Chittagong University Campus Event Management System

Database Project Report

Group-37

A project submitted to Dr. Rudra Pratap Deb Nath, Associate Professor, Department of Computer Science and Engineering, Chittagong University (CU) in partial fulfillment of the requirements for the Database Systems Lab course. The project is not submitted to any other organization at the same time.

Contents

List of Figures

List of Tables

Listings

1 Requirement Gathering and Analysis

Requirement analysis in this project involves identifying the data that needs to be stored in the database and understanding how it will be accessed by different users. The system is being developed after discussing needs with one of the campus event organizers. Ensuring clear and complete requirements before beginning is essential for creating a system that meets the needs of all stakeholders.

1.1 Process of Requirement Analysis

1.1.1 Discussion with an Event Organizer and students

Discussions with an event organizer and a student form the core of the analysis. Through these conversations, the aim is to identify the problems encountered in organizing campus events, managing attendees, and allocating resources. Below are some key points from the discussion.

- **Question 1:** What are some challenges in managing events and facilitating connections among students and teachers?
- **Event Organizer:** We face issues organizing events across various interests, as we have to coordinate with departments and manage resources like locations and materials. Also, students may not know about events that match their interests, limiting opportunities to connect.
- **Question 2:** How does the current system limit participant engagement?
- **Student:** Without a centralized platform, it's challenging to keep track of events and connect with others who share similar interests. We want a way to see all available events, know who's attending, and connect with participants after events.
- **Question 3:** What improvements would you like to see in event notifications and updates?
- **Event Organizer:** Notifications for event updates, changes, or cancellations should be instant and accessible. Participants often don't get updates in time, which can make engagement difficult.

1.1.2 Understanding the Requirement

After discussing these points, it was concluded that the system must handle all event-related data automatically, minimizing errors and delays. The system should allow organizers and participants to manage and access information seamlessly and ensure reliable, timely communication of updates.

Basic information requirements identified are:

- **Event Information**: Event name, date, time, location, and organizer.
- **Participant Information**: Name, ID, email.
- **Location Information**: area details, capacity, and availability for booking.

1.2 Identifying Stakeholders

Since the system is being developed to address event management challenges faced on campus, the key stakeholders are:

- **Event Organizers** (teachers, students, and staff involved in planning and managing events).
- **Participants** (students, teachers).
- **Campus Administration** (managing location bookings and maintaining event records).

As the Question , I aim to create a solution that is user-friendly and effective for all stakeholders.

1.3 System Specification

After analyzing requirements and identifying stakeholders, a **web-based system** has been selected for development. The next step is to design a conceptual model, focusing on a high-level view to confirm that it aligns with the specified requirements.

The system will include features for event creation, participant registration, resource management, and real-time notifications, ensuring that all necessary information is accessible and manageable for everyone involved.

2 Conceptual Model for Campus Event Management System

The conceptual model for the Campus Event Management System was developed using the Entity-Relationship Model (ERM). The ERM was chosen because of its clarity in representing data structures and its relevance to our coursework. ER diagrams define the logical structure of the database, showing how data elements and their relationships interact within the system.

2.1 Entity-Relationship Model

An **Entity-Relationship Diagram (ERD)** visually represents the data structure of the Campus Event Management System, showing how entities and their attributes relate to each other.

Key Concepts in the ER Model:

- **Entity**: An entity represents a real-world object or concept, like an event or user. Entities are represented as tables in the database, with each row representing a specific instance.
- **Attribute**: Attributes represent the properties or characteristics of an entity. For example, attributes of the `Event` entity include `event_name`, `event_date`, and `location_id`.
- **Relationship**: Relationships define the interactions between entities. Common types of relationships include:
  - **One-to-One**
  - **One-to-Many**
  - **Many-to-One**
  - **Many-to-Many**

2.2 Detailing the Entity-Relationship Model

From the analysis, the following **entities** and **relationships** have been identified for the Campus Event Management System.

