Attendance Management System

ICS1411--- Database Systems Laboratory

A MINI PROJECT REPORT

Submitted By

Pandiarajan D (3122237001035)
Rushabh S (3122237001044)
Sankara Narayanan V (3122237001046)

5-Year Integrated M.Tech
Computer Science and Engineering

Sri Sivasubramaniya Nadar College of Engineering
(An Autonomous Institution, Affiliated to Anna University)
Kalavakkam – 603110

April 2025

PROBLEM STATEMENT:

Managing student attendance manually using paper-based registers or spreadsheets is inefficient and can lead to data loss, human errors, and manipulation. Teachers often struggle with maintaining accurate records, while administrators face difficulties in analysing student attendance patterns. Additionally, students may miss classes due to various reasons and require a transparent system for checking attendance records.

The proposed Attendance Management System will allow faculty members to record attendance digitally, students to view their attendance status, and administrators to generate reports and analyse attendance trends.

Entities Identified:

- Department
- Student
- Course
- Teacher
- Lectures
- Attendance
- On_Duty

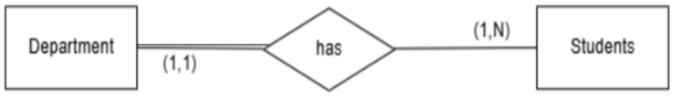
Relationships Identified:

- Department has Students
- Department has Teachers
- Department offers Courses
- Students enroll in courses
- Teachers teach Courses
- Courses have Lectures
- Lectures are taught by Teachers
- Students attend Lectures
- Attendance has ODs

<u>Relationships Identified</u> :-

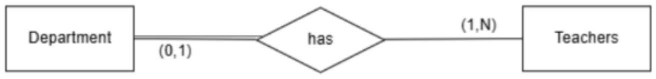
Department has Students

Department: Total (Every department must have at least one student)
Student: Partial (A student must belong to only one department)



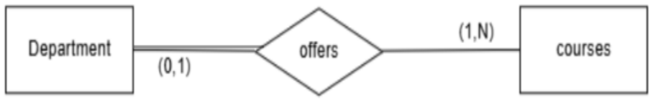
Department has Teachers

Department: Total (Every department must have at least one teacher)
Teacher: Partial (Not every teacher belongs to a department)



Department offers Courses

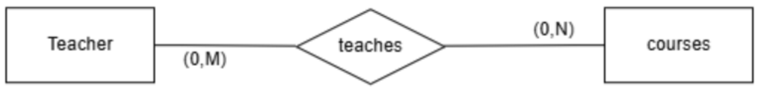
Department: Total (Every department must offer at least one course)
Course: Partial (Not every course must belong to a department)



Teacher teaches Courses

Teacher: Partial (Not all teachers may teach a course)

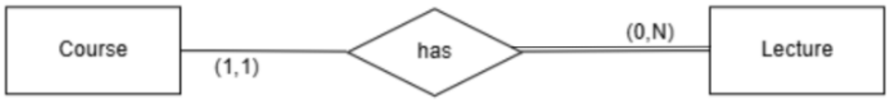
Course: Partial (Not all courses may have assigned teachers)



Course has Lecture

Lecture: Total (Each lecture is linked to a course)

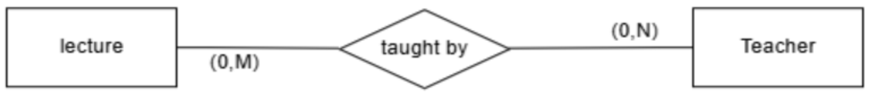
Course: Partial (Not all courses may have lectures)



Lecture taught by Teacher

Lecture: Partial (Not all courses have teachers assigned)

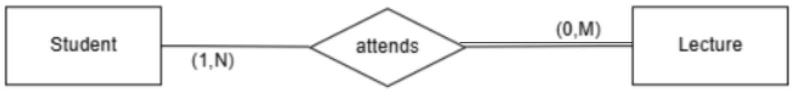
Teacher: Partial (A teacher may not be assigned to any course)



Student attends Lecture

Student: Partial (Not all students may attend lectures)

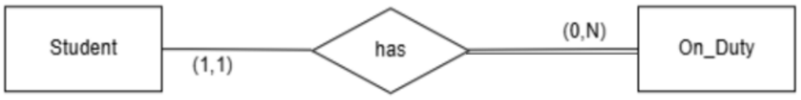
Lecture: Total (Every lecture must have at least one student)



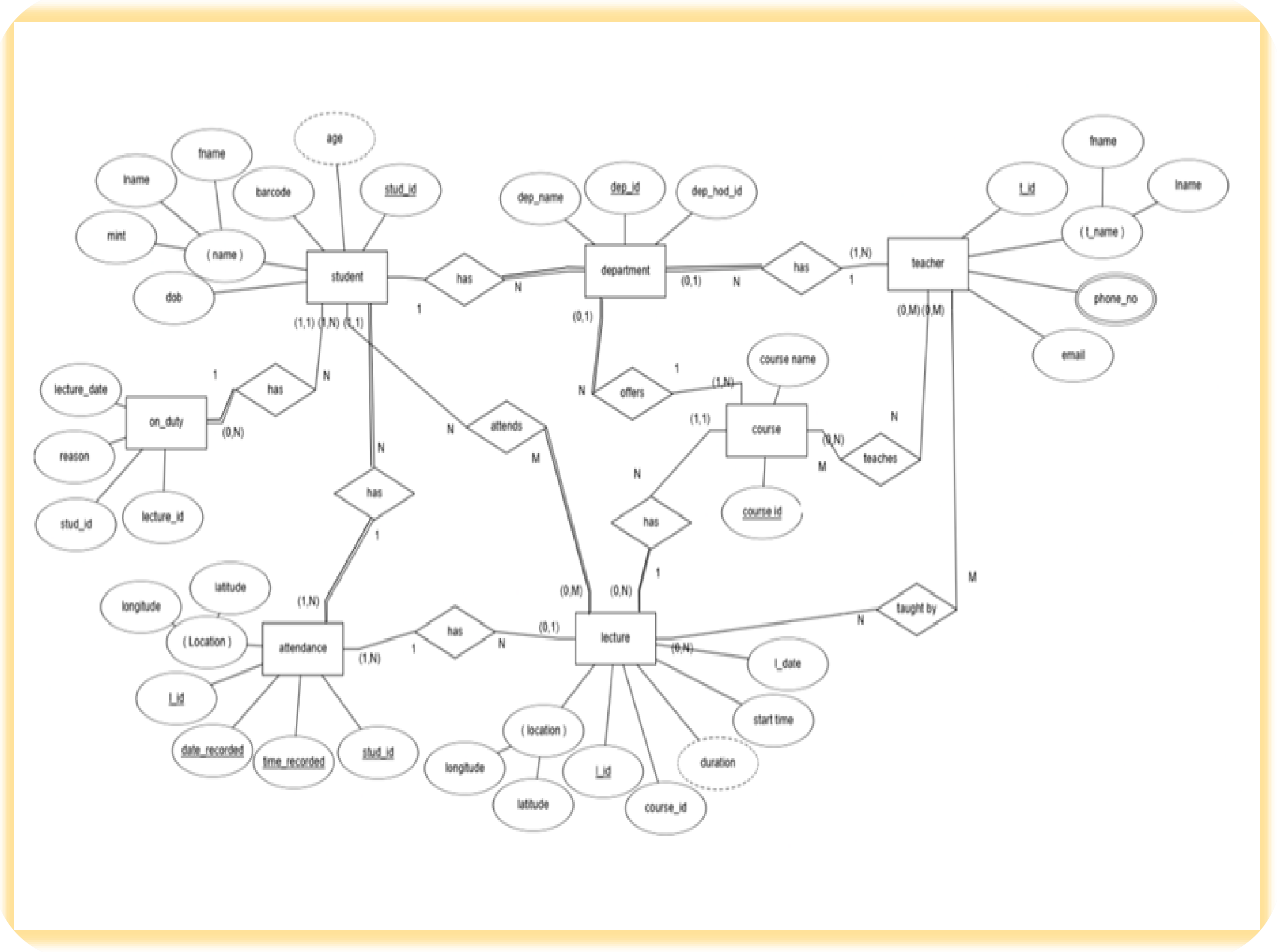
Student has On_Duty

Student: Partial (Not all students may apply for on-duty)

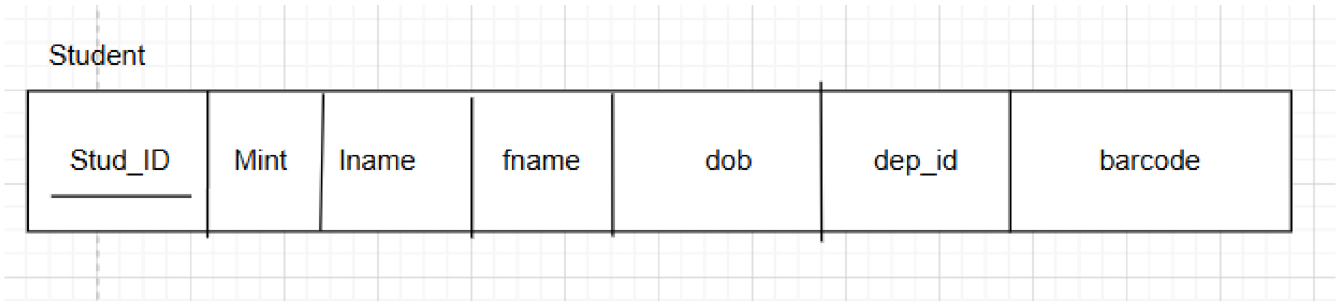
On_Duty: Total (Every on-duty record must belong to a student)



ER Diagram :-


age
fname
lname
barcode
stud_id
mint
(name)
dob
student
has
N
1
dep_name
dep_id
dep_hod_id
department
(0,1)
N
has
(1,N)
1
teacher
t_id
fname
lname
(t_name)
phone_no
email
(0,M) (0,M)
(1,1) (1,N) (1,1)
(0,1)
lecture_date
on_duty
has
1
N
(0,N)
reason
stud_id
lecture_id
offers
N
1
(1,N)
course name
course
(1,1)
(0,N)
teaches
N
M
course id
attends
N
M
has
N
1
has
N
1
(0,M)
(0,N)
M
taught by
N
longitude
latitude
(Location)
attendance
(1,N)
l_id
date_recorded
time_recorded
stud_id
has
1
N
(1,N)
(0,1)
lecture
(0,N)
l_date
start time
duration
(location)
longitude
latitude
l_id
course_id

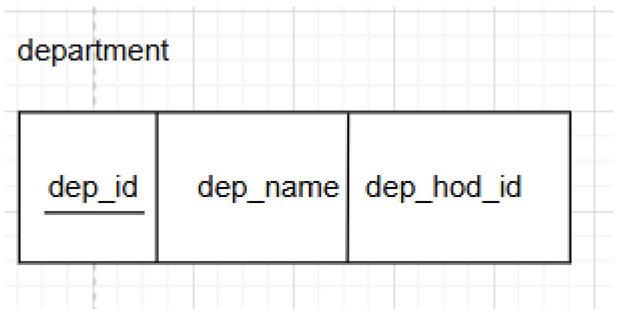
<u>ER to Relation Mapping</u> :-



1. **Student → Department (N:1)**

   - A student belongs to one department, but a department has many students.
   - Foreign Key: dep_id in Student table.

