics', 'Science'),
    ('Literature', 'Humanities'),
    ('Chemistry', 'Science'),
    ('Biology', 'Science'),
    ('Psychology', 'Social Science'),
    ('Sociology', 'Social Science'),
    ('Philosophy', 'Humanities'),
    ('Art', 'Fine Arts'),
    ('Music', 'Fine Arts'),
    ('Geography', 'Social Science'),
    ('Economics', 'Social Science'),
    ('Political Science', 'Social Science'),
    ('English', 'Humanities'),
    ('Business Administration', 'Business'),
    ('Engineering', 'Engineering'),
    ('Communications', 'Social Science'),
    ('Anthropology', 'Social Science'),
    ('Environmental Science', 'Science'),
    ('Health Sciences', 'Science'),
    ('Education', 'Social Science'),
    ('Foreign Languages', 'Humanities'),
    ('Journalism', 'Social Science'),
    ('Architecture', 'Fine Arts'),
    ('Information Technology', 'Technology'),
    ('Marketing', 'Business'),
    ('Nursing', 'Health Sciences'),
    ('Dance', 'Fine Arts'),
    ('Theater', 'Fine Arts'),
    ('Sports Science', 'Science'),
    ('Criminal Justice', 'Social Science'),
    ('Film Studies', 'Fine Arts'),
    ('Religious Studies', 'Humanities'),
    ('Graphic Design', 'Fine Arts'),
    ('Hospitality Management', 'Business'),
    ('Astronomy', 'Science'),
    ('Fashion Design', 'Fine Arts'),
    ('Public Relations', 'Social Science'),
    ('Social Work', 'Social Science'),
```

```
    ('Linguistics', 'Humanities'),
    ('Urban Planning', 'Social Science'),
    ('Nutrition', 'Health Sciences'),
    ('Veterinary Science', 'Science'),
    ('Law', 'Social Science'),
    ('Human Resources', 'Business'),
    ('Physical Therapy', 'Health Sciences'),
    ('Forensic Science', 'Science');

-- Dummy inserts for uni_tasks.country
INSERT INTO uni_tasks.country (code, name, flag) VALUES
    ('POR', 'Portugal', 'portugal_flag.png'),
    ('USA', 'United States', 'usa_flag.png'),
    ('GBR', 'United Kingdom', 'uk_flag.png'),
    ('CAN', 'Canada', 'canada_flag.png'),
    ('AUS', 'Australia', 'australia_flag.png'),
    ('DEU', 'Germany', 'germany_flag.png'),
    ('FRA', 'France', 'france_flag.png'),
    ('ESP', 'Spain', 'spain_flag.png'),
    ('ITA', 'Italy', 'italy_flag.png'),
    ('JPN', 'Japan', 'japan_flag.png'),
    ('CHN', 'China', 'china_flag.png'),
    ('IND', 'India', 'india_flag.png'),
    ('BRA', 'Brazil', 'brazil_flag.png'),
    ('RUS', 'Russia', 'russia_flag.png'),
    ('MEX', 'Mexico', 'mexico_flag.png'),
    ('ARG', 'Argentina', 'argentina_flag.png'),
    ('COL', 'Colombia', 'colombia_flag.png'),
    ('SAU', 'Saudi Arabia', 'saudi_arabia_flag.png'),
    ('ZAF', 'South Africa', 'south_africa_flag.png'),
    ('EGY', 'Egypt', 'egypt_flag.png'),
    ('NGA', 'Nigeria', 'nigeria_flag.png'),
    ('KEN', 'Kenya', 'kenya_flag.png'),
    ('THA', 'Thailand', 'thailand_flag.png'),
    ('MYS', 'Malaysia', 'malaysia_flag.png'),
    ('SGP', 'Singapore', 'singapore_flag.png'),
    ('KOR', 'South Korea', 'south_korea_flag.png'),
    ('TUR', 'Turkey', 'turkey_flag.png'),
    ('POL', 'Poland', 'poland_flag.png'),
    ('SWE', 'Sweden', 'sweden_flag.png'),
    ('NOR', 'Norway', 'norway_flag.png'),
    ('FIN', 'Finland', 'finland_flag.png'),
    ('NLD', 'Netherlands', 'netherlands_flag.png'),
```

```
    ('BEL', 'Belgium', 'belgium_flag.png'),
    ('AUT', 'Austria', 'austria_flag.png'),
    ('GRC', 'Greece', 'greece_flag.png'),
    ('ISR', 'Israel', 'israel_flag.png'),
    ('NZL', 'New Zealand', 'new_zealand_flag.png'),
    ('CZE', 'Czech Republic', 'czech_republic_flag.png'),
    ('HUN', 'Hungary', 'hungary_flag.png'),
    ('BGR', 'Bulgaria', 'bulgaria_flag.png'),
    ('ROU', 'Romania', 'romania_flag.png'),
    ('SRB', 'Serbia', 'serbia_flag.png'),
    ('HRV', 'Croatia', 'croatia_flag.png'),
    ('DNK', 'Denmark', 'denmark_flag.png'),
    ('ZWE', 'Zimbabwe', 'zimbabwe_flag.png'),
    ('VEN', 'Venezuela', 'venezuela_flag.png'),
    ('PER', 'Peru', 'peru_flag.png'),
    ('PAN', 'Panama', 'panama_flag.png');