2.2.1 Entity Types

1. **User**: Represents users of the system with attributes:

- `username` (Primary Key)
- `name`
- `email`
- `password`
- `user_type` (enum: ‘student‘, ‘teacher‘)
- `contact_number`

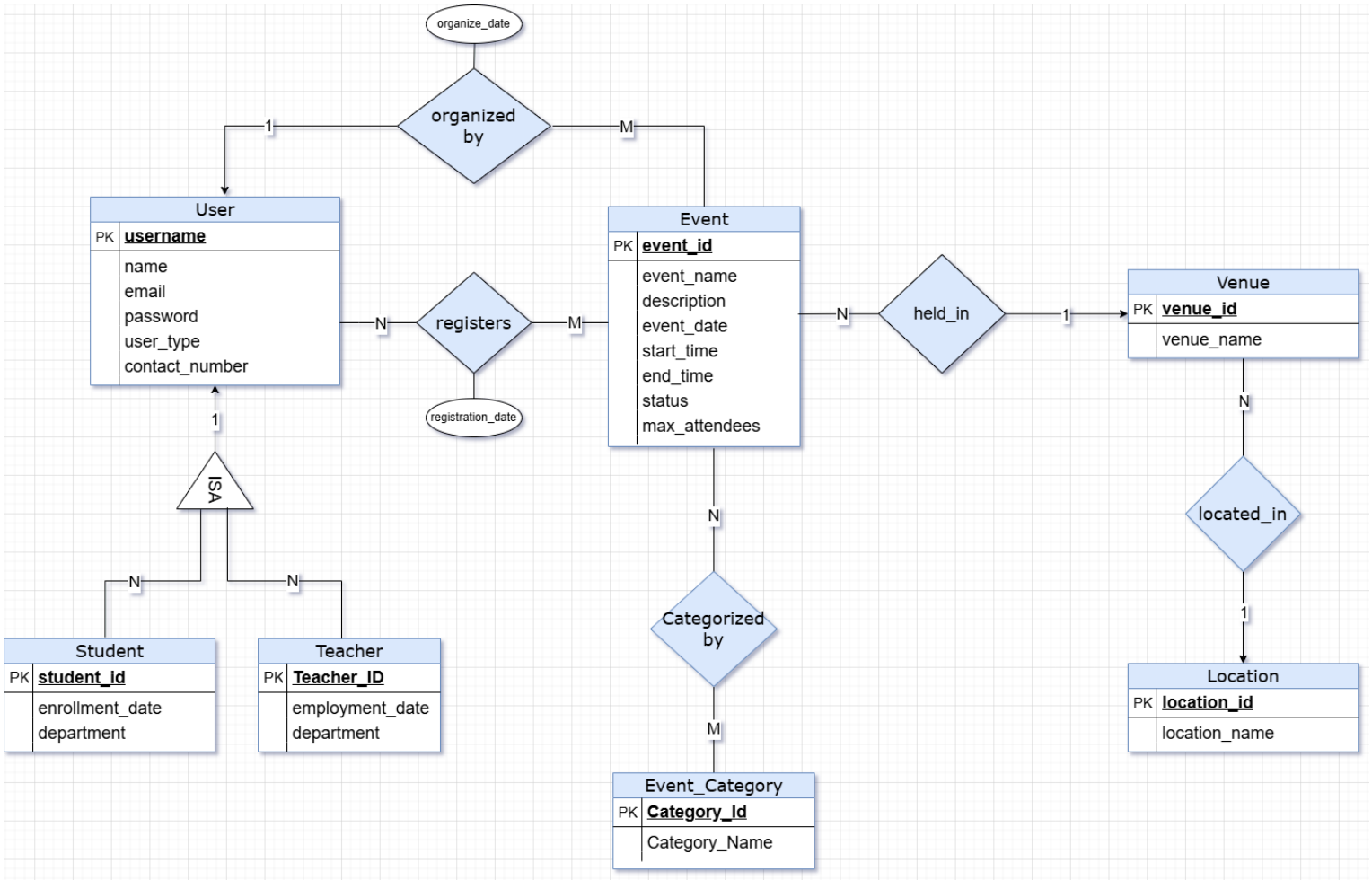
2. **Student**: Represents students who may attend events, with attributes:
   - `student_id` (Primary Key)
   - `enrollment_date`
   - `username` (Foreign Key referencing `User.username`)
3. **Teacher**: Represents faculty members involved in events, with attributes:
   - `teacher_id` (Primary Key)
   - `employment_date`
   - `username` (Foreign Key referencing `User.username`)
4. **Event**: Represents each campus event, with attributes:
   - `event_id` (Primary Key)
   - `event_name`
   - `description`
   - `event_date`
   - `start_time`
   - `end_time`
   - `status` (enum: ‘pending‘, ‘ongoing‘, ‘finished‘)
   - `max_attendees`
   - `category_id` (Foreign Key referencing `Event_Category.category_id`)
   - `location_id` (Foreign Key referencing `Location.location_id`)
5. **Location**: Represents locations of the venues:
   - `location_id` (Primary Key)
   - `location_name`

