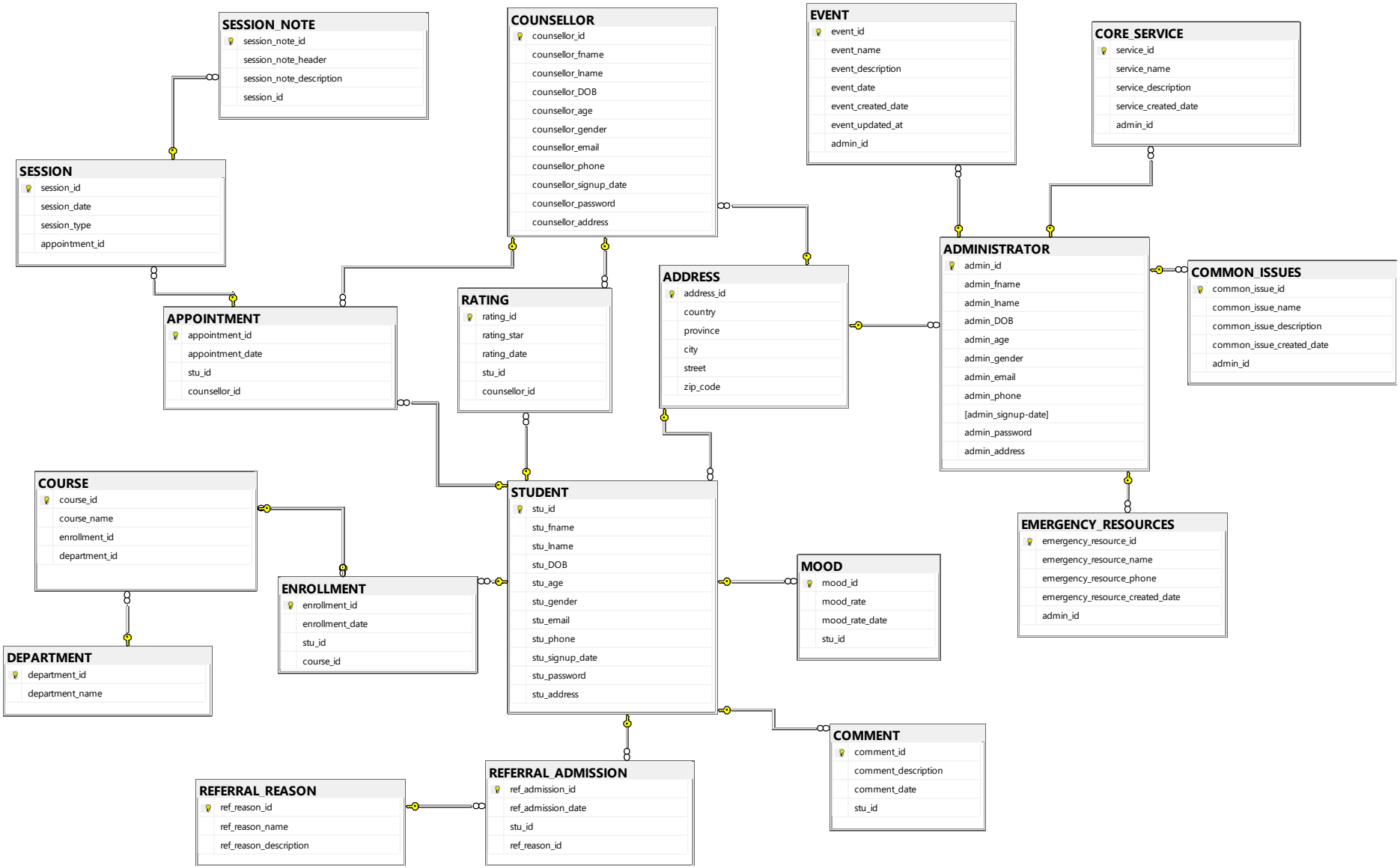
System Design Documentation.

Table of Content.

ENTITY	BUSINESS RULE
System Admin	System admin can create, record/read, update, or delete profiles of any registered users.
	System admin can manage and maintain accurate information within the system.
	System admin can view and manage report tools for metrics and analytics management.
	System admin can create important announcement notification for important events or notices.
	System admin can add, update, delete resources on system, such as adding new services, removing services, or updating services provided by department.
Counsellor	Counsellor can accept, view, or cancel booking from student.
	Counsellor can respond to chats or messages from students within the system chat feature.
	Counsellor can view ratings and feedback from students regarding sessions.
	Counsellor can make or take notes about a session between them and students. Documenting the gradual progression of the student's development, reason for attending counselling, and documenting minutes of the consultation.