-- Dummy inserts for uni_tasks.university
INSERT INTO uni_tasks.university (name, logo_path, ctry_code) VALUES
    ('University of Aveiro', 'ua_logo.png', 'POR'),
    ('California Institute of Technology', 'caltech_university_logo.png',
'USA'),
    ('University of Cambridge', 'cambridge_university_logo.png', 'GBR'),
    ('Imperial College London', 'imperial_university_logo.png', 'GBR'),
    ('University of Toronto', 'toronto_university_logo.png', 'CAN'),
    ('McGill University', 'mcgill_university_logo.png', 'CAN'),
    ('University of Melbourne', 'melbourne_university_logo.png', 'AUS'),
    ('University of Queensland', 'queensland_university_logo.png', 'AUS'),
    ('Technical University of Munich', 'tum_university_logo.png', 'DEU'),
    ('Ludwig Maximilian University of Munich',
'lmu_munich_university_logo.png', 'DEU'),
    ('Sorbonne University', 'sorbonne_university_logo.png', 'FRA'),
    ('École Normale Supérieure', 'ens_university_logo.png', 'FRA'),
    ('University of Barcelona', 'barcelona_university_logo.png', 'ESP'),
    ('Complutense University of Madrid', 'complutense_university_logo.png',
'ESP'),
    ('University of Milan', 'milan_university_logo.png', 'ITA'),
    ('Polytechnic University of Milan', 'polimi_university_logo.png',
'ITA'),
    ('University of Tokyo', 'tokyo_university_logo.png', 'JPN'),
    ('Osaka University', 'osaka_university_logo.png', 'JPN'),
    ('Peking University', 'peking_university_logo.png', 'CHN'),
    ('Stanford University', 'stanford_university_logo.png', 'USA'),
```

```
    ('Indian Institute of Technology Delhi',
'iit_delhi_university_logo.png', 'IND'),
    ('Indian Institute of Science Bangalore',
'iisc_bangalore_university_logo.png', 'IND'),
    ('University of São Paulo', 'saopaulo_university_logo.png', 'BRA'),
    ('Federal University of Rio de Janeiro', 'rio_university_logo.png',
'BRA'),
    ('Lomonosov Moscow State University', 'msu_university_logo.png',
'RUS'),
    ('Novosibirsk State University', 'novosibirsk_university_logo.png',
'RUS'),
    ('National Autonomous University of Mexico',
'unam_university_logo.png', 'MEX'),
    ('Monterrey Institute of Technology and Higher Education',
'monterrey_university_logo.png', 'MEX'),
    ('University of Buenos Aires', 'buenosaires_university_logo.png',
'ARG'),
    ('Pontifical Catholic University of Chile',
'puc_chile_university_logo.png', 'COL'),
    ('King Fahd University of Petroleum and Minerals',
'kfupm_university_logo.png', 'SAU'),
    ('University of Cape Town', 'uct_university_logo.png', 'ZAF'),
    ('American University in Cairo', 'cairo_university_logo.png', 'EGY'),
    ('University of Ibadan', 'ibadan_university_logo.png', 'NGA'),
    ('University of Nairobi', 'nairobi_university_logo.png', 'KEN'),
    ('Chulalongkorn University', 'chulalongkorn_university_logo.png',
'THA'),
    ('University of Malaya', 'malaya_university_logo.png', 'MYS'),
    ('University of Amsterdam', 'amsterdam_university_logo.png', 'NLD'),
    ('University of Sydney', 'sydney_university_logo.png', 'AUS'),
    ('University of Alberta', 'alberta_university_logo.png', 'CAN'),
    ('University of Oslo', 'oslo_university_logo.png', 'NOR'),
    ('University of Helsinki', 'helsinki_university_logo.png', 'FIN'),
    ('University of Copenhagen', 'copenhagen_university_logo.png', 'DNK'),
    ('École Polytechnique', 'polytechnique_university_logo.png', 'FRA'),
    ('University of Vienna', 'vienna_university_logo.png', 'AUT'),
    ('Seoul National University', 'snu_university_logo.png', 'KOR'),
    ('University of Warsaw', 'warsaw_university_logo.png', 'POL'),
    ('University of Belgrade', 'belgrade_university_logo.png', 'SRB'),
    ('University of Zagreb', 'zagreb_university_logo.png', 'HRV'),
    ('Aarhus University', 'aarhus_university_logo.png', 'DNK'),
    ('University of Zimbabwe', 'zimbabwe_university_logo.png', 'ZWE');
```

```sql
-- Dummy inserts for uni_tasks.offered_at
INSERT INTO uni_tasks.offered_at (uni_id, crs_id) VALUES
            (1, 1),
            (1, 2),
            (2, 3),
            (2, 4),
            (3, 5),
            (3, 1),
            (4, 2),
            (4, 3),
            (5, 4),
            (5, 5),
            (6, 1),
            (6, 2),
            (7, 3),
            (7, 4),
            (8, 5),
            (8, 1),
            (9, 2),
            (9, 3),
            (10, 4),
            (10, 5),
            (11, 1),
            (11, 2),
            (12, 3),
            (12, 4),
            (13, 5),
            (13, 1),
            (14, 2),
            (14, 3),
            (15, 4),
            (15, 5),
            (16, 1),
            (16, 2),
            (17, 3),
            (17, 4),
            (18, 5),
            (18, 1),
            (19, 2),
            (19, 3),
            (20, 4),
            (20, 5),
            (21, 1),
```

```
            (21, 2),
            (22, 3),
            (22, 4),
            (23, 5),
            (23, 1),
            (24, 2),
            (24, 3),
            (25, 4),
            (25, 5);

