```
from sqlalchemy import create_engine, Column, Integer, String, Enum, ForeignKey, Text,
DateTime
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, relationship
import enum
from datetime import datetime

# Base class for the models
Base = declarative_base()

# Enum for difficulty levels
class Difficulty(enum.Enum):
    easy = "easy"
    intermediate = "intermediate"
    hard = "hard"

# Enum for quiz types
class QuizType(enum.Enum):
    multiple_choice = "multiple choice"
    matching = "matching"
    calc = "calc"

# Define the Student model
class Student(Base):
    __tablename__ = 'students'

    id = Column(Integer, primary_key=True)
    first_name = Column(String(50), nullable=False)
    last_name = Column(String(50), nullable=False)
    email = Column(String(100), unique=True, nullable=False)
    password = Column(String(255), nullable=False)  # Store password securely (e.g.,
hashed)

    def __repr__(self):
        return f"<Student(id={self.id}, first_name={self.first_name},
last_name={self.last_name})>"

# Define the Topic model
class Topic(Base):
    __tablename__ = 'topics'

    id = Column(Integer, primary_key=True)
    name = Column(String(100), nullable=False)  # Name of the topic
    difficulty = Column(Enum(Difficulty), nullable=False)  # Difficulty level of the topic
    description = Column(String(255), nullable=False)  # One-sentence description of the
subtopics

    def __repr__(self):
```

```
        return f"<Topic(id={self.id}, name={self.name}, difficulty={self.difficulty},
description={self.description[:30]}...)>"

# Define the Quiz model
class Quiz(Base):
    __tablename__ = 'quizzes'

    id = Column(Integer, primary_key=True)
    topic_id = Column(Integer, ForeignKey('topics.id'), nullable=False)
    questions = Column(Text, nullable=False)  # Store questions in a text field (can be in
JSON format)
    type = Column(Enum(QuizType), nullable=False)  # Type of the quiz (multiple choice,
matching, calc)
    answers = Column(Text, nullable=False)  # Store answers in text format (JSON array,
CSV, etc.)
    correct_answers = Column(Text, nullable=False)  # Correct answers (JSON array or
comma-separated)

    # Relationship to the Topic table
    topic = relationship("Topic", back_populates="quizzes")

    def __repr__(self):
        return f"<Quiz(id={self.id}, topic_id={self.topic_id}, type={self.type},
questions={self.questions[:30]}...)>"

# Define the Results model
class Result(Base):
    __tablename__ = 'results'

    id = Column(Integer, primary_key=True)
    user_id = Column(Integer, ForeignKey('students.id'), nullable=False)
    quiz_id = Column(Integer, ForeignKey('quizzes.id'), nullable=False)
    score = Column(Integer, nullable=False)  # Store score as an integer
    timestamp = Column(DateTime, default=datetime.utcnow, nullable=False)  # Store
timestamp of when the quiz was taken

    # Relationships
    user = relationship("Student")
    quiz = relationship("Quiz")

    def __repr__(self):
        return f"<Result(id={self.id}, user_id={self.user_id}, quiz_id={self.quiz_id},
score={self.score}, timestamp={self.timestamp})>"

# Define the back relationship on Topic to access quizzes
Topic.quizzes = relationship("Quiz", back_populates="topic")

# Create the database engine (SQLite in this case)
```

```
DATABASE_URL = "sqlite:///example.db"  # You can change this to your DB URL
(PostgreSQL, MySQL, etc.)
engine = create_engine(DATABASE_URL, echo=True)

# Create all tables in the database (students, topics, quizzes, results tables)
Base.metadata.create_all(engine)

# Create a session to interact with the database
Session = sessionmaker(bind=engine)
session = Session()

# Example: Adding a student to the database
new_student = Student(
    first_name='John',
    last_name='Doe',
    email='john.doe@example.com',
    password='securepassword123'  # In real applications, hash this password!
)

# Example: Adding chemistry topics to the database
chemistry_topics = [
    Topic(
        name="Atomic Structure",
        difficulty=Difficulty.easy,
        description="An introduction to the structure of atoms, including protons, neutrons, and
electrons."
    ),
    Topic(
        name="Periodic Table",
        difficulty=Difficulty.intermediate,
        description="A study of the organization of elements based on their atomic number and
properties."
    ),
    Topic(
        name="Chemical Bonding",
        difficulty=Difficulty.intermediate,
        description="Understanding how atoms form bonds to create molecules through ionic
and covalent bonds."
    ),
    Topic(
        name="Organic Chemistry",
        difficulty=Difficulty.hard,
        description="An advanced study of carbon-based compounds, their structure, reactions,
and properties."
    ),
    Topic(
        name="Thermodynamics",
        difficulty=Difficulty.hard,
```

```
        description="The study of energy, heat, and the laws governing their transformations in chemical processes."
    )
]

# Add chemistry topics and student to the session
session.add(new_student)
session.add_all(chemistry_topics)

# Example: Adding quizzes related to the topics
quiz_for_atomic_structure = Quiz(
    topic_id=1,  # Linking to "Atomic Structure" topic
    questions="What is the atomic number of Carbon?; What subatomic particles are found in the nucleus?",
    type=QuizType.multiple_choice,
    answers="A) 6, B) 12, C) 8; A) Protons and Neutrons, B) Electrons, C) Neutrons and Electrons",
    correct_answers="A) 6; A) Protons and Neutrons"
)

quiz_for_organic_chemistry = Quiz(
    topic_id=4,  # Linking to "Organic Chemistry" topic
    questions="What is the structure of an alkane?; What is the general formula for alkenes?",
    type=QuizType.matching,
    answers="A) CH₄, B) C₂H₄; A) Saturated, B) Unsaturated",
    correct_answers="A) CH₄; A) Saturated"
)

# Add quizzes to session
session.add(quiz_for_atomic_structure)
session.add(quiz_for_organic_chemistry)

# Example: Adding quiz results for a student
result_for_atomic_structure = Result(
    user_id=1,  # Linking to student John Doe
    quiz_id=1,  # Linking to the "Atomic Structure" quiz
    score=8,  # Example score
    timestamp=datetime.utcnow()  # The timestamp is automatically set to the current time
)

# Add results to session
session.add(result_for_atomic_structure)

# Commit changes to the database
session.commit()

# Close the session
session.close()
```