Student	Student can schedule or make appointment and update content of session.
	Student can chat anonymously within the app, enabling hesitant or students who fear stigma associated with counselling.
	Student can manage their profiles, enabling students update, modify, or delete their accounts.
	Students can look or view information about coming events and workshops.
	Students can rate and comment counsellors and services offered by department.
	Student can ask questions regarding how to use, functionality, and how to use the app.

6.2. Entity Relationship Diagrams.


SESSION_NOTE
session_note_id
session_note_header
session_note_description
session_id
COUNSELLOR
counsellor_id
counsellor_fname
counsellor_lname
counsellor_DOB
counsellor_age
counsellor_gender
counsellor_email
counsellor_phone
counsellor_signup_date
counsellor_password
counsellor_address
EVENT
event_id
event_name
event_description
event_date
event_created_date
event_updated_at
admin_id
CORE_SERVICE
service_id
service_name
service_description
service_created_date
admin_id
SESSION
session_id
session_date
session_type
appointment_id
ADMINISTRATOR
admin_id
admin_fname
admin_lname
admin_DOB
admin_age
admin_gender
admin_email
admin_phone
[admin_signup-date]
admin_password
admin_address
COMMON_ISSUES
common_issue_id
common_issue_name
common_issue_description
common_issue_created_date
admin_id
APPOINTMENT
appointment_id
appointment_date
stu_id
counsellor_id
RATING
rating_id
rating_star
rating_date
stu_id
counsellor_id
ADDRESS
address_id
country
province
city
street
zip_code
COURSE
course_id
course_name
enrollment_id
department_id
STUDENT
stu_id
stu_fname
stu_lname
stu_DOB
stu_age
stu_gender
stu_email
stu_phone
stu_signup_date
stu_password
stu_address
EMERGENCY_RESOURCES
emergency_resource_id
emergency_resource_name
emergency_resource_phone
emergency_resource_created_date
admin_id
MOOD
mood_id
mood_rate
mood_rate_date
stu_id
ENROLLMENT
enrollment_id
enrollment_date
stu_id
course_id
DEPARTMENT
department_id
department_name
COMMENT
comment_id
comment_description
comment_date
stu_id
REFERRAL_ADMISSION
ref_admission_id
ref_admission_date
stu_id
ref_reason_id
REFERRAL_REASON
ref_reason_id
ref_reason_name
ref_reason_description

6.3. Structural Design.

ENTITY NAME	ATTRIBUTE	DATA TYPE	DESCRITPION
ADDRESS	admin_id	BIGINT	Unique identifier (Primary Key) for address tuple.
	country	VARCHAR (50)	Country column for storing all countries.
	province	VARCHAR (50)	Province column for storing province.
	street	VARCHAR (50)	Street address collection of users within the system
	zip_code	INTEGER	Zip code collection information for all registered system users.
ADMINISTRATOR	admin_id	BIGINT	Unique identity (Primary Key) for system administrator tuple.
	admin_fname	VARCHAR (50)	First name of the system administrator.
	admin_lname	VARCHAR (50)	Lastname of the system administrator.
	admin_DOB	DATE	Date of birth for the system administrator entity.
	admin_age	INTGER	A derived value based on DATEDIFF () function, calculating difference between admin_DOB year and current year.
	admin_gender	CHAR (1)	Gender indicator, value can be either M or F.
	admin_phone	VARCHAR (10)	Phone number for admin can only contain 10 characters
	admin_email	VARCHAR (50)	Email address of all registered users.
	admin_signup_date	DATE	Automated date captured on registration.
	admin_password	VARBINARY (200)	Encrypted password collection for all users.