-- Dummy inserts for uni_tasks._user
INSERT INTO uni_tasks._user (name, password_hash, uni_id) VALUES
    ('John Doe', HASHBYTES('SHA2_256', 'password123'), 1),
    ('Jane Smith', HASHBYTES('SHA2_256', 'letmein2023'), 2),
    ('David Johnson', HASHBYTES('SHA2_256', 'securepassword'), 3),
    ('Emily Brown', HASHBYTES('SHA2_256', 'password456'), 4),
    ('Michael Davis', HASHBYTES('SHA2_256', 'password789'), 5),
    ('Jennifer Wilson', HASHBYTES('SHA2_256', 'mypassword'), 6),
    ('Christopher Taylor', HASHBYTES('SHA2_256', 'qwerty123'), 7),
    ('Jessica Anderson', HASHBYTES('SHA2_256', 'pass123word'), 8),
    ('Matthew Martinez', HASHBYTES('SHA2_256', 'hello123'), 9),
    ('Sarah Thompson', HASHBYTES('SHA2_256', 'welcome456'), 10),
    ('Daniel Clark', HASHBYTES('SHA2_256', 'password789'), 11),
    ('Olivia Rodriguez', HASHBYTES('SHA2_256', 'testpassword'), 12),
    ('Andrew Lewis', HASHBYTES('SHA2_256', 'password123'), 13),
    ('Sophia Lee', HASHBYTES('SHA2_256', 'letmein123'), 14),
    ('William Walker', HASHBYTES('SHA2_256', 'password456'), 15),
    ('Ava Hall', HASHBYTES('SHA2_256', 'secure123'), 16),
    ('James Young', HASHBYTES('SHA2_256', 'password789'), 17),
    ('Mia White', HASHBYTES('SHA2_256', 'mypassword'), 18),
    ('Logan Green', HASHBYTES('SHA2_256', 'password123'), 19),
    ('Charlotte King', HASHBYTES('SHA2_256', 'qwerty123'), 20),
    ('Benjamin Baker', HASHBYTES('SHA2_256', 'password456'), 21),
    ('Sophie Turner', HASHBYTES('SHA2_256', 'letmein567'), 22),
    ('Henry Moore', HASHBYTES('SHA2_256', 'password789'), 23),
    ('Ella Davis', HASHBYTES('SHA2_256', 'welcome789'), 24),
    ('Alexander Wilson', HASHBYTES('SHA2_256', 'pass567word'), 25),
    ('Oliver Thompson', HASHBYTES('SHA2_256', 'hello789'), 26),
    ('Chloe Clark', HASHBYTES('SHA2_256', 'test123password'), 27),
    ('Daniel Roberts', HASHBYTES('SHA2_256', 'qwerty567'), 28),
    ('Emma Green', HASHBYTES('SHA2_256', 'secure456'), 29),
    ('Jacob Hall', HASHBYTES('SHA2_256', 'mypassword789'), 30),
    ('Avery Johnson', HASHBYTES('SHA2_256', 'password567'), 31),
```

```
    ('Grace Anderson', HASHBYTES('SHA2_256', 'letmein567'), 32),
    ('Samuel Martinez', HASHBYTES('SHA2_256', 'password123456'), 33),
    ('Aria Taylor', HASHBYTES('SHA2_256', 'qwerty789'), 34),
    ('Liam Baker', HASHBYTES('SHA2_256', 'secure567'), 35),
    ('Lily Young', HASHBYTES('SHA2_256', 'mypassword567'), 36),
    ('Lucas Wilson', HASHBYTES('SHA2_256', 'welcome123'), 37),
    ('Scarlett Moore', HASHBYTES('SHA2_256', 'pass789word'), 38),
    ('Noah Davis', HASHBYTES('SHA2_256', 'hello567'), 39),
    ('Grace Thompson', HASHBYTES('SHA2_256', 'test567password'), 40),
    ('Benjamin Clark', HASHBYTES('SHA2_256', 'qwerty123456'), 41),
    ('Mila Roberts', HASHBYTES('SHA2_256', 'secure567password'), 42),
    ('Jacob Lewis', HASHBYTES('SHA2_256', 'mypassword123456'), 43),
    ('Amelia Anderson', HASHBYTES('SHA2_256', 'password567890'), 44),
    ('Elijah Martinez', HASHBYTES('SHA2_256', 'letmein567890'), 45),
    ('Olivia Taylor', HASHBYTES('SHA2_256', 'welcome567890'), 46),
    ('Lucas Moore', HASHBYTES('SHA2_256', 'pass567890word'), 47),
    ('Ava Davis', HASHBYTES('SHA2_256', 'hello567890'), 48),
    ('Ethan Wilson', HASHBYTES('SHA2_256', 'test567890password'), 49),
    ('Isabella Roberts', HASHBYTES('SHA2_256', 'qwerty567890'), 50);

-- Dummy inserts for uni_tasks.class
INSERT INTO uni_tasks.class (name, crs_id, prof_id) VALUES
    ('English Literature 101', 1, 1),
    ('Chemistry Lab 201', 2, 2),
    ('Computer Science 301', 3, 3),
    ('Mathematics 401', 4, 4),
    ('History of Art 501', 5, 5),
    ('Physics 102', 1, 6),
    ('Economics 202', 2, 7),
    ('Psychology 302', 3, 8),
    ('Business Ethics 402', 4, 9),
    ('Biology 502', 5, 10),
    ('Sociology 103', 1, 11),
    ('Shakespearean Literature 203', 2, 12),
    ('Environmental Science 303', 3, 13),
    ('Art History 403', 4, 14),
    ('Political Science 503', 5, 15),
    ('Introduction to Engineering 104', 1, 16),
    ('Philosophy 204', 2, 17),
    ('Music Theory 304', 3, 1),
    ('Film Studies 404', 4, 2),
    ('Communication Studies 504', 5, 3),
    ('Journalism 105', 1, 4),
```

