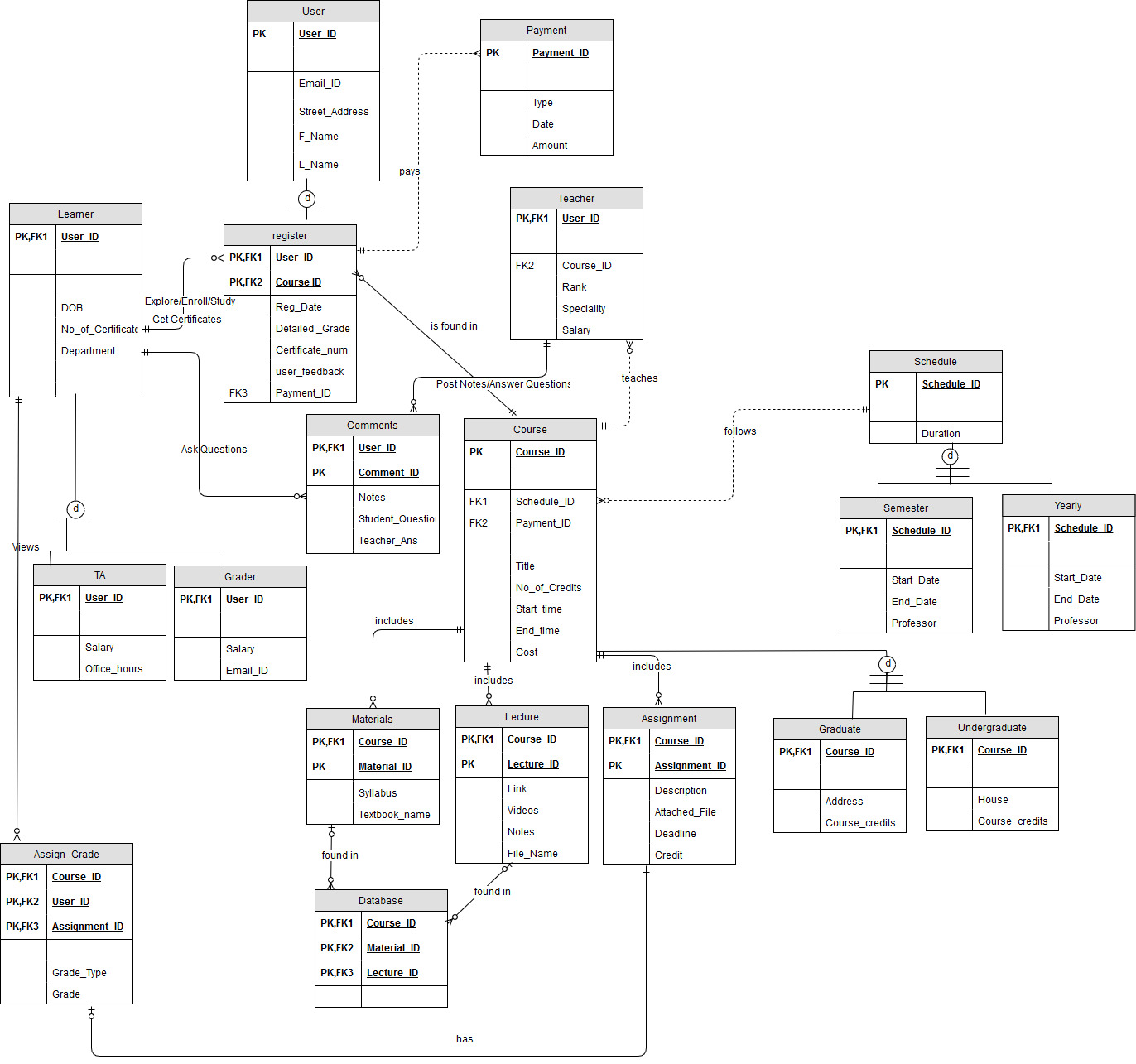
****

**CSCI 585 HW 1 Report**

The ERD has an entity user which has a primary key User\_ID. It has various attributes like Email\_ID, Street\_Address, F\_Name and L\_Name which take an e-learning system user’s email address, street address, first name and last name respectively.