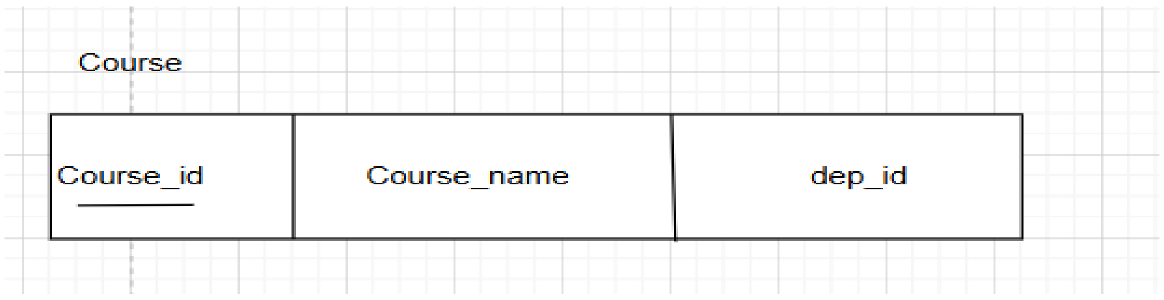
Rule: The many-side (Student) gets the foreign key of the one-side (Department).



2. **Course → Department (N:1)**

   - A course belongs to one department, but a department can have many courses.
   - Foreign Key: dep_id in Course.

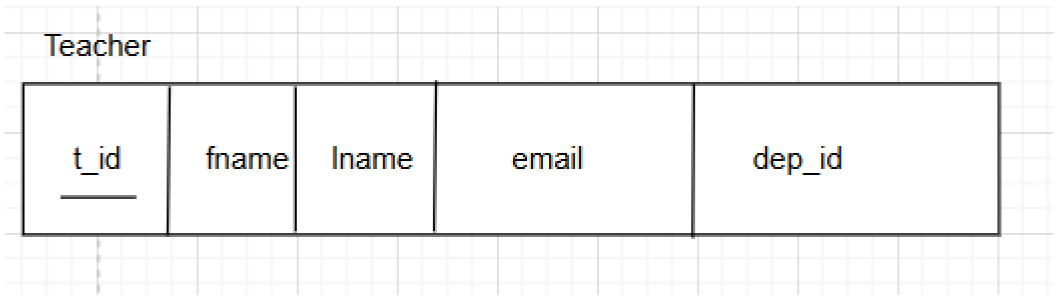
Rule: The many-side (Course) gets the foreign key of the one-side (Department).



3. Department → Teacher (1:N)

- A department can have multiple teachers, but each teacher belongs to only one department.
- Foreign Key: dep_id in Teacher.

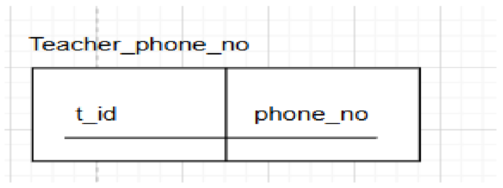
Rule: The teacher table gets the foreign key of the department.



4. Teacher → Teacher_Phone_No (1:N)

- phone_no is a multi valued attribute.
- A teacher can have multiple phone numbers, but each phone number belongs to one teacher.
- Foreign Key: t_id in Teacher_Phone_No.

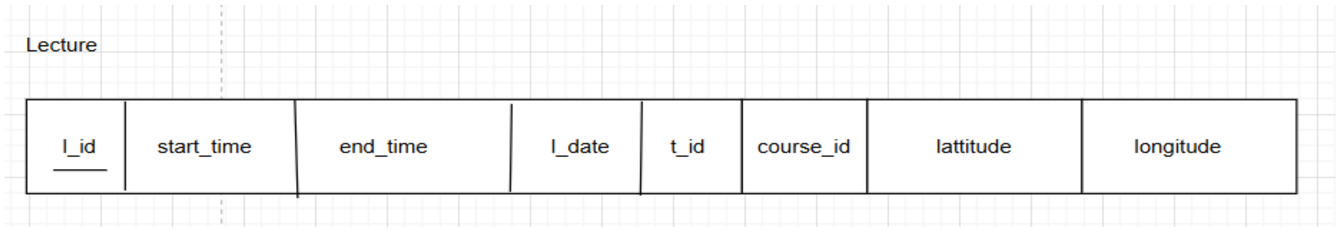
Rule: The Teacher_Phone_No table gets the foreign key of the Teacher.



5. Lecture → Course (N:1)

- A lecture is associated with one course, but a course can have multiple lectures.
- Foreign Key: course_id in Lecture.

Rule: The Lecture table gets the foreign key of the Course table.



6. Lecture → Teacher (N:1)

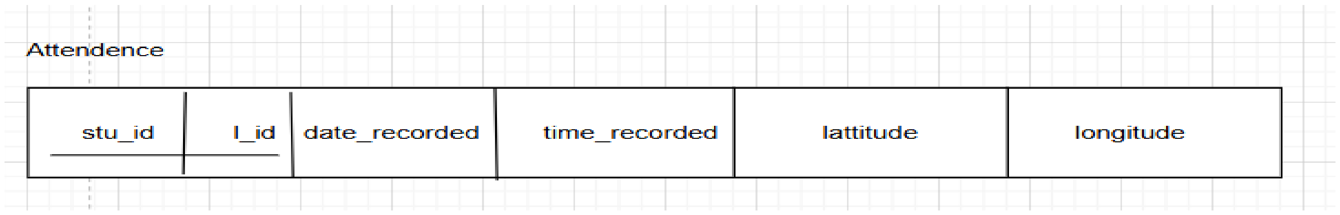
- A lecture is conducted by one teacher, but a teacher can conduct multiple lectures.
- Foreign Key: t_id in Lecture.

Rule: The Lecture table gets the foreign key of the Teacher table.

7. Student → Attendance (1:N)

- A student can have multiple attendance records, but each attendance record belongs to one student.
- Foreign Key: stu_id in Attendance.

Rule: The child entity (Attendance) takes the foreign key of the parent entity (Student).


Attendence

stu_id	l_id	date_recorded	time_recorded	lattitude	longitude

8. Attendance → Lecture (1:1)

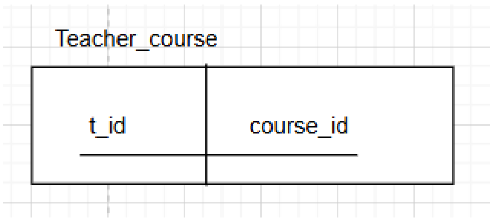
- An attendance record is for one lecture and vice versa.
- Foreign Key: l_id in Attendance.

Rule: The Attendance table gets the foreign key of the Lecture table.

9. Course → Teacher (M:N)

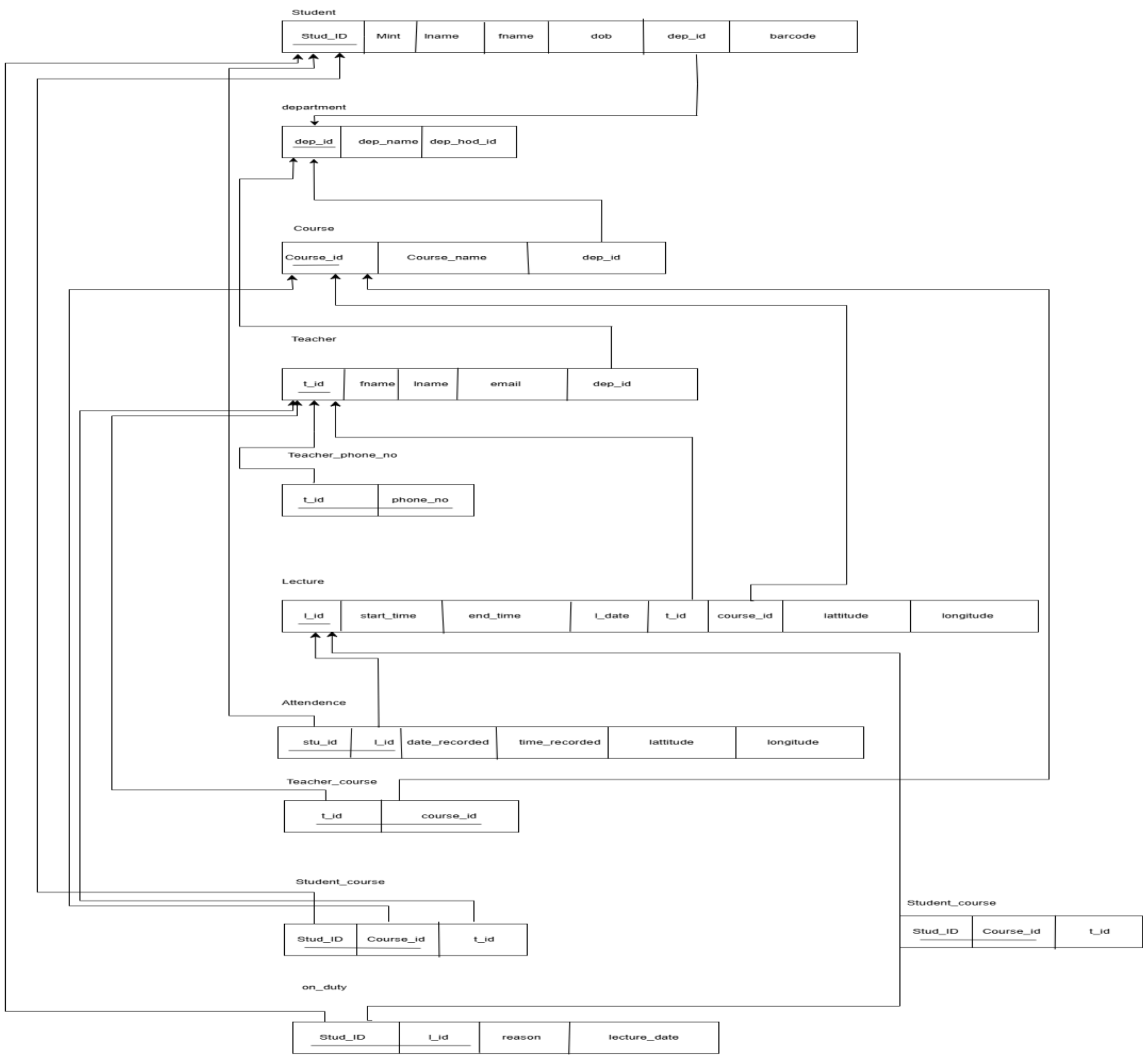
- A course is taught by many teachers, a teacher can teach many courses.
- A new entity teacher_course is created.