```
    ('Health Sciences 205', 2, 1),
    ('Geography 305', 3, 2),
    ('Economics 405', 4, 3),
    ('Anthropology 505', 5, 4),
    ('Education 106', 1, 5),
    ('Architecture 206', 2, 6),
    ('Theater Arts 306', 3, 7),
    ('Information Technology 406', 4, 8),
    ('Marketing 506', 5, 9),
    ('Nursing 107', 1, 10),
    ('Criminal Justice 207', 2, 11),
    ('Nutrition and Dietetics 307', 3, 12),
    ('Public Health 407', 4, 13),
    ('Social Work 507', 5, 14),
    ('Sports Science 108', 1, 15),
    ('Library and Information Science 208', 2, 16),
    ('Linguistics 308', 3, 17),
    ('Religious Studies 408', 4, 1),
    ('Dentistry 508', 5, 2),
    ('Veterinary Medicine 109', 1, 3),
    ('Fashion Design 209', 2, 5),
    ('Graphic Design 309', 3, 4),
    ('Journalism and Media Studies 409', 4, 6),
    ('Kinesiology 509', 5, 7),
    ('Computer Engineering 110', 1, 8),
    ('Interior Design 210', 2, 9),
    ('Biochemistry 310', 3, 8),
    ('Fine Arts 410', 4, 5),
    ('International Relations 510', 5, 11);

-- Dummy inserts for uni_tasks.follows
INSERT INTO uni_tasks.follows (usr_id_followee, usr_id_follower) VALUES
            (1, 19),
            (15, 7),
            (14, 5),
            (11, 5),
            (6, 14),
            (13, 2),
            (15, 9),
            (17, 11),
            (3, 4),
            (8, 6),
            (10, 16),
```

```
            (20, 3),
            (14, 9),
            (13, 14),
            (1, 7),
            (4, 6),
            (3, 14),
            (5, 2),
            (18, 19),
            (10, 5),
            (9, 19),
            (2, 20),
            (9, 4),
            (1, 12),
            (13, 7),
            (11, 18),
            (5, 7),
            (12, 14),
            (18, 6),
            (10, 1),
            (7, 6),
            (13, 3),
            (14, 2),
            (3, 1),
            (17, 14),
            (1, 15),
            (19, 10),
            (12, 15),
            (5, 17),
            (14, 16),
            (1, 20),
            (3, 7),
            (4, 10),
            (20, 16),
            (15, 4),
            (6, 18),
            (9, 20),
            (13, 10),
            (2, 9),
            (16, 7);

INSERT INTO uni_tasks.assignment (grade, date, usr_id, cl_id) VALUES
    (85.5, '2023-02-15', 1, 10),
    (75.0, '2023-03-01', 2, 7),
```

```
    (92.5, '2023-02-25', 3, 2),
    (80.0, '2023-03-10', 4, 15),
    (88.5, '2023-02-20', 5, 12),
    (77.0, '2023-03-05', 6, 5),
    (93.5, '2023-02-18', 7, 9),
    (81.0, '2023-03-03', 8, 11),
    (89.5, '2023-02-28', 9, 18),
    (78.0, '2023-03-08', 10, 1),
    (94.5, '2023-02-16', 11, 4),
    (82.0, '2023-03-01', 12, 13),
    (90.5, '2023-02-23', 13, 20),
    (79.0, '2023-03-06', 14, 6),
    (95.5, '2023-02-21', 15, 16),
    (83.0, '2023-02-27', 16, 17),
    (91.5, '2023-02-17', 17, 19),
    (80.0, '2023-03-04', 1, 3),
    (96.5, '2023-02-19', 2, 8),
    (84.0, '2023-03-02', 3, 14),
    (92.5, '2023-02-24', 4, 20),
    (81.0, '2023-03-09', 5, 4),
    (97.5, '2023-02-22', 3, 17),
    (85.0, '2023-02-28', 4, 11),
    (93.5, '2023-02-26', 3, 13),
    (82.0, '2023-03-07', 12, 2),
    (98.5, '2023-02-20', 15, 19),
    (86.0, '2023-03-05', 11, 10),
    (94.5, '2023-02-23', 2, 15),
    (83.0, '2023-03-02', 8, 1),
    (99.5, '2023-02-18', 7, 9),
    (87.0, '2023-03-01', 6, 5);

-- Dummy inserts for uni_tasks.enrolled_in
INSERT INTO uni_tasks.enrolled_in (grade, usr_id, cl_id) VALUES
    (17, 1, 1),
    (15, 2, 2),
    (18, 3, 3),
    (16, 4, 4),
    (17, 5, 5),
    (14, 6, 6),
    (19, 7, 7),
    (15, 8, 8),
    (18, 9, 9),
    (13, 10, 10),
```