6. **Venue**: Represents locations where events are held, with attributes:
    - `venue_id` (Primary Key)
    - `venue_name`
    - `location_id` (Foreign Key referencing `Location.location_id`)
7. **Event Category**: Represents categories for events, with attributes:
    - `category_id` (Primary Key)
    - `category_name`

2.2.2 Relationship Types

The relationships between the entities are as follows:

- **Organized_By**: Connects `User` and `Event`, where each user (acting as an organizer) can manage multiple events.
    - `username` (Foreign Key referencing `User.username`)
    - `event_id` (Foreign Key referencing `Event.event_id`)
- **Registers**: Connects `User` (as a participant) and `Event`, where a participant can register for multiple events, and each event can have multiple participants.
    - `username` (Foreign Key referencing `User.username`)
    - `event_id` (Foreign Key referencing `Event.event_id`)
    - `registration_date`
- **held At**: Connects `Event` and `Venue`, where each event is held at a specified venue.
    - `venue_id` (Foreign Key referencing `Venue.venue_id`)
    - `event_id` (Foreign Key referencing `Event.event_id`)
- **located in**: Connects `Venue` and `Location`, where each venue is located in a specified location.
    - `venue_id` (Foreign Key referencing `Venue.venue_id`)
    - `location_id` (Foreign Key referencing `Location.location_id`)



This ER model provides a high-level structure for managing campus events, representing how data is organized within the Campus Event Management System. The relationships between entities allow efficient linking of users, events, locations, and categories, facilitating seamless interactions within the system.

3 Logical Modelling

After defining the conceptual model, we proceed with the logical data model for the Campus Event Management System. The logical model outlines the structure of the data items and their relationships in a formal way. For our system, we use the relational model.

3.1 Relational Model

In a relational model, data and their relationships are represented through a collection of interlinked tables. Each table contains rows and columns, where each column represents an attribute of an entity, and each row holds a record. The relational schema defines how each table (relation) is structured, specifying the table name and a list of attribute names, each tied to a specific domain.

3.1.1 Tables and Attributes

- **User**
  `user(username, username, email, password, user_type,contact_number)`
  Primary Key: username

- **Student**
  `student(student_id, enrollment_date, username)`
  Primary Key: student_id
  Foreign Key: username → user(username)

- **Teacher**
  `teacher(teacher_id, employment_date, username)`
  Primary Key: teacher_id
  Foreign Keys:
  username → user(username)

- **Venue**
  `Venue(venue_id, venue_name)`
  Primary Key: venue_id

- **Location**
  `location(location_id, location_name)`
  Primary Key: location_id

- **Event_Category**
  `event_category(category_id, category_name)`
  Primary Key: category_id

- **Event**
  `event(event_id, event_name, description, event_date, start_time, end_time, status, max_attendees, category_id, venue_id)`
  Primary Key: event_id
  Foreign Keys:
  category_id → event_category(category_id)
  venue_id → Venue(venue_id)

- **Registers**
  `registers(username, event_id, registration_date)`
  Primary Key: (username, event_id)
  Foreign Keys:
  username → user(username)
  event_id → event(event_id)

- **Organized_By**
  `organized_by(username, event_id)`
  Primary Key: (username, event_id)
  Foreign Keys:
  username → user(username)
  event_id → event(event_id)

4 Normalization of Database Tables

Normalization is a process used to minimize or remove data redundancy in a set of relations. The primary goal of normalization is to eliminate anomalies that can occur during data insertion, update, or deletion, thus making the database consistent, dependency-preserving, and free of redundancy. Below, we analyze the normalization forms (specifically 3NF) for each table in the campus event management database.

4.1 Normalizing user Table

The `user` table has the attributes:

$$\{\texttt{username}, \texttt{name}, \texttt{email}, \texttt{password}, \texttt{user\_type}, \texttt{contact\_number}\}$$

username	name	email	password	user_type	contact_number
arafat123	Arafat Sheikh	arafat@example.com	password789	student	01733333333
jane_smith	Jane Smith	jane@example.com	password456	teacher	01722222222
lisa_brown	Lisa Brown	lisa@example.com	password321	teacher	01744444444
nazmul	Nazmul Hasan	namzul@example.com	password654	student	01755555555
sanzid	Sanzid Islam	sanzid@example.com	password123	student	01711111111

Based on the demo data, we identify the following functional dependencies and analyze the candidate keys, prime attributes, and non-prime attributes to determine its normalization status.