Rule: The Teacher_course takes the t_id and course_id and makes it the composite key.


Teacher_course

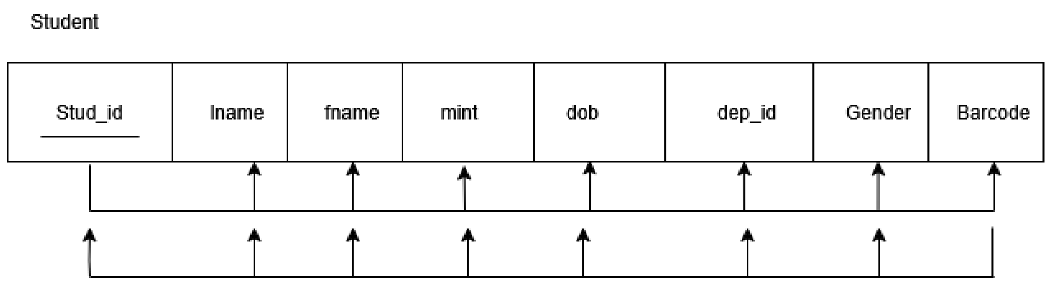
t_id	course_id

<u>Schema Diagram</u> :-



Functional Dependencies :-

Student



Stud_id → lname, fname, mint,dob, dep_id, Gender,Barcode

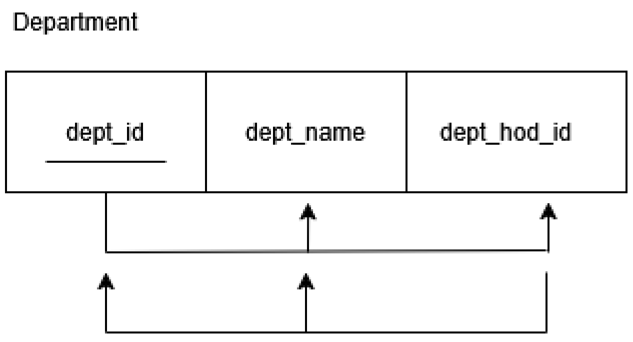
Barcode → Stud_id, dep_id, mint,Gender, lname, fname, dob

Stud_id → fname
Stud_id → lname
Stud_id → mint
Stud_id → dob
Stud_id → dep_id
Stud_id → Gender
Stud_id → Barcode

So the irreducible Functional Dependencies for Student table is

Stud_id → Barcode
Barcode→ Stud_id

Department



dept_id → dept_name, dept_hod_id
dept_hod_id → dept_id, dept_name

Step 1 :-

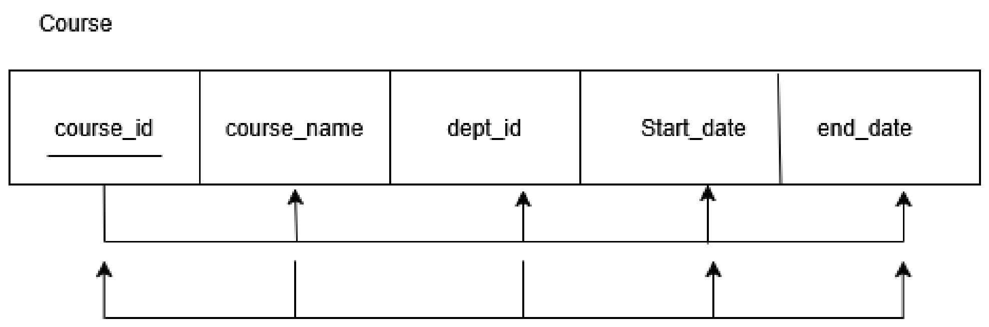
dept_id → dept_name
dept_id → dept_hod_id

dept_hod_id → dept_id
dept_hod_id → dept_name

So the irreducible Functional Dependencies for Department table is

Dept_id → dept_hod_id
Dept_hod_id → dep_id

Course



course_id → course_name, dept_id, Start_date, end_date
course_name, dept_id → course_id, Start_date, end_date

Step 1 :-

course_id → course_name
course_id → dept_id
course_id → Start_date
course_id → end_date

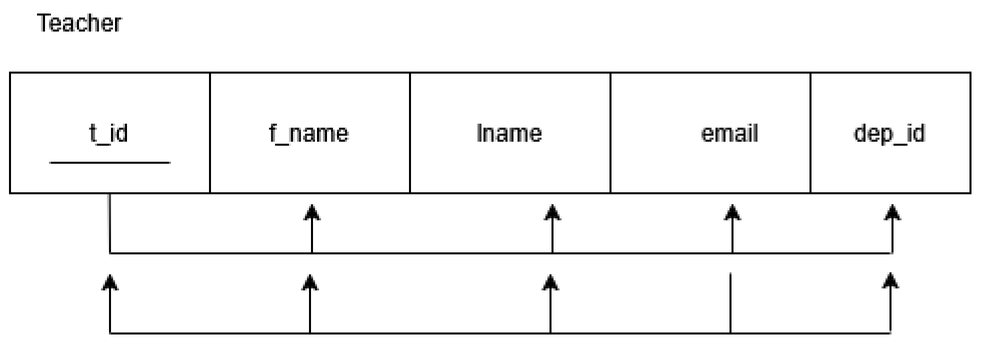
course_name, dept_id → course_id
course_name, dept_id → Start_date
course_name, dept_id → end_date

So the irreducible Functional Dependencies for Course table is

course_id → course_name
course_id → dept_id

course_name, dept_id → course_id
course_name, dept_id → Start_date
course_name, dept_id → end_date

Teacher



t_id → fname, lname, email, dep_id
email → t_id, fname, lname, dep_id

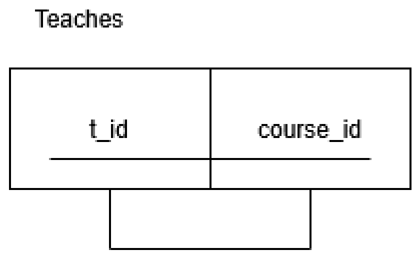
Step 1 :-

t_id → fname	email → t_id
t_id → lname	email → fname
t_id → email	email → lname
t_id → dep_id	email → dep_id

So the irreducible Functional Dependencies for Teacher table is

t_id → email
email → t_id

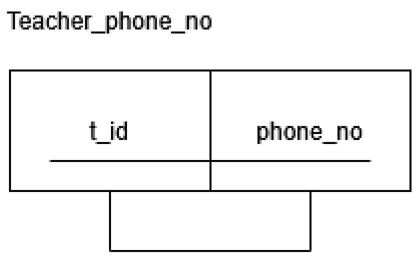
Teacher_course



So the irreducible Functional Dependencies for Teaches table is

t_id, course_id → (No additional attributes)

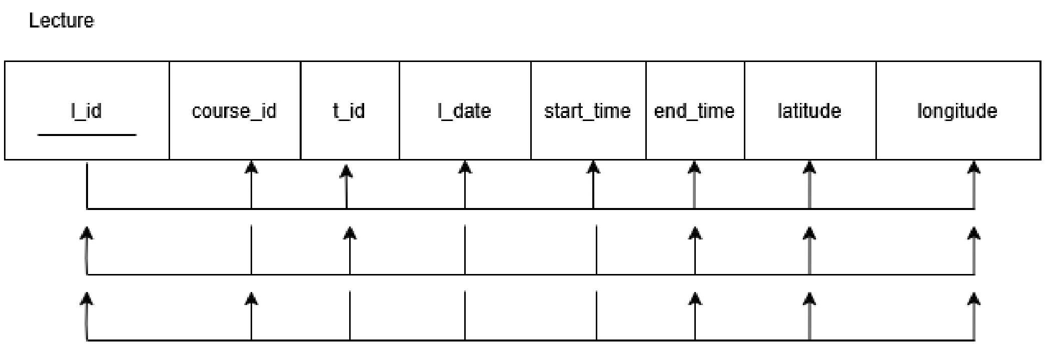
Teacher_phone



So the irreducible Functional Dependencies for Teacher_phone_no table is

t_id, phone_no → (No additional attributes)

Lecture



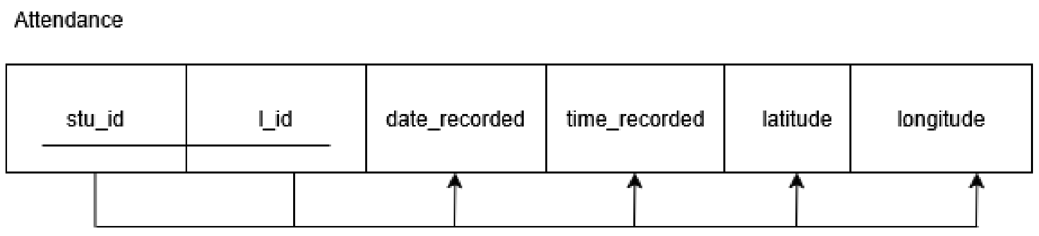
L_id → course_id, t_id, start_time, end_time, L_date,latitude,longitude
course_id, start_time, L_date → L_id, t_id, end_time
t_id, start_time, L_date → L_id, course_id, end_time

L_id → course_id	course_id, start_time, L_date → L_id	t_id, start_time, L_date → L_id
L_id → t_id	course_id, start_time, L_date → t_id	t_id, start_time, L_date → course_id
L_id → start_time	course_id, start_time, L_date → end_time	L_date → end_time
L_id → end_time	course_id, start_time, L_date → latitude	t_id, start_time, L_date → latitude
L_id → L_date	course_id, start_time, L_date → longitude	t_id, start_time, L_date → longitude
L_id → latitude		
L_id → longitude		