```
    (20, 11, 11),
    (16, 12, 12),
    (19, 13, 13),
    (14, 14, 14),
    (20, 15, 15),
    (17, 16, 16),
    (18, 17, 17),
    (15, 18, 18),
    (19, 19, 19),
    (13, 20, 20),
    (20, 21, 21),
    (16, 1, 22),
    (19, 2, 23),
    (14, 3, 24),
    (20, 4, 25),
    (17, 5, 26),
    (18, 6, 27),
    (15, 7, 28),
    (19, 8, 29),
    (13, 9, 30),
    (20, 11, 31),
    (16, 12, 32),
    (19, 1, 33),
    (14, 2, 34),
    (20, 13, 35),
    (17, 14, 36),
    (18, 3, 37),
    (15, 15, 38),
    (19, 16, 39),
    (13, 4, 40),
    (20, 5, 41),
    (16, 6, 42),
    (19, 7, 43),
    (14, 8, 44),
    (20, 9, 45),
    (17, 4, 46),
    (18, 2, 47),
    (15, 7, 48),
    (19, 5, 49),
    (13, 10, 50);

-- Dummy inserts for uni_tasks.task
INSERT INTO uni_tasks.task (name, cl_id, is_deleted) VALUES
```

```
('Research paper on Modern Literature', 1, 0),
('Chemistry Lab Report', 2, 0),
('Computer Science Programming Assignment', 3, 0),
('Mathematics Problem Set', 4, 0),
('History Essay on World War II', 5, 0),
('Physics Experiment Analysis', 6, 0),
('Foreign Language Translation Exercise', 7, 0),
('Psychology Case Study', 8, 0),
('Business Ethics Group Presentation', 9, 0),
('Biology Field Research', 10, 0),
('Sociology Data Analysis Project', 11, 0),
('Literary Critique of Shakespearean Play', 12, 0),
('Environmental Science Research Proposal', 13, 0),
('Art History Visual Analysis', 14, 0),
('Political Science Policy Brief', 15, 0),
('Engineering Design Project', 16, 0),
('Philosophy Argumentative Essay', 17, 0),
('Music Composition Assignment', 18, 0),
('Film Studies Movie Review', 19, 0),
('Communication Studies Public Speaking Task', 20, 0),
('Journalism News Article Writing', 21, 0),
('Health Sciences Case Report', 22, 0),
('Geography Fieldwork Analysis', 23, 0),
('Economics Data Modeling and Analysis', 24, 0),
('Anthropology Cultural Research', 25, 0),
('Education Pedagogy Project', 26, 0),
('Architecture Design Proposal', 27, 0),
('Theater Performance Critique', 28, 0),
('Information Technology System Development', 29, 0),
('Marketing Market Research Study', 30, 0),
('Nursing Clinical Care Plan', 31, 0),
('Criminal Justice Case Analysis', 32, 0),
('Nutrition and Dietetics Dietary Assessment', 33, 0),
('Public Health Epidemiology Report', 34, 0),
('Social Work Community Outreach Project', 35, 0),
('Sports Science Exercise Physiology Experiment', 36, 0),
('Library and Information Science Cataloging Task', 37, 0),
('Linguistics Phonetics Research', 38, 0),
('Religious Studies Comparative Religion Essay', 39, 0),
('Dentistry Oral Health Assessment', 40, 0),
('Veterinary Medicine Animal Clinical Case Study', 41, 0),
('Fashion Design Portfolio Creation', 42, 0),
('Graphic Design Visual Branding Project', 43, 0),
```

```
    ('Journalism Investigative Reporting Task', 44, 0),
    ('Kinesiology Biomechanics Analysis', 45, 0),
    ('Computer Engineering Circuit Design', 46, 0),
    ('Interior Design Space Planning Project', 47, 0),
    ('Biochemistry Laboratory Experiment', 48, 0),
    ('Fine Arts Sculpture Creation', 49, 0),
    ('Political Science International Relations Essay', 50, 0);

-- Dummy inserts for uni_tasks.attributes
INSERT INTO uni_tasks.attributes (t_id, description, [group], status,
start_date, end_date, priority_lvl, is_public) VALUES
    (1, 'Description 1', 'Group 1', 'Completed', '2023-01-01',
'2023-01-31', 3, 1),
    (2, 'Description 2', 'Group 2', 'In Progress', '2023-02-01',
'2023-02-28', 2, 1),
    (3, 'Description 3', 'Group 1', 'Pending', '2023-03-01', '2023-03-31',
4, 0),
    (4, 'Description 4', 'Group 2', 'Pending', '2023-04-01', '2023-04-30',
5, 1),
    (5, 'Description 5', 'Group 1', 'In Progress', '2023-05-01',
'2023-05-31', 1, 0),
    (6, 'Description 6', 'Group 2', 'Completed', '2023-06-01',
'2023-06-30', 3, 1),
    (7, 'Description 7', 'Group 1', 'Pending', '2023-07-01', '2023-07-31',
2, 1),
    (8, 'Description 8', 'Group 2', 'In Progress', '2023-08-01',
'2023-08-31', 4, 0),
    (9, 'Description 9', 'Group 1', 'Completed', '2023-09-01',
'2023-09-30', 5, 1),
    (10, 'Description 10', 'Group 2', 'Pending', '2023-10-01',
'2023-10-31', 1, 0),
    (11, 'Description 11', 'Group 1', 'In Progress', '2023-11-01',
'2023-11-30', 3, 1),
    (12, 'Description 12', 'Group 2', 'Pending', '2023-12-01',
'2023-12-31', 2, 1),
    (13, 'Description 13', 'Group 1', 'Completed', '2024-01-01',
'2024-01-31', 4, 0),
    (14, 'Description 14', 'Group 2', 'In Progress', '2024-02-01',
'2024-02-29', 5, 1),
    (15, 'Description 15', 'Group 1', 'Pending', '2024-03-01',
'2024-03-31', 1, 0),
    (16, 'Description 16', 'Group 2', 'Completed', '2024-04-01',
'2024-04-30', 3, 1),
```