COUNSELLOR	counsellor_id	BIGINT	Unique identity (Primary Key) for a counsellor tuple.
	counsellor_fname	VARCHAR (50)	First name of counsellor.
	counsellor_lname	VARCHAR (50)	Lastname of counsellor.
	counsellor_DOB	DATE	Date of birth of teacher entity.
	counsellor_age	INTEGER	A derived value based on DATEDIFF () function, calculating difference between counsellor_DOB year and current year.
	counsellor_gender	CHAR (1)	Gender indicator, value can be either M or F.
	counsellor_phone	VARCHAR (10)	Phone number for counsellor can only contain 10 characters
	counsellor_email	VARCHAR (50)	Email address of counsellor.
	counsellor_signup_date	DATE	Automated date captured on registration.
	counsellor_password	VARBINARY (200)	Encrypted password for counsellor entity.
STUDENT	stu_id	BIGINT	Unique identifier (Primary Key) for a student tuple.
	stu_fname	VARCHAR (50)	First name of student.
	stu_lname	VARCHAR (50)	Lastname of student.
	stu_DOB	DATE	Date of birth of student entity.
	stu_age	INTEGER	A derived value based on DATEDIFF () function, calculating difference between stu_DOB year and current year.
	stu_gender	CHAR (1)	Gender indicator, value can either be M of F.
	stu_phone	VARCHAR (10)	Phone number for student can only contain 10 characters
	stu_email	VARCHAR (50)	Email address for student.

	stu_signup_date	DATE	Automated date captured on registration.
	stu_password	VARBINARY (200)	Encrypted password for a student entity.
EVENT	event_id	BIGINT	Unique identifier (Primary Key) for a entity created.
	event_name	VARCHAR (50)	Title or name of the event.
	event_description	TEXT	Descriptive information about event.
	event_date	DATE	Date in which event will be held.
	event_created_date	DATE	Date in which event information was inserted into database. This is metadata.
	event_updated_date	DATE	Date in which there was modification on content of event.
	admin_id	BIGINT	Foreign key of the admin.
CORE_SERVICE	service_id	BIGINT	Unique identifier (Primary Key) for a core service offered within the department of counselling.
	service_name	VARCHAR (50)	Title or name of core service.
	service_description	TEXT	Descriptive information about core service.
	service_created_date	DATE	Date in which information was inserted into database. Metadata data.
	admin_id	BIGINT	Foreign key of the system admin.
COMMON_ISSUE	common_issue_id	BIGINT	Unique identifier (Primary Key) for a common issue entity in database.
	common_issue_name	VARCHAR (50)	Title or name of the common issue.
	common_issue_decription	TEXT	Descriptive information about the common issue.
	common_issue_created_date	DATE	Date in which information was inserted into database. Metadata data.

	admin_id	BIGINT	Foreign key of system admin.
EMERGENCY_RESOURCE	emergency_resource_id	BIGINT	Unique identifier (Primary Key) for an emergency resource in database.
	emergency_resource_name	VARCHAR (50)	Title or name of the emergency resource.
	emergency_resource_description	TEXT	Descriptive information about emergency resource.
	emergency_resource_phone		Phone number for the emergency resource entity.
	emergency_resource_created_date	DATE	Date in which the emergency resource was inserted into database. Metadata data.
	admin_id	BIGINT	Foreign key of system admin
RATING	rating_id	INTEGER	Unique identifier (Primary Key) for a rating entity.
	rating_star	CHAR (1)	Set of predefined value ranging from 1 to 5 in which students can select to rate a counsellor.
	rating_date	DATE	Automatic captured date of when rating was created.
	stu_id	BIGINT	Foreign key of student entity who made rating.
	counsellor_id	BIGINT	Counsellor which rated by student.
APPOINTMENT	appointment_id	BIGINT	Unique identifier (Primary key) for a appointment entity created.
	appointment_date	DATE	Date in which the appointment as requested by student.
	stu_id	BIGINT	Foreign key referencing student entity.
	counsellor_id	BIGINT	Foreign key referencing counsellor entity.

SESSION	session_id	BIGINT	Unique identifier (Primary key) for a session entity created.
	session_date	DATE	Date to which session will be conducted or take place.
	session_type	VARCHAR (50)	Predefined values in which how session can be conducted Sessions can either be Physical or Virtual.
	appointment_id	BIGINT	Foreign key referencing appointment entity.
SESSION_NOTE	session_note_id	BIGINT	Unique identifier (Primary key) for a session note created.
	session_note_header	VARCHAR (50)	Header or title of the session note.
	session_note_description	TEXT	Descriptive information observed by counsellor during session with client (student).
	session_id	BIGINT	Foreign key referencing the session entity.
COMMENT	comment_id	BIGINT	Unique identifier (Primary key) for a comment created.
	comment_description	TEXT	Description of the comment.
	comment_date	DATE	Date on which the comment was created by student.
	stu_id	BIGINT	Foreign key referencing student entity.
MOOD	mood_id	BIGINT	Unique identifier (Primary key) for a mood response by a student.
	mood_rate	VRCHAR (50)	A set of predefined value to which the student can choose from to best represent their current state of mind or