So the irreducible Functional Dependencies for Lecture table is

L_id → course_id
L_id → start_time
L_id → L_date
course_id, start_time, L_date → L_id
t_id, start_time, L_date → L_id

Attendance



stu_id, L_id → date_recorded, time_recorded,latitude,longitude

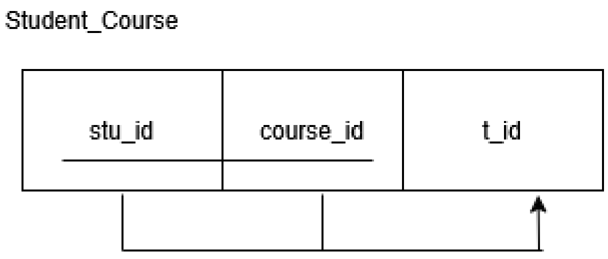
Step 1 :-

stu_id, L_id → date_recorded
stu_id, L_id → time_recorded
stu_id, L_id → latitude
stu_id, L_id → longitude

So the irreducible Functional Dependencies for Attendance table is

stu_id, L_id → date_recorded
stu_id, L_id → time_recorded
stu_id, L_id → latitude
stu_id, L_id → longitude

Student_course



stu_id, course_id→ t_id

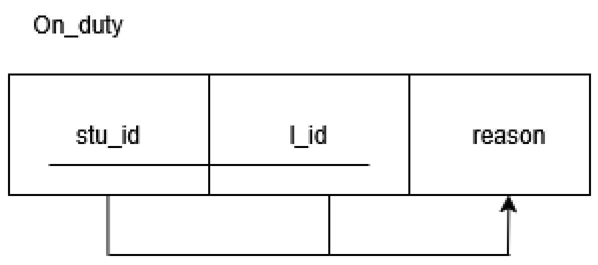
Step 1 :-

stu_id, course_id → tid

So the irreducible Functional Dependencies for Student_Course table is

stu_id, course_id → tid

On_Duty



stu_id, l_id→ reason

Step 1 :-

stu_id, l_id → reason

So the irreducible Functional Dependencies for on_duty table is

stu_id, l_id → reason

Database Normalization Analysis (Up to BCNF)

1. Student Relation

Original Functional Dependencies (FDs):
Stud_id → lname, fname, mint, dob, dep_id, Gender, Barcode
Barcode → Stud_id, dep_id, mint, Gender, lname, fname, dob

Analysis:

- Both Stud_id and Barcode are candidate keys (they can uniquely identify a tuple)
- All attributes are functionally dependent on these keys
- No partial dependencies or transitive dependencies
- The relation is already in BCNF since for every FD X → Y, X is a superkey

BCNF Decomposition:
Student(Stud_id, lname, fname, mint, dob, dep_id, Gender, Barcode)
Candidate keys: {Stud_id}, {Barcode}

2. Department Relation

Original FDs:
dept_id → dept_name, dept_hod_id
dept_hod_id → dept_id, dept_name

Analysis:

- Both dept_id and dept_hod_id are candidate keys
- No partial or transitive dependencies
- Already in BCNF since for every FD X → Y, X is a superkey

BCNF Decomposition:
Department(dept_id, dept_name, dept_hod_id)
Candidate keys: {dept_id}, {dept_hod_id}

3. Course Relation

Original FDs:
course_id → course_name, dept_id, Start_date, end_date
(course_name, dept_id) → course_id, Start_date, end_date

Analysis:

- Both course_id and (course_name, dept_id) are candidate keys
- No partial or transitive dependencies
- Already in BCNF since for every FD X → Y, X is a superkey

BCNF Decomposition:
Course(course_id, course_name, dept_id, Start_date, end_date)
Candidate keys: {course_id}, {course_name, dept_id}

4. Teacher Relation

Original FDs:
t_id → fname, lname, email, dep_id
email → t_id, fname, lname, dep_id

Analysis:

- Both t_id and email are candidate keys
- No partial or transitive dependencies
- Already in BCNF since for every FD X → Y, X is a superkey

BCNF Decomposition:
Teacher(t_id, fname, lname, email, dep_id)
Candidate keys: {t_id}, {email}

5. Teaches Relation

Original FDs:
(t_id, course_id) → (No additional attributes)

Analysis:

- The only FD is the entire composite key determining no additional attributes
- Already in BCNF since the only FD has a superkey on the left side

BCNF Decomposition:
Teaches(t_id, course_id)
Candidate key: {t_id, course_id}

6. Teacher_phone_no Relation

Original FDs:
(t_id, phone_no) → (No additional attributes)

Analysis:

- The only FD is the entire composite key determining no additional attributes
- Already in BCNF since the only FD has a superkey on the left side

BCNF Decomposition:
Teacher_phone_no(t_id, phone_no)
Candidate key: {t_id, phone_no}

7. Lecture Relation

Original FDs:
L_id → course_id, t_id, start_time, end_time, L_date, latitude, longitude
(course_id, start_time, L_date) → L_id, t_id, end_time
(t_id, start_time, L_date) → L_id, course_id, end_time

Analysis:

- L_id is a candidate key
- (course_id, start_time, L_date) is a candidate key
- (t_id, start_time, L_date) is a candidate key
- All attributes are functionally dependent on these keys
- No partial or transitive dependencies
- Already in BCNF since for every FD X → Y, X is a superkey

BCNF Decomposition:
Lecture(L_id, course_id, t_id, start_time, end_time, L_date, latitude, longitude)
Candidate keys: {L_id}, {course_id, start_time, L_date}, {t_id, start_time, L_date}

8. Attendance Relation

Original FDs:
(stu_id, L_id) → date_recorded, time_recorded, latitude, longitude

Analysis:

- The only FD is the entire composite key determining all other attributes
- Already in BCNF since the only FD has a superkey on the left side

BCNF Decomposition:
Attendance(stu_id, L_id, date_recorded, time_recorded, latitude, longitude)
Candidate key: {stu_id, L_id}

9. Student_Course Relation

Original FDs:
(stu_id, course_id) → t_id

Analysis:

- The only FD is the entire composite key determining t_id
- Already in BCNF since the only FD has a superkey on the left side

BCNF Decomposition:
Student_Course(stu_id, course_id, t_id)
Candidate key: {stu_id, course_id}

10. On_duty Relation

Original FDs:
(stu_id, l_id) → reason

Analysis:

- The only FD is the entire composite key determining reason
- Already in BCNF since the only FD has a superkey on the left side

BCNF Decomposition:
On_duty(stu_id, l_id, reason)
Candidate key: {stu_id, l_id}

Final BCNF Schema :-

1. Student(Stud_id, lname, fname, mint, dob, dep_id, Gender, Barcode)
Candidate keys: {Stud_id}, {Barcode}

2. Department(dept_id, dept_name, dept_hod_id)
Candidate keys: {dept_id}, {dept_hod_id}

3. Course(course_id, course_name, dept_id, Start_date, end_date)
Candidate keys: {course_id}, {course_name, dept_id}

4. Teacher(t_id, fname, lname, email, dep_id)
Candidate keys: {t_id}, {email}

5. Teaches(t_id, course_id)
Candidate key: {t_id, course_id}

6. Teacher_phone_no(t_id, phone_no)
Candidate key: {t_id, phone_no}

7. Lecture(L_id, course_id, t_id, start_time, end_time, L_date, latitude, longitude)
Candidate keys: {L_id}, {course_id, start_time, L_date}, {t_id, start_time, L_date}

8. Attendance(stu_id, L_id, date_recorded, time_recorded, latitude, longitude)
Candidate key: {stu_id, L_id}

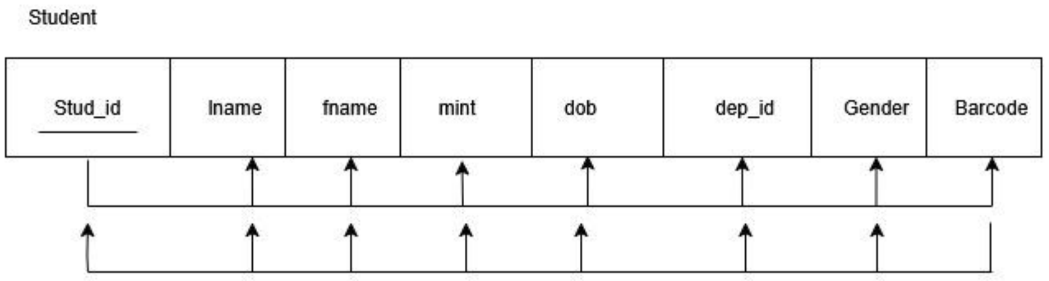
9. Student_Course(stu_id, course_id, t_id)
Candidate key: {stu_id, course_id}

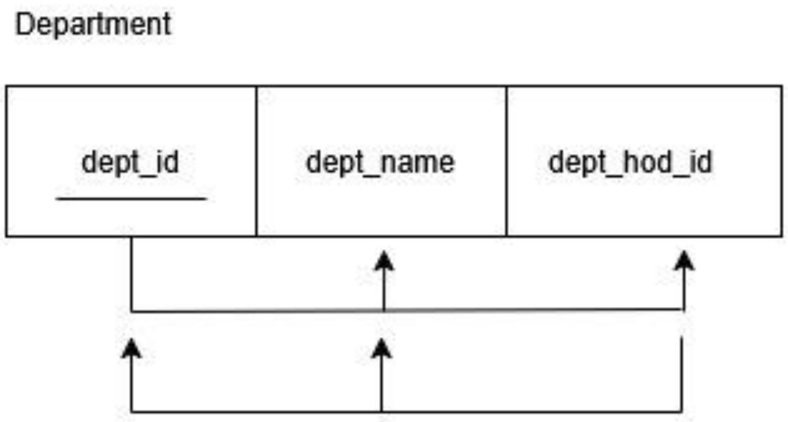
10. On_duty(stu_id, l_id, reason)
Candidate key: {stu_id, l_id}

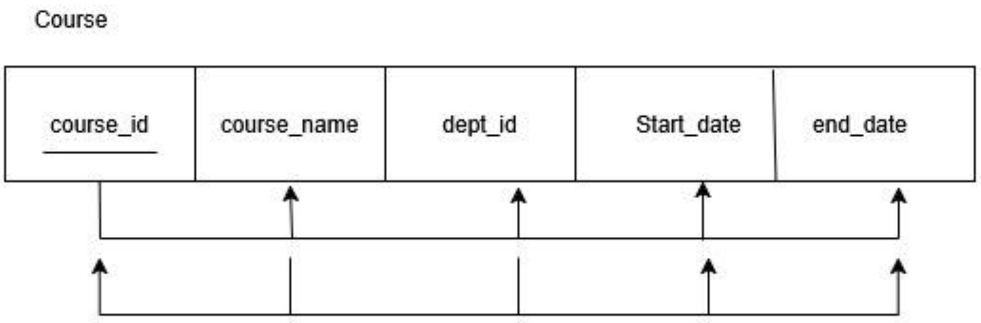
All relations are now in BoyceCodd Normal Form (BCNF) as for every functional dependency

X → Y, X is a superkey in each relation.