```
    (17, 'Description 17', 'Group 1', 'Pending', '2024-05-01',
'2024-05-31', 2, 1),
    (18, 'Description 18', 'Group 2', 'In Progress', '2024-06-01',
'2024-06-30', 4, 0),
    (19, 'Description 19', 'Group 1', 'Completed', '2024-07-01',
'2024-07-31', 5, 1),
    (20, 'Description 20', 'Group 2', 'Pending', '2024-08-01',
'2024-08-31', 1, 0),
    (21, 'Description 21', 'Group 1', 'In Progress', '2024-09-01',
'2024-09-30', 3, 1),
    (22, 'Description 22', 'Group 2', 'Pending', '2024-10-01',
'2024-10-31', 2, 1),
    (23, 'Description 23', 'Group 1', 'Completed', '2024-11-01',
'2024-11-30', 4, 0),
    (24, 'Description 24', 'Group 2', 'In Progress', '2024-12-01',
'2024-12-31', 5, 1),
    (25, 'Description 25', 'Group 1', 'Pending', '2025-01-01',
'2025-01-31', 1, 0),
    (26, 'Description 26', 'Group 2', 'Completed', '2025-02-01',
'2025-02-28', 3, 1),
    (27, 'Description 27', 'Group 1', 'Pending', '2025-03-01',
'2025-03-31', 2, 1),
    (28, 'Description 28', 'Group 2', 'In Progress', '2025-04-01',
'2025-04-30', 4, 0),
    (29, 'Description 29', 'Group 1', 'Completed', '2025-05-01',
'2025-05-31', 5, 1),
    (30, 'Description 30', 'Group 2', 'Pending', '2025-06-01',
'2025-06-30', 1, 0),
    (31, 'Description 31', 'Group 1', 'In Progress', '2025-07-01',
'2025-07-31', 3, 1),
    (32, 'Description 32', 'Group 2', 'Pending', '2025-08-01',
'2025-08-31', 2, 1),
    (33, 'Description 33', 'Group 1', 'Completed', '2025-09-01',
'2025-09-30', 4, 0),
    (34, 'Description 34', 'Group 2', 'In Progress', '2025-10-01',
'2025-10-31', 5, 1),
    (35, 'Description 35', 'Group 1', 'Pending', '2025-11-01',
'2025-11-30', 1, 0),
    (36, 'Description 36', 'Group 2', 'Completed', '2025-12-01',
'2025-12-31', 3, 1),
    (37, 'Description 37', 'Group 1', 'Pending', '2026-01-01',
'2026-01-31', 2, 1),
    (38, 'Description 38', 'Group 2', 'In Progress', '2026-02-01',
```

```
'2026-02-28', 4, 0),
    (39, 'Description 39', 'Group 1', 'Completed', '2026-03-01',
'2026-03-31', 5, 1),
    (40, 'Description 40', 'Group 2', 'Pending', '2026-04-01',
'2026-04-30', 1, 0),
    (41, 'Description 41', 'Group 1', 'In Progress', '2026-05-01',
'2026-05-31', 3, 1),
    (42, 'Description 42', 'Group 2', 'Pending', '2026-06-01',
'2026-06-30', 2, 1),
    (43, 'Description 43', 'Group 1', 'Completed', '2026-07-01',
'2026-07-31', 4, 0),
    (44, 'Description 44', 'Group 2', 'In Progress', '2026-08-01',
'2026-08-31', 5, 1),
    (45, 'Description 45', 'Group 1', 'Pending', '2026-09-01',
'2026-09-30', 1, 0),
    (46, 'Description 46', 'Group 2', 'Completed', '2026-10-01',
'2026-10-31', 3, 1),
    (47, 'Description 47', 'Group 1', 'Pending', '2026-11-01',
'2026-11-30', 2, 1),
    (48, 'Description 48', 'Group 2', 'In Progress', '2026-12-01',
'2026-12-31', 4, 0),
    (49, 'Description 49', 'Group 1', 'Completed', '2027-01-01',
'2027-01-31', 5, 1),
    (50, 'Description 50', 'Group 2', 'Pending', '2027-02-01',
'2027-02-28', 1, 0);

-- Dummy inserts for uni_tasks._resource
INSERT INTO uni_tasks._resource (path, type, cl_id) VALUES
      ('resource1.pdf', 'pdf', 1),
      ('resource2.docx', 'docx', 1),
      ('resource3.png', 'image', 2),
      ('resource4.pptx', 'pptx', 2),
      ('resource5.pdf', 'pdf', 3),
      ('resource6.docx', 'docx', 3),
      ('resource7.png', 'image', 4),
      ('resource8.pptx', 'pptx', 4),
      ('resource9.pdf', 'pdf', 5),
      ('resource10.docx', 'docx', 5),
      ('resource11.png', 'image', 6),
      ('resource12.pptx', 'pptx', 6),
      ('resource13.pdf', 'pdf', 7),
      ('resource14.docx', 'docx', 7),
      ('resource15.png', 'image', 8),
```