- **Functional Dependencies:**
  - $\underline{\texttt{username}} \rightarrow \texttt{username,name, email, password, user\_type, contact\_number}$
  - $\texttt{email} \rightarrow \texttt{username,name, email, password, user\_type, contact\_number}$
  - $\texttt{contact\_number} \rightarrow \texttt{username,name, email, password, user\_type, contact\_number}$
- **Candidate Keys:** `username, email, contact_number`
- **Prime Attributes:** `username,email, contact_number`
- **Non-Prime Attributes:** `name, password, user_type`

- **Analysis:** Based on our analysis, all the functional dependencies in the `user` table have a candidate key on the left-hand side. This satisfies the requirements of 3NF, and no transitive dependencies are present. All non-prime attributes (name, email, password, user_type, department, and contact_number) are fully functionally dependent on the candidate key, `username`.
- **Conclusion:** The `user` table is in 3NF.

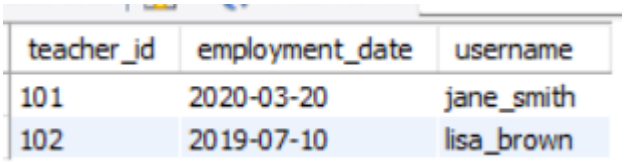
4.2 Normalizing student Table

The `student` table has the attributes:

$$\{\texttt{student\_id}, \texttt{enrollment\_date}, \texttt{username}\}$$

student_id	enrollment_date	username
22701017	2021-09-01	arafat123
22701030	2023-01-10	nazmul
22701065	2022-01-15	Sanzid

- **Functional Dependencies:**
  - `student_id` $\rightarrow$ `enrollment_date, username`
- **Candidate Keys:** `student_id`
- **Prime Attributes:** `student_id`
- **Non-Prime Attributes:** `enrollment_date, username`
- **Analysis:** In this table, all functional dependencies have a candidate key on the left-hand side, fulfilling the conditions of 3NF. There are no transitive dependencies, and all non-prime attributes are fully functionally dependent on `student_id`.
- **Conclusion:** The `student` table is in 3NF.



teacher_id	employment_date	username
101	2020-03-20	jane_smith
102	2019-07-10	lisa_brown

4.3 Normalizing teacher Table

The `teacher` table has the attributes:

$$\{\texttt{teacher\_id}, \texttt{employment\_date}, \texttt{username}\}$$

- **Functional Dependencies:**
  - `teacher_id` → `employment_date, username`
- **Candidate Keys:** `teacher_id`
- **Prime Attributes:** `teacher_id`
- **Non-Prime Attributes:** `employment_date, username`
- **Analysis:** This table meets the conditions of 3NF as all functional dependencies have a candidate key on the left-hand side. There are no transitive dependencies, and all non-prime attributes are fully functionally dependent on `teacher_id`.
- **Conclusion:** The `teacher` table is in 3NF.

4.4 Normalizing venue Table

The `venue` table has the attributes:

$$\{\texttt{venue\_id}, \texttt{venue\_name}, \texttt{location\_id}\}$$

- **Functional Dependencies:**
  - `venue_id` → `venue_name, location_id`
- **Candidate Keys:** `venue_id`
- **Prime Attributes:** `venue_id`

venue_id	venue_name	location_id
1	Shahid Minar	1
2	Jarul Tola	1
3	room 413	3
4	room 412	3
5	Virtual Classroom	3
6	Physics Workshop	4
7	Badminton Ground 1	5
8	Teletalk Pahar	6

- **Non-Prime Attributes:** `venue_name,location_id`
- **Analysis:** The functional dependency has a candidate key (`venue_id`) on the left-hand side. This satisfies the conditions of 3NF, and there are no transitive dependencies. All non-prime attributes (`venue_name,location_id`) are fully functionally dependent on `venue_id`.
- **Conclusion:** The `venue` table is in 3NF.

4.5 Normalizing location Table

The `location` table has the attributes:

$$\{\texttt{location\_id}, \texttt{location\_name}\}$$

location_id	location_name
1	Shahid Minar
2	Botanical Gargen
3	Engineering Faculty
4	Science Faculty
5	Central Field
6	TeleTalk Pahar

- **Functional Dependencies:**
  - `location_id` $\rightarrow$ `location_name`

- **Candidate Keys:** `location_id`
- **Prime Attributes:** `location_id`
- **Non-Prime Attributes:** `location_name`
- **Analysis:** The functional dependency has a candidate key (`location_id`) on the left-hand side. This satisfies the conditions of 3NF, and there are no transitive dependencies. All non-prime attributes (`location_name`) are fully functionally dependent on `location_id`.
- **Conclusion:** The `location` table is in 3NF.