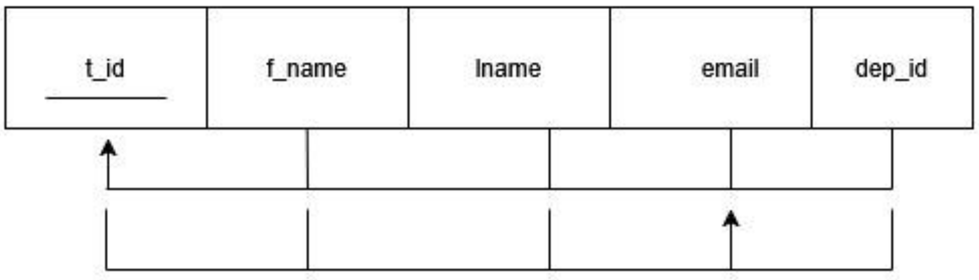
Decomposed Tables :-



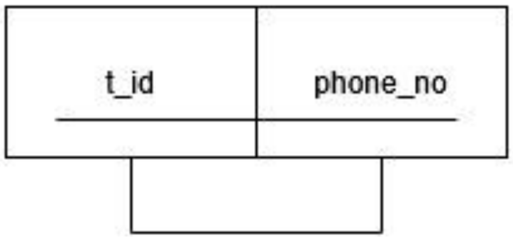




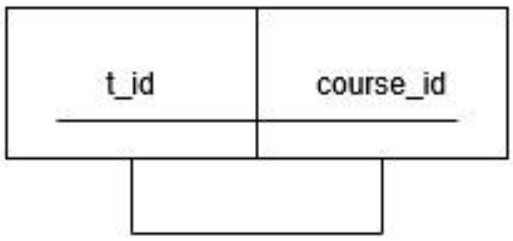
Teacher



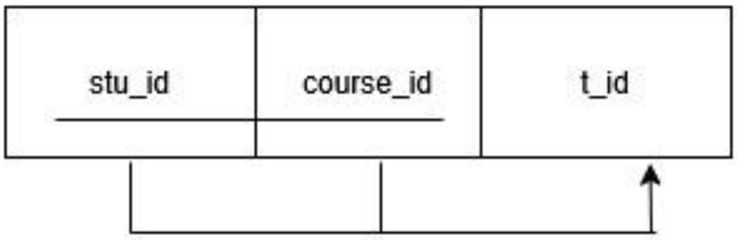
Teacher_phone_no



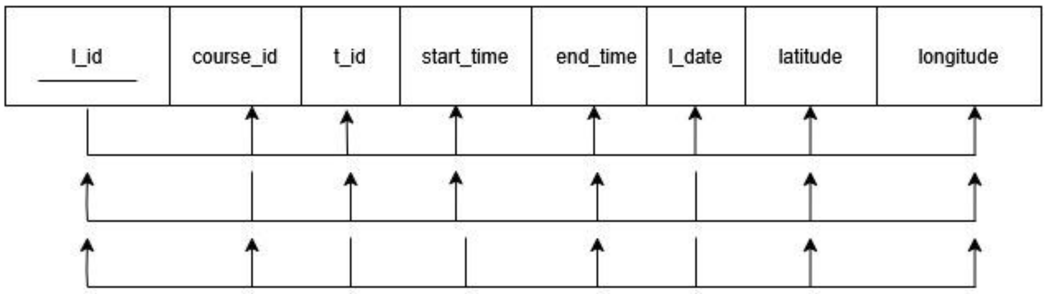
Teaches



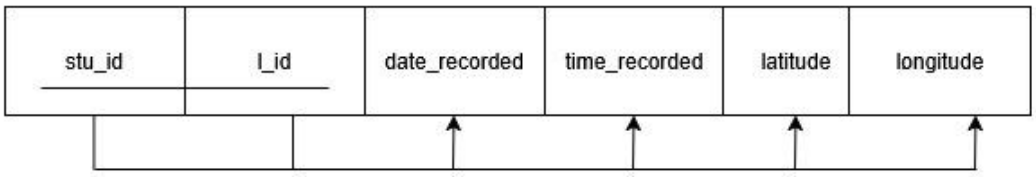
Student_Course



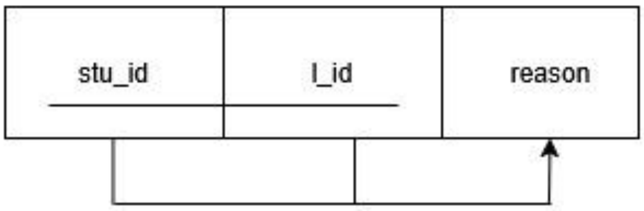
Lecture



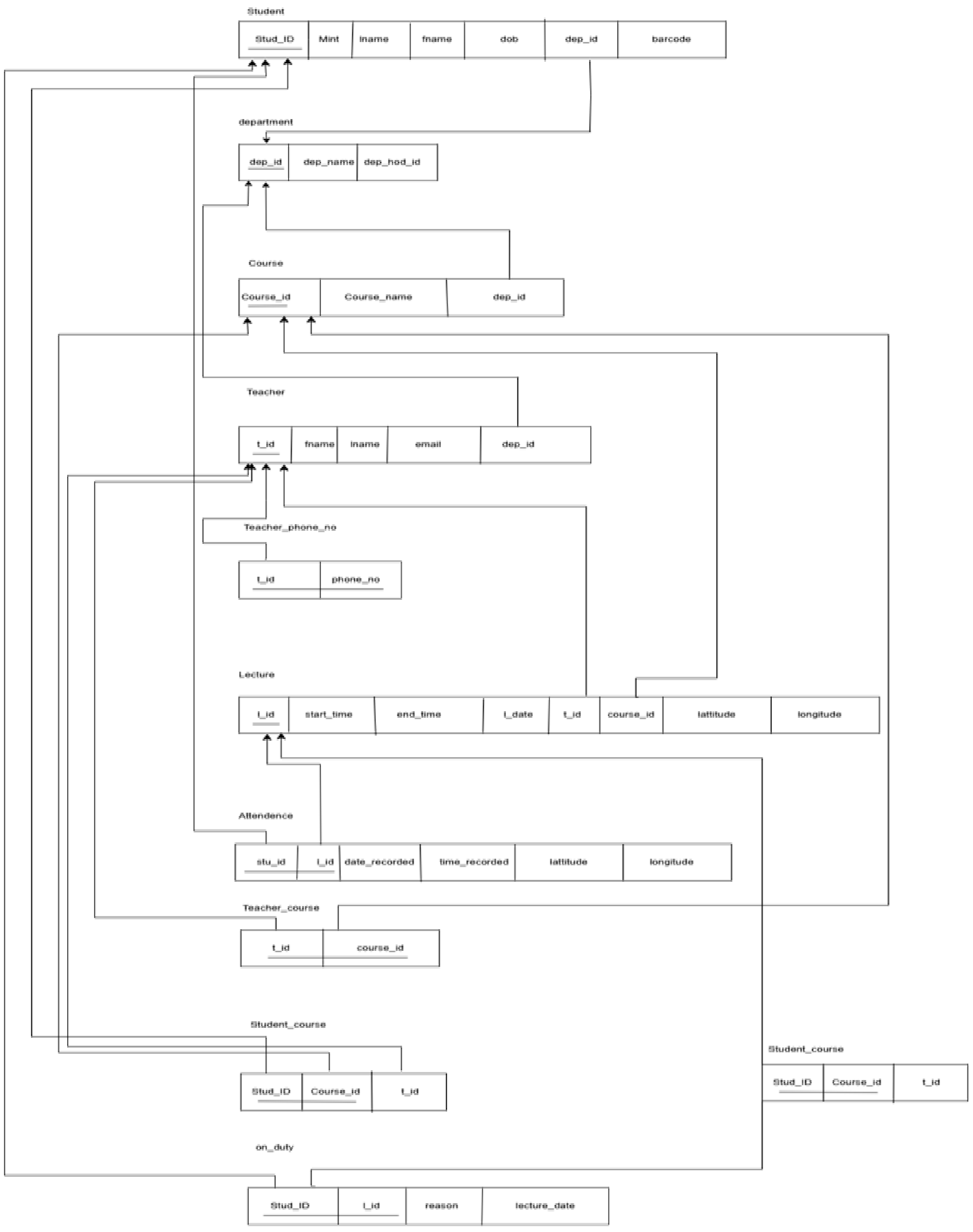
Attendance



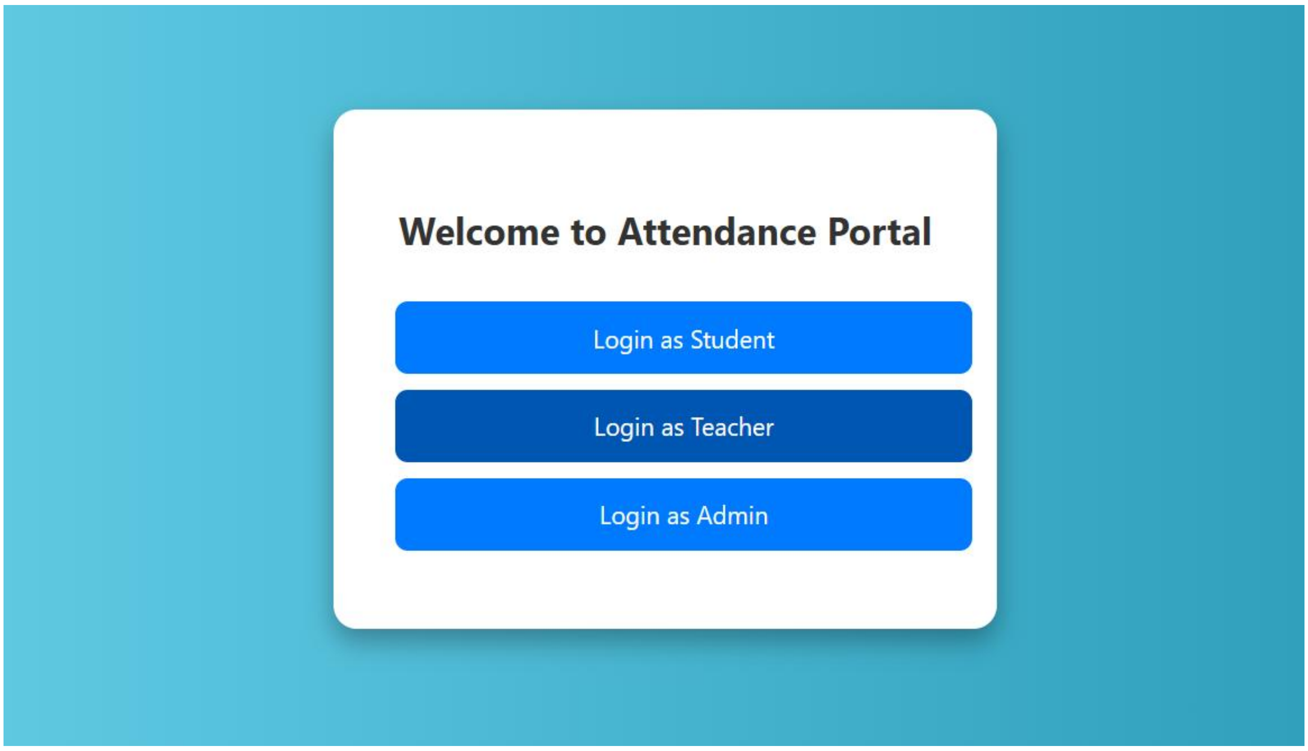
On_duty

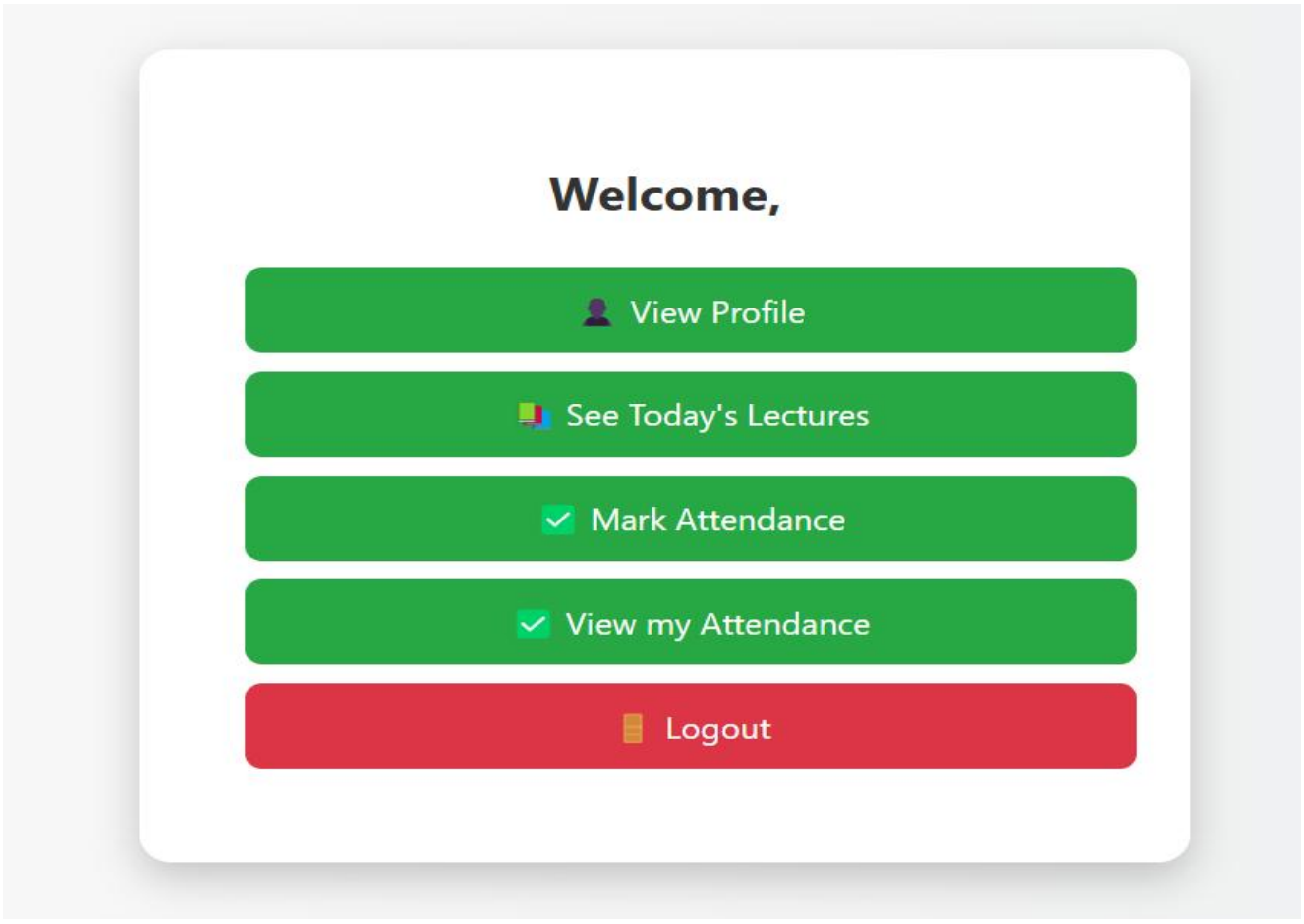


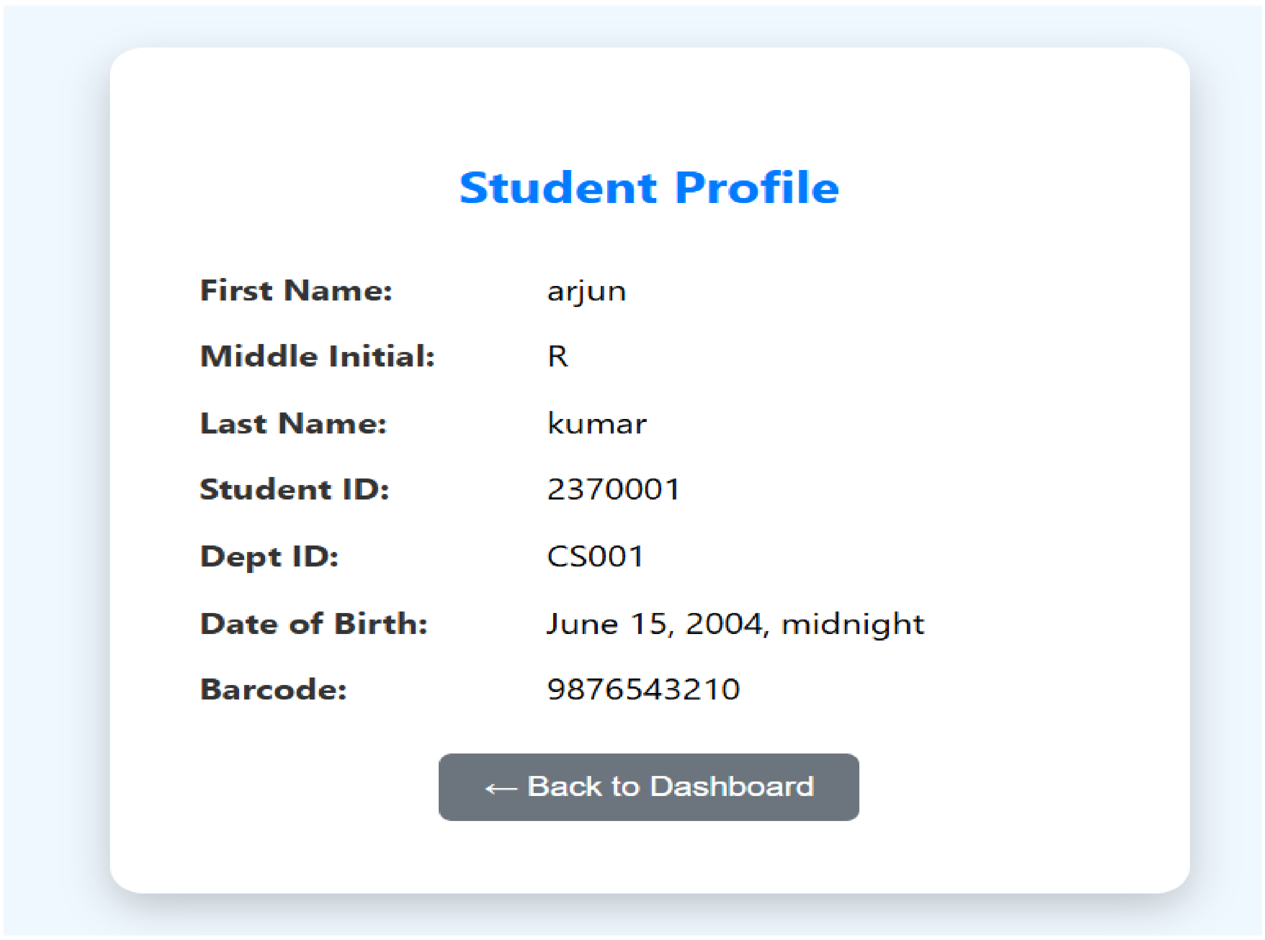