```
      ('resource16.pptx', 'pptx', 8),
      ('resource17.pdf', 'pdf', 9),
      ('resource18.docx', 'docx', 9),
      ('resource19.png', 'image', 10),
      ('resource20.pptx', 'pptx', 10),
      ('resource21.pdf', 'pdf', 11),
      ('resource22.docx', 'docx', 11),
      ('resource23.png', 'image', 12),
      ('resource24.pptx', 'pptx', 12),
      ('resource25.pdf', 'pdf', 13),
      ('resource26.docx', 'docx', 13),
      ('resource27.png', 'image', 14),
      ('resource28.pptx', 'pptx', 14),
      ('resource29.pdf', 'pdf', 15),
      ('resource30.docx', 'docx', 15),
      ('resource31.png', 'image', 16),
      ('resource32.pptx', 'pptx', 16),
      ('resource33.pdf', 'pdf', 17),
      ('resource34.docx', 'docx', 17),
      ('resource35.png', 'image', 18),
      ('resource36.pptx', 'pptx', 18),
      ('resource37.pdf', 'pdf', 19),
      ('resource38.docx', 'docx', 19),
      ('resource39.png', 'image', 20),
      ('resource40.pptx', 'pptx', 20),
      ('resource41.pdf', 'pdf', 21),
      ('resource42.docx', 'docx', 21),
      ('resource43.png', 'image', 22),
      ('resource44.pptx', 'pptx', 22),
      ('resource45.pdf', 'pdf', 23),
      ('resource46.docx', 'docx', 23),
      ('resource47.png', 'image', 24),
      ('resource48.pptx', 'pptx', 24),
      ('resource49.pdf', 'pdf', 25),
      ('resource50.docx', 'docx', 25);

-- Dummy inserts for uni_tasks.assigned_to
INSERT INTO uni_tasks.assigned_to (usr_id, t_id) VALUES
      (1, 1),
      (1, 2),
      (2, 3),
      (2, 4),
      (3, 5),
```

```
    (3, 6),
    (4, 7),
    (4, 8),
    (5, 9),
    (5, 10),
    (6, 11),
    (6, 12),
    (7, 13),
    (7, 14),
    (8, 15),
    (8, 16),
    (9, 17),
    (9, 18),
    (10, 19),
    (10, 20),
    (11, 21),
    (11, 22),
    (12, 23),
    (12, 24),
    (13, 25),
    (13, 26),
    (14, 27),
    (14, 28),
    (15, 29),
    (15, 30),
    (16, 31),
    (16, 32),
    (17, 33),
    (17, 34),
    (18, 35),
    (18, 36),
    (19, 37),
    (19, 38),
    (20, 39),
    (20, 40),
    (21, 41),
    (21, 42),
    (1, 43),
    (5, 44),
    (4, 45),
    (3, 46),
    (7, 47),
    (5, 48),
```

```
      (5, 49),
      (2, 50);
```

Appendix 3: Stored Procedures

```
CREATE PROCEDURE uni_tasks.AssociateTaskWithUser
    @task_id INT,
    @usr_id INT
AS
BEGIN
    INSERT INTO uni_tasks.assigned_to (usr_id, t_id)
    VALUES (@usr_id, @task_id);
END
GO

CREATE PROCEDURE uni_tasks.DeleteTask
    @id INT
AS
BEGIN
    DELETE FROM uni_tasks.task
    WHERE id = @id;
END
GO

CREATE PROCEDURE uni_tasks.FollowUser
    @follower_id INT,
    @followee_id INT
AS
BEGIN
    INSERT INTO uni_tasks.follows (usr_id_follower, usr_id_followee)
    VALUES (@follower_id, @followee_id);

END
GO

CREATE PROCEDURE uni_tasks.UnfollowUser
    @follower_id INT,
    @followee_id INT
```

```
AS
BEGIN
    DELETE FROM uni_tasks.follows
    WHERE usr_id_follower = @follower_id AND usr_id_followee =
@followee_id;
END
GO

CREATE PROCEDURE uni_tasks.ListFollowees
    @usr_id INT
AS
BEGIN
    SELECT u.[id], u.[name], u.[uni_id]
    FROM uni_tasks._user u
    WHERE EXISTS (SELECT 1 FROM uni_tasks.follows WHERE [usr_id_follower] =
@usr_id AND [usr_id_followee] = u.[id]);
END
GO

CREATE PROCEDURE uni_tasks.ListFollowers
    @usr_id INT
AS
BEGIN
    SELECT u.[id], u.[name], u.[uni_id]
    FROM uni_tasks._user u
    WHERE EXISTS (SELECT 1 FROM uni_tasks.follows WHERE [usr_id_follower] =
u.[id] AND [usr_id_followee] = @usr_id);
END
GO

CREATE PROCEDURE uni_tasks.ListTasks
    @usr_id INT,
    @is_public BIT
AS
BEGIN
    IF @is_public = 1
    BEGIN
            SELECT
                  task_id,
            task_name,
            class_name,
            description,
            [group],
```

```
            [status],
            start_date,
            end_date,
            priority_lvl
            FROM
                  uni_tasks.TaskListView
            WHERE
            usr_id = @usr_id
            AND is_public = 1
        ORDER BY
            class_name, task_name;

    END
    ELSE
    BEGIN
            SELECT
                  task_id,
            task_name,
            class_name,
            description,
            [group],
            [status],
            start_date,
            end_date,
            priority_lvl
            FROM
                  uni_tasks.TaskListView
        WHERE
            usr_id = @usr_id
        ORDER BY
            class_name, task_name;