			mood. The values are Very Sad, Sad, Normal, Happy, Very Happy.
	mood_rate_date	DATE	Automated date captured when student responds to mood checker notification.
	stu_id	BIGINT	Foreign key referencing student entity.
ENROLLMENT	enrolment_id	BIGINT	Unique identifier (Primary key) for an enrolment entity.
	enrolment_date	DATE	Automat date captured when student makes or books an appointment.
	stu_id	BIGINT	Foreign key referencing student entity.
	course_id	BIGINT	Foreign key referencing course entity.
COURSE	course_id	BIGINT	Unique identifier (Primary Key) for a course entity.
	course_name	VARCHAR (50)	Name for course which must be unique.
	department_id	INTEGER	Foreign key referencing department entity.
DEPARTMENT	department_id	INTEGER	Unique identifier (Primary key) for a department entity.
	department_name	VARCHAR (50)	Name of the departments for which students came from, or their causes fall under. This information will provide data such as finding which faculty has the highest number of students enrolled into counselling.
REFERRAL_ADMISSION	ref_admission_id	BIGINT	Unique identifier (Primary key)
	ref_admission_date	DATE	Automatic date captured on admission
	stu_id	BIGINT	Foreign key references student entity
	ref_reason_id	BIGINT	Foreign key referencing referral reason

REFERRAL_REASON	ref_reason_id	BIGINT	Unique identifier (Primary key) for referral reason entity.
	ref_reason_name	VARCHAR (50)	Name or title for why student is attending counselling.
	ref_reason_description	TEXT	Descriptive information regarding the reason behind the students' reasons for referral admission.
FAQ	faq_id	INTEGER	Unique identifier (Primary key) for FAQ by a student.
	faq_question	TEXT	Descriptive information about the question.
	faq_created_date	DATE	Automat date captured when user asks question.
	stu_id	BIGINT	Foreign key referencing student entity.
FAQ_RESPONCE	faq_response_id	INTEGER	Unique identifier (Primary key) for a FAQ response.
	faq_response_header	VARCHAR (50)	Header of which question the admins is responding to.
	faq_response_description	TEXT	Descriptive information answering the student inquiry.
	faq_response_date	DATE	Automatic date captured when admin answers or respond to student FAQ.
	faq_question_id	INTEGER	Foreign key referencing the FAQ question of student.
	admin_id	BIGINT	Foreign key referencing system admin.

6.4. Database Structural SQL.

ADDRESS RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.name='ADDRESS')
CREATE TABLE [ADDRESS]
(
 [address_id] INTEGER IDENTITY (0,1) NOT NULL,
 [country] VARCHAR (50) NOT NULL,
 [province] VARCHAR (50) NOT NULL,
 [city]   VARCHAR (50) NOT NULL,
 [street] VARCHAR (50) NOT NULL,
 [zip_code] INTEGER NOT NULL,

 CONSTRAINT [Address_address_id] PRIMARY KEY CLUSTERED ([address_id] ASC)
);
  CREATE INDEX address_indx ON ADDRESS (city)
```

STUDENT RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.name='STUDENT')
CREATE TABLE STUDENT
(
      [stu_id] BIGINT IDENTITY (22200,1) NOT NULL,
      [stu_fname] VARCHAR (50) NOT NULL,
      [stu_lname] VARCHAR (50) NOT NULL,
      [stu_DOB] DATE NOT NULL,
      [stu_age] AS (DATEDIFF (year, stu_DOB, GETDATE ())),
      [stu_gender] CHAR (1) CHECK (stu_gender IN ('M','F')) NOT NULL,
      [stu_email] VARCHAR (50) NOT NULL,
      [stu_phone] VARCHAR (10) NOT NULL,
      [stu_signup_date] DATETIME DEFAULT GETDATE (),
      [stu_password] VARBINARY (200),
      [stu_address] INTEGER

      CONSTRAINT [stu_address_fk] FOREIGN KEY REFERENCES ADDRESS (address_id)
      CONSTRAINT [STUDENT_stu_id] PRIMARY KEY CLUSTERED([stu_id] ASC)
);
 CREATE INDEX student_index ON STUDENT(stu_fname);
```