Final Schema :-



<u>Sample Execution</u> :-

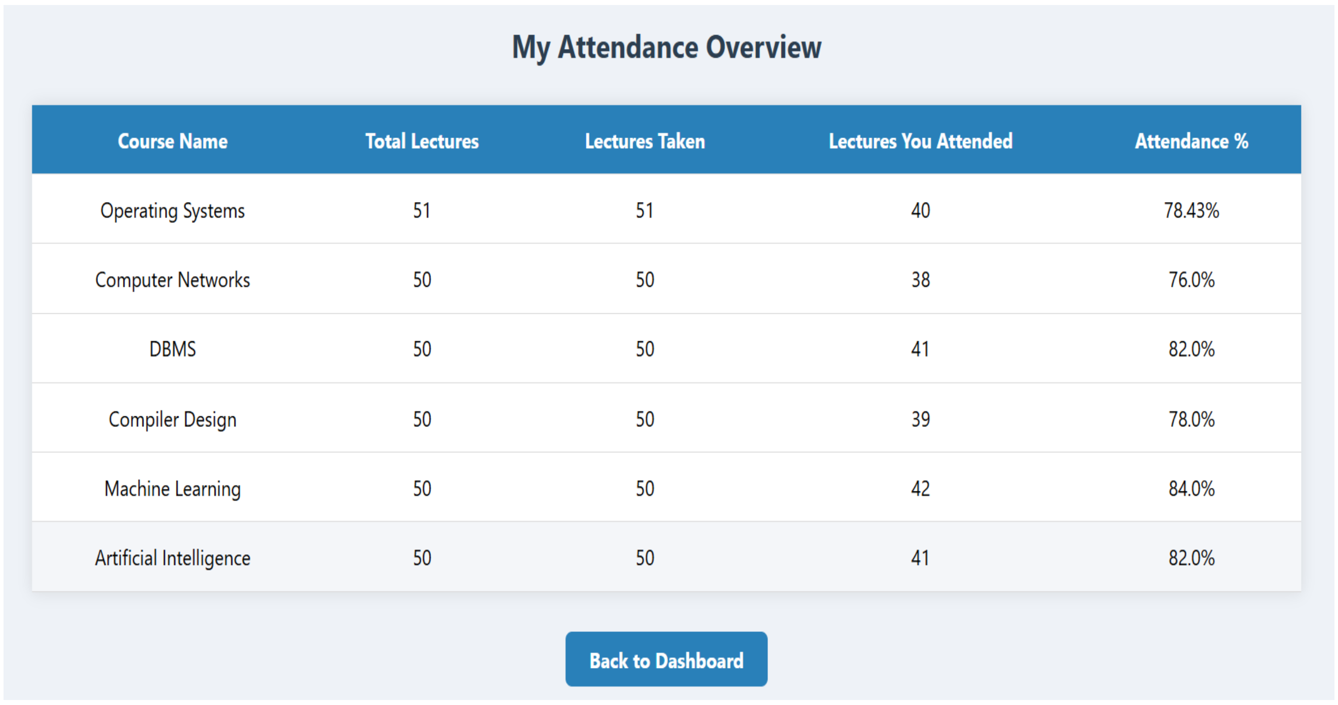






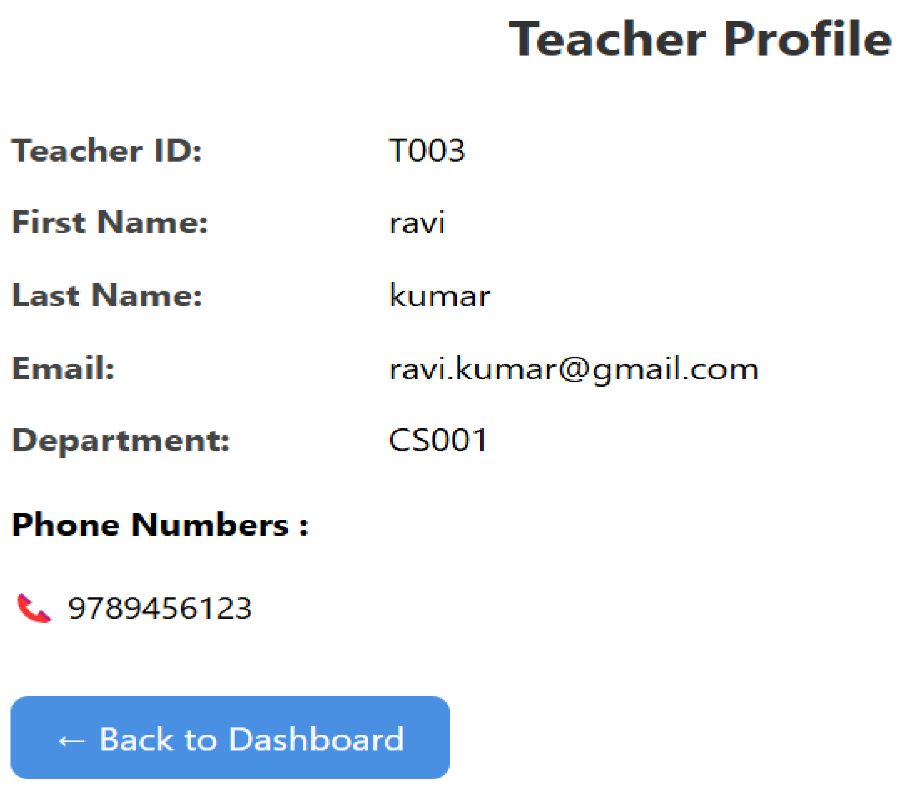
Student Profile

First Name:	arjun
Middle Initial:	R
Last Name:	kumar
Student ID:	2370001
Dept ID:	CS001
Date of Birth:	June 15, 2004, midnight
Barcode:	9876543210



My Attendance Overview

Course Name	Total Lectures	Lectures Taken	Lectures You Attended	Attendance %
Operating Systems	51	51	40	78.43%
Computer Networks	50	50	38	76.0%
DBMS	50	50	41	82.0%
Compiler Design	50	50	39	78.0%
Machine Learning	50	50	42	84.0%