    END
END
GO

CREATE PROCEDURE uni_tasks.ListUniversities
AS
BEGIN
    SELECT [id], [name]
    FROM uni_tasks.university;
END
GO
```

```
CREATE PROCEDURE uni_tasks.LoginUser
    @username VARCHAR(128),
    @password VARCHAR(128),
    @login_result BIT OUTPUT
AS
BEGIN
      SET @login_result = (SELECT uni_tasks.isLoginValid(@username,
@password))
END;
GO

CREATE PROCEDURE uni_tasks.NewTask
    @task_name VARCHAR(32),
    @class_id INT,
    @description VARCHAR(256),
    @group VARCHAR(64),
    @status VARCHAR(16),
    @start_date DATE,
    @end_date DATE,
    @priority_lvl INT,
    @is_public BIT,
    @usr_id INT
AS
BEGIN
    DECLARE @task_id INT;

    -- Insert new task
    INSERT INTO uni_tasks.task (name, cl_id)
    VALUES (@task_name, @class_id);

    -- Get the ID of the newly inserted task
    SET @task_id = SCOPE_IDENTITY();
    PRINT 'Task ID: ' + CONVERT(VARCHAR(10), @task_id);

    -- Insert attributes for the task
    INSERT INTO uni_tasks.attributes (t_id, description, [group], [status],
start_date, end_date, priority_lvl, is_public)
    VALUES (@task_id, @description, @group, @status, @start_date,
@end_date, @priority_lvl, @is_public);

    -- Associate user with the task
    EXEC uni_tasks.AssociateTaskWithUser @task_id, @usr_id;
```

```
END
GO

CREATE PROCEDURE uni_tasks.RegisterUser
    @username VARCHAR(128),
    @password VARCHAR(128),
      @uni_id INT
AS
BEGIN
    -- Hash the password
    DECLARE @passwordHash VARBINARY(256);
    SET @passwordHash = HASHBYTES('SHA2_256', @password);

    -- Store the hashed password in the database
    INSERT INTO uni_tasks._user ([name], password_hash, uni_id)
    VALUES (@username, @passwordHash, @uni_id);

    -- Return the inserted user record
    SELECT id FROM uni_tasks._user WHERE id = SCOPE_IDENTITY();
END
GO

CREATE PROCEDURE uni_tasks.SearchUser
    @user_name VARCHAR(128),
    @usr_id INT
AS
BEGIN
    -- Based on @usr_id, check which users he can follow
    SELECT u.[id], u.[name], u.[uni_id],
        CASE
            WHEN EXISTS (SELECT 1 FROM uni_tasks.follows WHERE
[usr_id_followee] = u.[id] AND [usr_id_follower] = @usr_id) THEN 0
            ELSE 1
        END AS can_follow
    FROM uni_tasks._user u
    WHERE u.[id] <> @usr_id -- Exclude the entry with @usr_id
        AND u.[name] LIKE '%' + @user_name + '%'
    ORDER BY u.[name];
END
GO

CREATE PROCEDURE uni_tasks.UpdateTask
    @task_id INT,
```

```
    @task_name VARCHAR(32),
    @class_id INT,
    @description VARCHAR(256),
    @group VARCHAR(64),
    @status VARCHAR(16),
    @start_date DATE,
    @end_date DATE,
    @priority_lvl INT,
    @is_public BIT
AS
BEGIN
    -- Update task
    UPDATE uni_tasks.task
    SET name = @task_name, cl_id = @class_id
    WHERE id = @task_id;

    -- Update task attributes
    IF EXISTS (SELECT 1 FROM uni_tasks.attributes WHERE t_id = @task_id)
    BEGIN
        UPDATE uni_tasks.attributes
        SET description = @description,
            [group] = @group,
            [status] = @status,
            start_date = @start_date,
            end_date = @end_date,
            priority_lvl = @priority_lvl,
            is_public = @is_public
        WHERE t_id = @task_id;
    END
    ELSE
    BEGIN
        INSERT INTO uni_tasks.attributes (t_id, description, [group],
[status], start_date, end_date, priority_lvl, is_public)
        VALUES (@task_id, @description, @group, @status, @start_date,
@end_date, @priority_lvl, @is_public);
    END
END
GO

CREATE PROCEDURE uni_tasks.ListClasses
      @usr_id INT
AS
BEGIN
```

```
      IF EXISTS(SELECT 1 FROM uni_tasks._user WHERE [id] = @usr_id AND
[uni_id] IS NOT NULL)
      BEGIN
            -- User has a university, list classes from the university
            SELECT cl.[id] AS [id], cl.[name] AS [name]
            FROM uni_tasks.class AS cl
            INNER JOIN uni_tasks.offered_at AS o ON o.crs_id = cl.crs_id
            INNER JOIN uni_tasks._user AS u ON u.uni_id = o.uni_id
            WHERE u.[id] = @usr_id
      END
      ELSE
      BEGIN
            -- User does not have a university, list all classes
            SELECT [id], [name]
            FROM uni_tasks.class
      END
END
GO

CREATE PROCEDURE uni_tasks.GetUser
    @usr_id INT
AS
BEGIN
    -- Return user name and university name
    SELECT u.[name], un.[name]
    FROM uni_tasks._user u
    INNER JOIN uni_tasks.university un ON u.uni_id = un.id
    WHERE u.[id] = @usr_id;
END
```