COUNSELLOR RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='COUNSELLOR')
CREATE TABLE COUNSELLOR
(
 [counsellor_id] BIGINT IDENTITY(200220, 1),
 [counsellor_fname] VARCHAR(50) NOT NULL,
 [counsellor_lname] VARCHAR(50) NOT NULL,
 [counsellor_DOB] DATE NOT NULL,
 [counsellor_age] AS(DATEDIFF(year, counsellor_DOB, GETDATE())),
 [counsellor_gender] CHAR(1) CHECK(counsellor_gender IN ('M','F')) NOT NULL,
 [counsellor_email] VARCHAR(50) NOT NULL UNIQUE,
 [counsellor_phone] VARCHAR(10) NOT NULL UNIQUE,
 [counsellor_signup_date] DATE DEFAULT GETDATE(),
 [counsellor_password] VARBINARY(200) NOT NULL,
 [counsellor_address] INTEGER

 CONSTRAINT[counsellor_address_fk] FOREIGN KEY REFERENCES ADDRESS(address_id)
 CONSTRAINT[COUNSELLOR_counsellor_index] PRIMARY KEY([counsellor_id] ASC)
);
 CREATE INDEX counsellor_index ON COUNSELLOR(counsellor_fname);
```

ADMINISTRATION

```
 IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='ADMINISTRATOR')
 CREATE TABLE ADMINISTRATOR(
      [admin_id] BIGINT IDENTITY(1011100,1),
      [admin_fname] VARCHAR(50) NOT NULL,
      [admin_lname] VARCHAR(50) NOT NULL,
      [admin_DOB] DATE NOT NULL,
      [admin_age] AS(DATEDIFF(year, admin_DOB, GETDATE())),
      [admin_gender] CHAR(1) CHECK( admin_gender IN ('M','F')) NOT NULL,
      [admin_email] VARCHAR(50) NOT NULL UNIQUE,
      [admin_phone] VARCHAR(10) NOT NULL UNIQUE,
      [admin_signup-date] DATE DEFAULT GETDATE(),
      [admin_password] VARBINARY(200) NOT NULL,
      [admin_address] INTEGER

      CONSTRAINT[admin_address_fk] FOREIGN KEY REFERENCES ADDRESS(address_id)
```

```
      CONSTRAINT[ADMIN_admin_index] PRIMARY KEY CLUSTERED([admin_id] ASC)
 );
CREATE INDEX admin_index ON ADMINISTRATOR(admin_fname);
```

TABLE_EVENT

```
IF NOT EXISTS(SELECT * FROM sys.tables t WHERE T.NAME='EVENT')
 CREATE TABLE EVENT(
      [event_id] BIGINT IDENTITY(10930, 1),
      [event_name] VARCHAR(50) NOT NULL,
      [event_description] TEXT NOT NULL,
      [event_date] DATE NOT NULL,
      [event_created_date] DATE DEFAULT GETDATE(),
      [event_updated_at] DATE DEFAULT GETDATE(),
      [admin_id] BIGINT

      CONSTRAINT[EVENT_admin_id_fk] FOREIGN KEY REFERENCES ADMINISTRATOR(admin_id)
      CONSTRAINT[EVENT_event_id] PRIMARY KEY CLUSTERED([event_id] ASC)
 );
  CREATE INDEX event_index ON EVENT(event_name);
```

CORE_SERVICE

```
IF NOT EXISTS(SELECT * FROM sys.tables t WHERE t.NAME='CORE_SERVICE')
 CREATE TABLE CORE_SERVICE(
      [service_id] BIGINT IDENTITY(806640,1),
      [service_name] VARCHAR(50) NOT NULL,
      [service_description] TEXT NOT NULL,
      [service_created_date] DATE DEFAULT GETDATE(),
      [admin_id] BIGINT

      CONSTRAINT[SERVICE_admin_id_fk] FOREIGN KEY REFERENCES ADMINISTRATOR(admin_id)
      CONSTRAINT[SERVICE_service_id] PRIMARY KEY CLUSTERED([service_id]  ASC)
 );
 CREATE INDEX service_index ON CORE_SERVICE(service_name);
```

COMMON_ISSUE RELATION

```
 IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='COMMON_ISSUES')
  CREATE TABLE COMMON_ISSUES(
      [common_issue_id] BIGINT IDENTITY(20500, 1),
      [common_issue_name] VARCHAR(50) NOT NULL UNIQUE,
      [common_issue_description] TEXT NOT NULL,
      [common_issue_created_date] DATE DEFAULT GETDATE(),
      [admin_id] BIGINT