Artificial Intelligence	50	50	41	82.0%

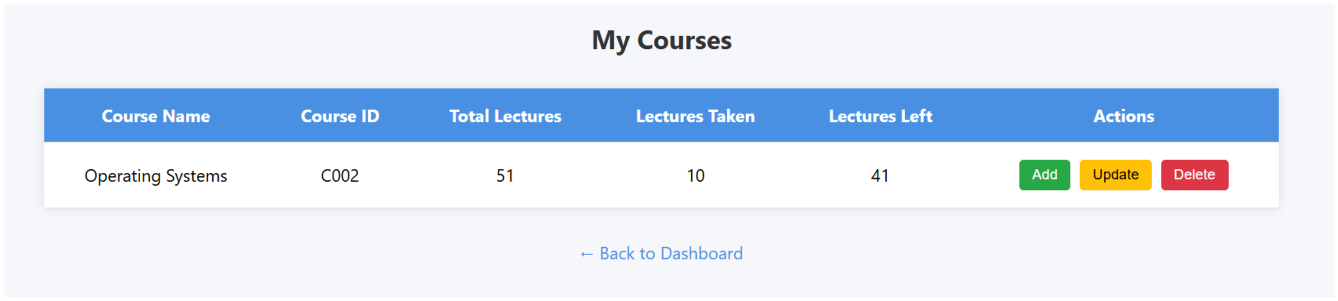


Teacher Profile

Teacher ID:	T003
First Name:	ravi
Last Name:	kumar
Email:	ravi.kumar@gmail.com
Department:	CS001

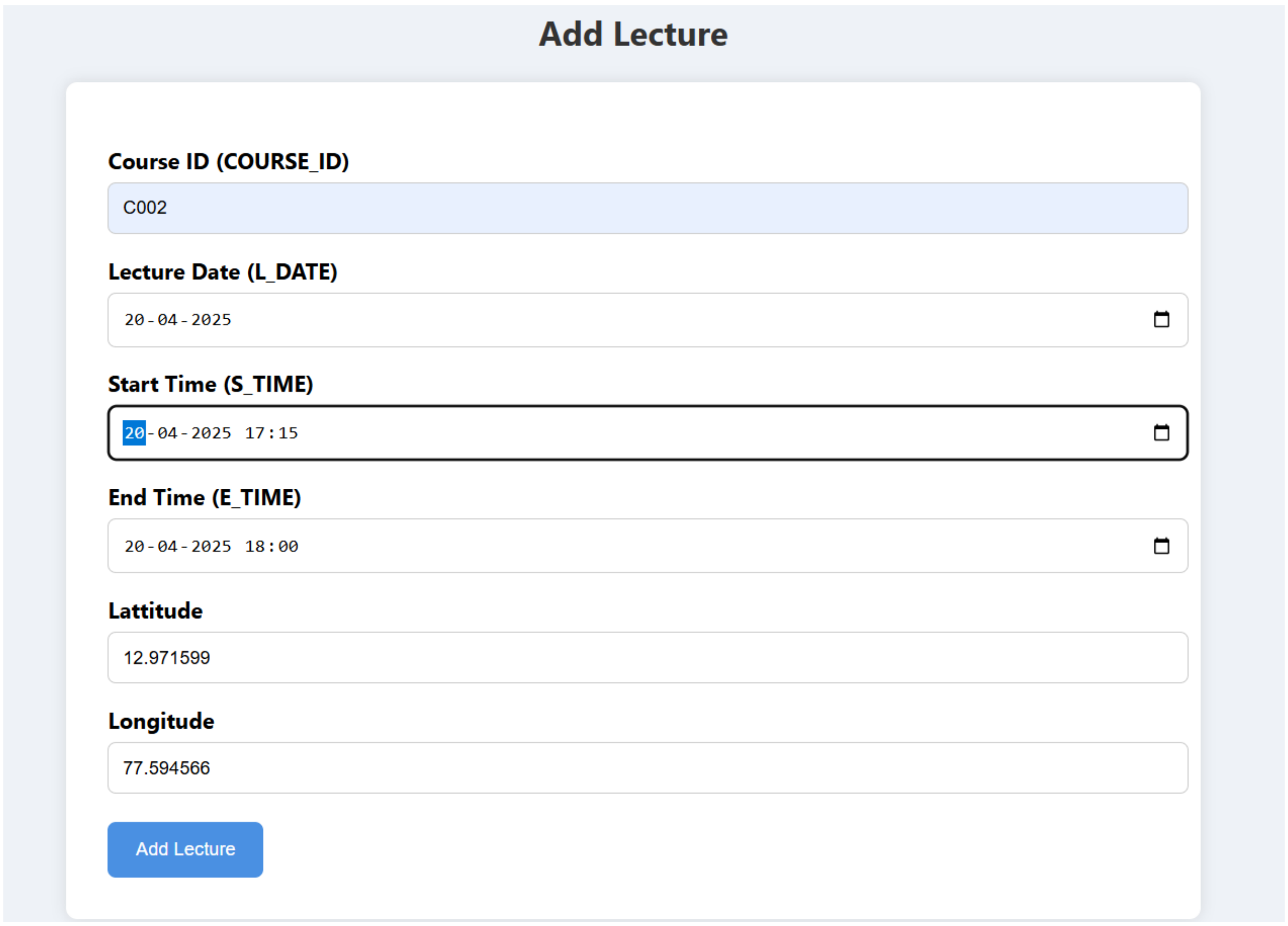
Phone Numbers :

9789456123



My Courses

Course Name	Course ID	Total Lectures	Lectures Taken	Lectures Left	Actions
Operating Systems	C002	51	10	41	Add Update Delete



Add Lecture

Course ID (COURSE_ID)

C002

Lecture Date (L_DATE)

20-04-2025

Start Time (S_TIME)

20-04-2025 17:15

End Time (E_TIME)