We use a specialization relation now where User is specialized into Learner and a Teacher. This is an IS-A relation where a user can be either a learner or a teacher. This is a disjoint specialization as I have assumed that a person can’t be a learner and a teacher at the same time. It could even have been overlap specialization wherein a student is permitted to conduct lectures but I have kept it a disjoint specialization to avoid complexity. There is a partial participation as I have assumed there can be few users of the e-learning system that are not included in either learner or teacher categories. Accountants, clerks, administrative staff, etc are included in this category. This could have been a total participation but I have assumed and made it a partial participation to improve the scope of this model.

Both Learner and Teacher have their respective primary and foreign keys as User\_ID as it’s a specialization. A learner has attributes like DOB, No\_of\_Certificate, Department which takes a user’s date of birth, number of certificates on a course completion and his department (eg. Computer Science, Mechanical, Industrial). A learner can be further specialized into a TA and a Grader. This specialization could have been avoided but I have increased the scope of my model by showing TA and a grader as two different entities. I have assumed that a person can’t be a TA and a Grader at the same time and show it as a disjoint specialization. Also, I have shown a partial participation by assuming that there exists many learners who do not fit in either of the TA or Grader categories and are mere students.

There exists an entity Course which has a primary key Course\_ID and has attributes like Title, Course\_strength, Description, No\_of\_Credits, Start\_time, End\_time and cost which stores title for the course, maximum registration limit for a course, description for a course, number of credits assigned to the course, it’s start time, end time and cost for the course. Many other attributes could have been added to a course like it’s sections, class location, website link, etc but I have not included them to keep the ERD simple. There is a separate entity Materials which is connected to Course using a strong one to optional many relationship such that each course includes zero to many materials. Thus Materials has a composite key Course\_ID and Material\_ID. There is a separate entity Lectures which is connected to Course using a strong one to optional many relationship such that each course includes zero to many lectures. Thus Lectures has a composite key Course\_ID and Lecture\_ID. Also, there is a separate entity Assignments which is connected to Course using a strong one to optional many relationship such that each course includes zero to many assignments. Thus Lectures has a composite key Course\_ID and Assignment\_ID. Each of Materials, Lecture and Assignment have foreign key as Course\_ID due to the strong relationship. Materials have attributes like Syllabus, Textbook\_name which store the syllabus and the textbook name for the materials. Lecture has attributes like Link, Videos, Notes and File\_Name which store values like lecture link, video url, additional notes and file name for a lecture. Assignment has attributes like Description, Attached\_File, Deadline and Points which store values like description for a particular assignment, attachment (image or sample code), deadline submission date and number of points for a particular assignment. I have assumed a many to many relationship between Materials and Lecture by introducing Database as a bridge table. The relation between Materials and Lectures is such that a material consists of many lectures and a lecture can have materials included as well. Thus Database has a composite key of Material\_ID, Course\_ID and Lecture\_ID which also act as it’s foreign keys. The design could have been altered by showing a one to many relationship between Materials and Lectures but it would have limited the model scope. I have assumed a many to many relationship between Learner and Course wherein a learner can register for 0 to multiple courses and each course can have 0 to many learners in it. So I have used a register entity between learner and course to connect them. There is a strong relationship between learner and register wherein learner can explore, enroll, study or get certificates from multiple courses and a course can be be found in multiple registrations. register has a composite key made with User\_ID and Course\_ID which also act as foreign keys. register has attributes like Reg\_date, Detailed\_Grade, Certificate\_num, user\_feedback which stores a learner’s registration date, detailed grade report, certificate number on completion of a course as well as a user’s feedback or rating for a course. A course can be also specialized into Graduate and Undergraduate entities. Both graduate and undergraduate entities have Course\_ID as the primary and foreign key. Both Graduate and Undergraduate entities have Location and Course\_credits as the attributes which store the location where course lectures are conducted and the number of credits covered by a course. I have assumed a disjoint specialization here wherein a course can’t have both graduate and undergraduate level standing at the same time. There is a total participation here as I have assumed that a course should fit in either of these two categories. It could have been partial participation wherein a course has a phd level standing as is neither a graduate level nor undergraduate level course. But I have used total participation to avoid complex ERD. Also, a course could have been categorized into many other categories like mandatory or optional course, etc.