      CONSTRAINT[ISSUE_admin_id_fk] FOREIGN KEY REFERENCES ADMINISTRATOR(admin_id)
      CONSTRAINT[ISSUE_issue_id] PRIMARY KEY CLUSTERED([common_issue_id] ASC)
  );
   CREATE INDEX issue_index ON COMMON_ISSUES(common_issue_name);
```

EMERGENCY RESOURCE RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='EMERGENCY_RESOURCES')
 CREATE TABLE EMERGENCY_RESOURCES(
      [emergency_resource_id] BIGINT IDENTITY(90460, 1),
      [emergency_resource_name] VARCHAR(50) NOT NULL,
      [emergency_resource_phone] VARCHAR(20) NOT NULL UNIQUE,
      [emergency_resource_created_date] DATE DEFAULT GETDATE(),
      [admin_id] BIGINT

      CONSTRAINT[EMERGENCY_RESOURCE_admin_id] FOREIGN KEY REFERENCES ADMINISTRATOR(admin_id)
      CONSTRAINT[EMERGENCY_emergency_resource_id] PRIMARY KEY CLUSTERED([emergency_resource_id] ASC)
 );
    CREATE INDEX emergency_index ON EMERGENCY_RESOURCES(emergency_resource_name);
```

RATING

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='RATING')
 CREATE TABLE RATING(
      [rating_id] BIGINT IDENTITY(0,1),
      [rating_star] CHAR(1) CHECK(rating_star IN ('1','2','3','4','5')),
      [rating_date] DATE DEFAULT GETDATE(),
      [stu_id] BIGINT FOREIGN KEY REFERENCES STUDENT(stu_id),
```

```
      [counsellor_id] BIGINT FOREIGN KEY REFERENCES COUNSELLOR(counsellor_id)

      CONSTRAINT[RATING_rating_id] PRIMARY KEY CLUSTERED([rating_id] ASC)
 );
  CREATE INDEX rating_index ON RATING(rating_star);
```

APPOINTMENT

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='APPOINTMENT')
  CREATE TABLE APPOINTMENT(
      [appointment_id] BIGINT IDENTITY(40770, 1),
      [appointment_date] DATE DEFAULT GETDATE(),
      [stu_id] BIGINT FOREIGN KEY REFERENCES STUDENT(stu_id),
      [counsellor_id] BIGINT FOREIGN KEY REFERENCES COUNSELLOR(counsellor_id)

      CONSTRAINT[APPOINTMENT_appointment_id] PRIMARY KEY CLUSTERED ([appointment_id] ASC)
  );
  CREATE INDEX appointment_index ON APPOINTMENT(appointment_date);
```

SESSION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='SESSION')
 CREATE TABLE SESSION(
      [session_id] BIGINT IDENTITY(5100, 1),
      [session_date] DATE NOT NULL,
      [session_type] VARCHAR(20) CHECK ([session_type] IN ('PHYSICAL', 'VIRTUAL')) NOT NULL,
      [appointment_id] BIGINT FOREIGN KEY REFERENCES APPOINTMENT(appointment_id)

      CONSTRAINT[SESSION_session_id] PRIMARY KEY CLUSTERED ([session_id] ASC)
 );
     CREATE INDEX session_index ON SESSION(session_type)
```

SESSION_NOTE

```
 IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='SESSION_NOTE')
 CREATE TABLE SESSION_NOTE(
      [session_note_id] BIGINT IDENTITY(30150, 1),
      [session_note_header] VARCHAR(50) NOT NULL,
      [session_note_description] TEXT NOT NULL,
```

```
      [session_id] BIGINT FOREIGN KEY REFERENCES SESSION(session_id)

      CONSTRAINT[SESSION_session_note_pk] PRIMARY KEY CLUSTERED ([session_note_id] ASC)
 );
    CREATE INDEX session_note_index ON SESSION_NOTE(session_note_header);
```

COMMENT RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='COMMENT')
 CREATE TABLE COMMENT(
      [comment_id] BIGINT IDENTITY(0, 1),
      [comment_description] TEXT NOT NULL,
      [comment_date] DATE DEFAULT GETDATE(),
      [stu_id] BIGINT FOREIGN KEY REFERENCES STUDENT(stu_id)

      CONSTRAINT[COMMENT_comment_id] PRIMARY KEY CLUSTERED ([comment_id] ASC)
 );
  CREATE INDEX comment_index ON COMMENT(comment_date)
```