20-04-2025 18:00

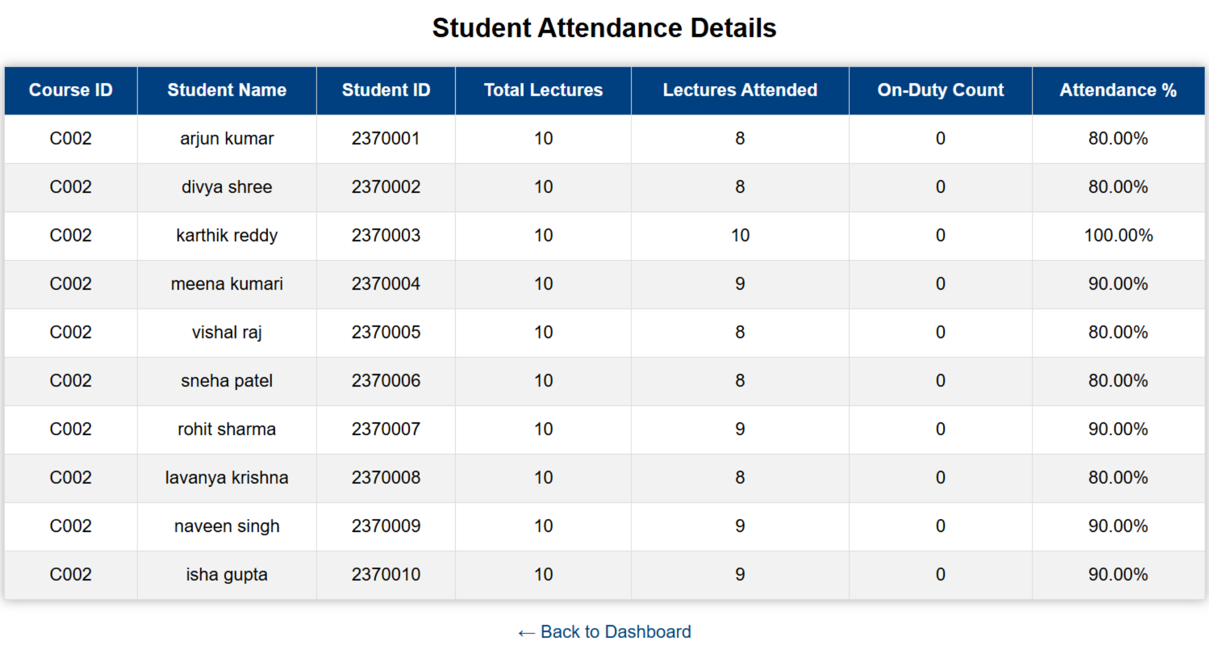
Lattitude

12.971599

Longitude

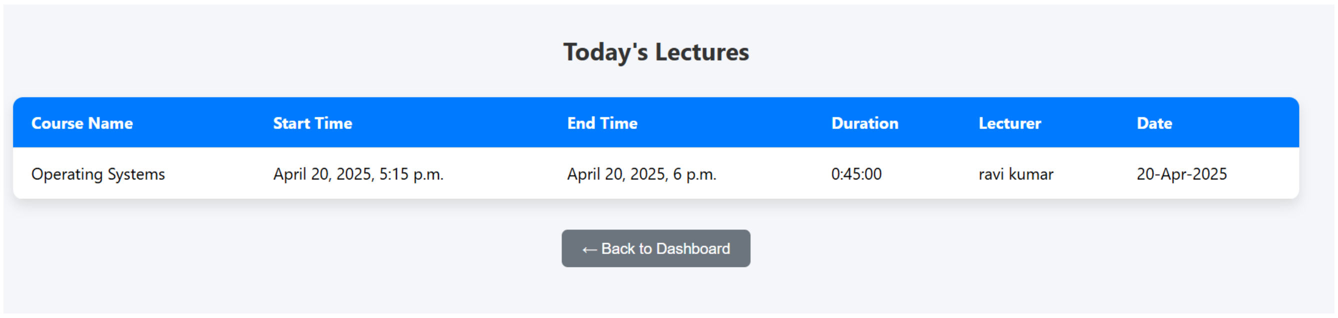
77.594566

Success ! Lecture created with Lecture id L01302



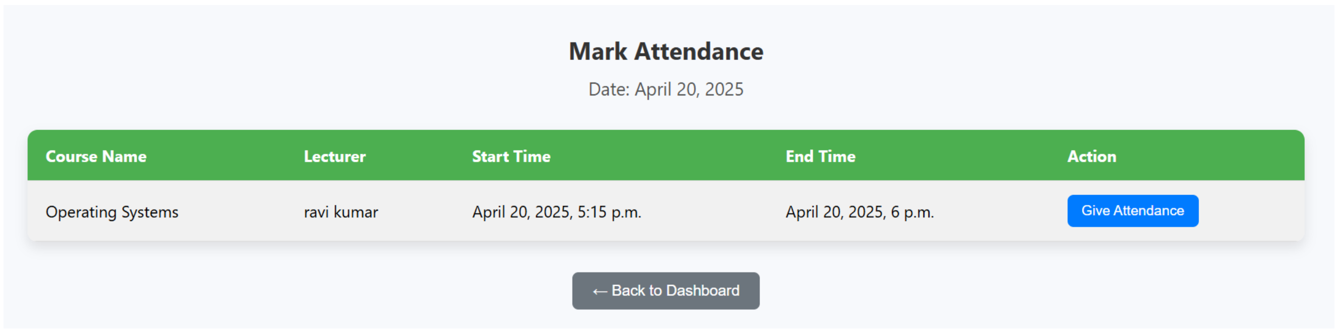
Student Attendance Details

Course ID	Student Name	Student ID	Total Lectures	Lectures Attended	On-Duty Count	Attendance %
C002	arjun kumar	2370001	10	8	0	80.00%
C002	divya shree	2370002	10	8	0	80.00%
C002	karthik reddy	2370003	10	10	0	100.00%
C002	meena kumari	2370004	10	9	0	90.00%
C002	vishal raj	2370005	10	8	0	80.00%
C002	sneha patel	2370006	10	8	0	80.00%
C002	rohit sharma	2370007	10	9	0	90.00%
C002	lavanya krishna	2370008	10	8	0	80.00%
C002	naveen singh	2370009	10	9	0	90.00%
C002	isha gupta	2370010	10	9	0	90.00%



Today's Lectures

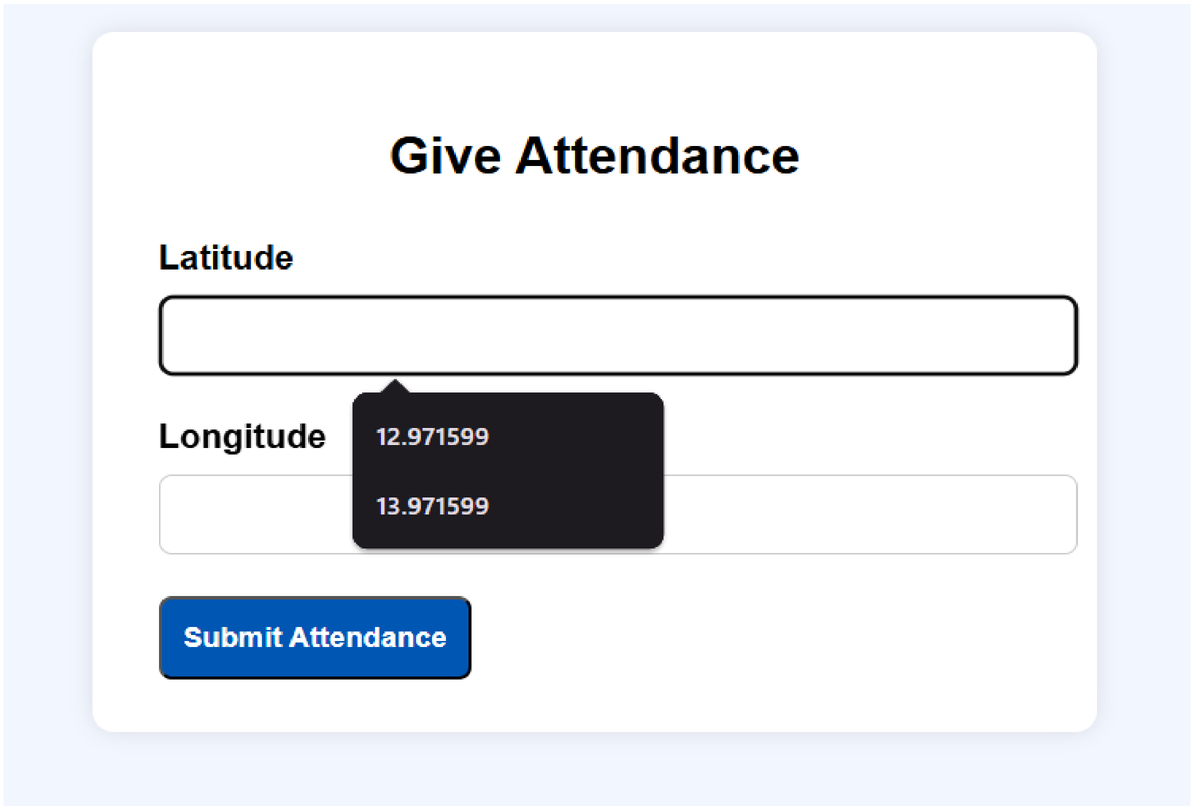
Course Name	Start Time	End Time	Duration	Lecturer	Date
Operating Systems	April 20, 2025, 5:15 p.m.	April 20, 2025, 6 p.m.	0:45:00	ravi kumar	20-Apr-2025



Mark Attendance

Date: April 20, 2025

Course Name	Lecturer	Start Time	End Time	Action
Operating Systems	ravi kumar	April 20, 2025, 5:15 p.m.	April 20, 2025, 6 p.m.	Give Attendance



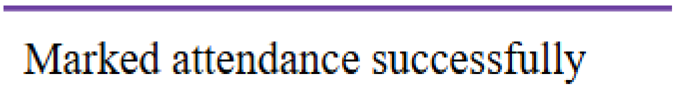
Give Attendance

Latitude

Longitude

12.971599

13.971599



Marked attendance successfully

Department of Computer Science and Engineering

Learning Outcomes :-

- I have got Practical Knowledge on Database Connectivity with Python by this Project.
- I have learnt to use SQL Database for developing some high level application or software applications.
- I have learnt to use SQL Queries for retrieving, inserting, Updating or deleting records in the Database using Some high level Programs (Python).