There is an entity called Assign\_Grade which is connected to the Assignment table using a strong has-A one to one relationship such that each assignment has 0 or 1 grade. Assign\_Grade is connected to the Learner entity using one to optional many strong relationship such that each learner views 0…N grades for those number of assignments. Thus Assign\_Grade has a composite key Course\_ID, User\_ID and Assignment\_ID. They all act as foreign keys as well. Assignment table has attributes like Grade\_Type and Grade which stores the type of grade ( example- Letter Grade, Pass/Fail, etc) and the actual grade value. The model could have been altered by showing a many to many relationship between the entities Assignment and Learner such that one learner can have multiple assignments and one assignment is assigned to multiple Learner. This would have increased the scope of model but would have introduced complications concerning addition of a bridge table as we already have a grade table introduced establishing a relation between learner and assignments.

I have created an entity named Schedule which records the schedule for a course, it keeps a track of number of times a course is conducted in a year (eg. every semester, once in a year, etc). Course is related to Schedule using a weak one to optional many relationship such that many courses follow one schedule. Schedule has a primary key Schedule\_ID and has Duration as an attribute which records the duration of a course (eg. 3 months, 2 months, etc). Thus Schedule\_ID acts as a foreign key in the Course table. Schedule can be further specialized into Semester and Yearly. I have used a disjoint specialization such that a course can’t be scheduled both semesterwise and yearly. There is a total participation in this case. Both Semester and Yearly entities have attributes Start\_Date, End\_Date and a Professor. This

fulfills the assignment requirement of a course reopening multiple times a year. This will in-turn change the teachers, assignments, students, etc. The model can be altered by showing a one to one relationship between course and schedule such that each course gets an exclusive schedule or it was even possible to add a schedule attribute to a course which stores semester/year keywords indicating that the course is conducted semesterwise or yearly. I have created a separate schedule entity to increase the model scope by making it more flexible.

There is an entity Teacher having primary key User\_ID. User\_ID also acts as a foreign key due to the specialization relation. Teacher table has attributes Rank, Speciality and Salary which accepts the ranking, specialization area and the salary for a particular teacher. We can add other attributes like designation, office hours, achievements but I have kept the ERD simple by not adding them. Teacher is related to course by one to optional many weak relationship such that one course is taught by multiple teachers. Thus Teacher has a foreign key as Course\_ID. This design could have been made different by showing many to many relation between Teacher and Course entities by introducing a bridge table but would have made a more complex model so I stuck to the requirement mentioned in the assignment. There is an entity Comments for handling the interaction between Learner and Teacher. Thus a Teacher can Post Notes or Answer Questions using the Comments table. There exists a one to optional many strong relationship between Teacher and Comments such that one teacher can add zero to many comments. Similarly, there is one to optional many strong relationship between Learner and Comments such that a Learner can ask zero to multiple questions. Thus Comments is made up of a composite key User\_ID and Comment\_ID. User\_ID also acts It has attributes like Notes, Student\_Questions and Teacher\_Ans which holds the notes posted by teachers, questions asks by students and answers posted by teachers respectively. The model could have been altered by not creating Comments as a separate entity but I have made Comments a separate entity to make the model clearer.

There is an entity Payment which has Payment\_ID as the primary key. Payment has attributes like Type, Date and Amount which stores the type of payment method used like Visa, Mastercard, PayPal, etc, date the payment was made and the amount of payment. It is linked to register by a weak one to many relationship such that one student can have multiple payment options after registration. Thus Payment\_ID acts as a foreign key in the register table.