MOOD RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='MOOD')
 CREATE TABLE MOOD(
      [mood_id] BIGINT IDENTITY(1000, 1),
      [mood_rate] VARCHAR(50) CHECK ([mood_rate] IN ('VERY SAD', 'SAD', 'NORMAL', 'HAPPY', 'VERY
HAPPY')),
      [mood_rate_date] DATE DEFAULT GETDATE(),
      [stu_id] BIGINT FOREIGN KEY REFERENCES STUDENT(stu_id)

      CONSTRAINT[MOOD_mood_id] PRIMARY KEY CLUSTERED ([mood_id] ASC)
 );
 CREATE INDEX mood_index ON MOOD(mood_rate);
```

ENROLLMENT RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='ENROLLMENT')
 CREATE TABLE ENROLLMENT(
```

```
      [enrollment_id] BIGINT IDENTITY(90400, 1),
      [enrollment_date] DATE DEFAULT GETDATE(),
      [stu_id] BIGINT FOREIGN KEY REFERENCES STUDENT(stu_id),
      [course_id] BIGINT FOREIGN KEY REFERENCES COURSE(course_id)

      CONSTRAINT[ENROLLMENT_enrollment_id] PRIMARY KEY CLUSTERED ([enrollment_id] ASC)
 );
  CREATE INDEX enrollment_index ON ENROLLMENT(enrollment_date);
```

COURSE RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='COURSE')
 CREATE TABLE COURSE(
      [course_id] BIGINT IDENTITY(10800, 1),
      [course_name] VARCHAR(50) NOT NULL,
      [department_id] INTEGER FOREIGN KEY REFERENCES DEPARTMENT(department_id)

      CONSTRAINT[COURSE_course_id] PRIMARY KEY CLUSTERED ([course_id] ASC)
 );
   CREATE INDEX course_index ON COURSE(course_name);
```

DEPARTMENT RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='DEPARTMENT')
 CREATE TABLE DEPARTMENT(
      [department_id] INTEGER IDENTITY(100, 1),
      [department_name] VARCHAR(50) NOT NULL UNIQUE,

    CONSTRAINT[DEPARTMENT_department_id] PRIMARY KEY CLUSTERED ([department_id] ASC)
 );
    CREATE INDEX department_name ON DEPARTMENT(department_name);
```

REFERRAL_ADMISSION RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='REFERRAL_ADMISSION')
 CREATE TABLE REFERRAL_ADMISSION(
```

```
      [ref_admission_id] BIGINT IDENTITY(10200, 1),
      [ref_admission_date] DATE DEFAULT GETDATE(),
      [stu_id] BIGINT FOREIGN KEY REFERENCES STUDENT(stu_id),
      [ref_reason_id] BIGINT FOREIGN KEY REFERENCES REFERRAL_REASON(ref_reason_id)

      CONSTRAINT[REFERRAL_ADMISSION_ref_admission_id] PRIMARY KEY CLUSTERED ([ref_admission_id] ASC)
 );
    CREATE INDEX ref_admission_index ON REFERRAL_ADMISSION(ref_admission_date);
```

REFERRAL_REASON RELATION

```
IF NOT EXISTS (SELECT * FROM sys.tables t WHERE t.NAME='REFERRAL_REASON')
 CREATE TABLE REFERRAL_REASON(
      [ref_reason_id] BIGINT IDENTITY(6100, 1),
      [ref_reason_name] VARCHAR(50) NOT NULL,
      [ref_reason_description] TEXT NOT NULL,

      CONSTRAINT[REFERRAL_REASON_ref_reason_id] PRIMARY KEY ([ref_reason_id] ASC)
 );
    CREATE INDEX ref_reason_index ON REFERRAL_REASON(ref_reason_name);
```

7. System Security and Integrity Control.

- 7.1. Integrity Control
  - 7.1.1. Input Control.
  - 7.1.2. Completeness Control.
  - 7.1.3. Data Validation Control.
  - 7.1.4. Field Combination Control.
  - 7.1.5. Output Control.
- 7.2. Redundancy, Backup, & Recovery
  - 7.2.1. Data Redundancy
  - 7.2.2. Backup & Recovery
  - 7.2.3. Fraud Prevention
  - 7.2.4. Security Option

8. References