4.6 Normalizing event_category Table

The `event_category` table has the attributes:

$$\{\texttt{category\_id}, \texttt{category\_name}\}$$

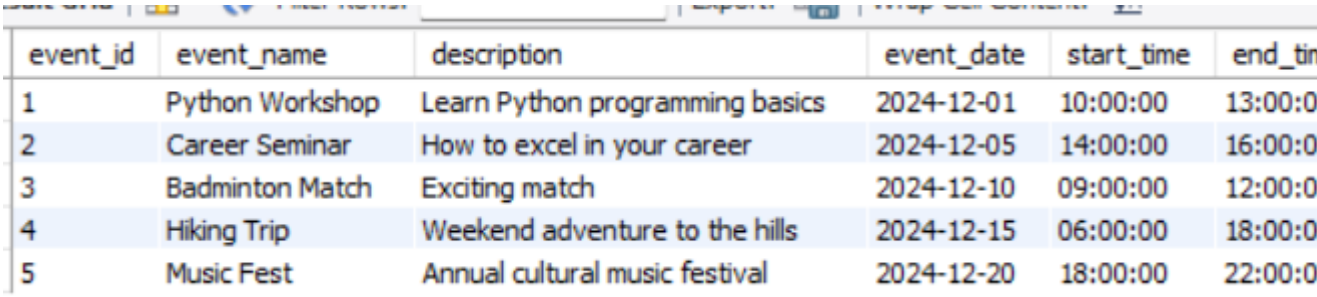
category_id	category_name
1	Workshop
2	Seminar
3	Sports
4	Adventure
5	Cultural

- **Functional Dependencies:**
  - `category_id` $\rightarrow$ `category_name`
- **Candidate Keys:** `category_id`
- **Prime Attributes:** `category_id`
- **Non-Prime Attributes:** `category_name`
- **Analysis:** The functional dependency has a candidate key (`category_id`) on the left-hand side, meeting 3NF requirements. There are no transitive dependencies, and all non-prime attributes are fully functionally dependent on `category_id`.
- **Conclusion:** The `event_category` table is in 3NF.

4.7 Normalizing event Table

The `event` table has the attributes:

{`event_id`, `event_name`, `description`, `event_date`, `start_time`, `end_time`, `status`, `max_attendee`



event_id	event_name	description	event_date	start_time	end_tir
1	Python Workshop	Learn Python programming basics	2024-12-01	10:00:00	13:00:0
2	Career Seminar	How to excel in your career	2024-12-05	14:00:00	16:00:0
3	Badminton Match	Exciting match	2024-12-10	09:00:00	12:00:0
4	Hiking Trip	Weekend adventure to the hills	2024-12-15	06:00:00	18:00:0
5	Music Fest	Annual cultural music festival	2024-12-20	18:00:00	22:00:0

- **Functional Dependencies:**
  - `event_id` → `event_name, description, event_date, start_time, end_time, status, max_attendees, category_id, venue_id`
- **Candidate Keys:** `event_id`
- **Prime Attributes:** `event_id`
- **Non-Prime Attributes:** `event_name, description, event_date, start_time, end_time, status, max_attendees, category_id, venue_id`
- **Analysis:** All functional dependencies have a candidate key (`event_id`) on the left-hand side. There are no transitive dependencies, and all non-prime attributes are fully functionally dependent on `event_id`, fulfilling 3NF.
- **Conclusion:** The `event` table is in 3NF.

4.8 Normalizing registers Table

The `registers` table has the attributes:

{`username`, `event_id`, `registration_date`}

- **Functional Dependencies:**
  - `(username, event_id)` → `registration_date`

- **Candidate Keys:** (username, event_id)
- **Prime Attributes:** username, event_id
- **Non-Prime Attributes:** registration_date
- **Analysis:** The only functional dependency has a composite candidate key (username, event_id) on the left-hand side, fulfilling 3NF. There are no transitive dependencies, and the non-prime attribute registration_date is fully functionally dependent on the composite key.
- **Conclusion:** The registers table is in 3NF.

4.9 Normalizing organized_by Table

The organized_by table has the attributes:

$$\{\texttt{username}, \texttt{event\_id}\}$$

- **Functional Dependencies:**
  - (username, event_id) → [no additional attributes]
- **Candidate Keys:** (username, event_id)
- **Prime Attributes:** username, event_id
- **Non-Prime Attributes:** None
- **Analysis:** The table consists only of the composite candidate key (username, event_id), so there are no additional attributes to check for dependency. Thus, it satisfies 3NF.
- **Conclusion:** The organized_by table is in 3NF.