# NORMALISATION REPORT

For

## REVELS MANAGEMENT SYSTEM

Version 1.0

Prepared by

Akhil Varanasi 57 - 230953496

Saurabh Sharma 43 - 230953374

Shreyas Kumar PM 33- 230953300

**Manipal Institute of Technology**

**1st April 2025**

# Normalization Report: Revels Management System Database

### 5.1 Performance Requirements

- NFR-1: The system shall support at least 1000 concurrent users without performance degradation.
- NFR-2: Average page load times must be less than 3 seconds under normal operating conditions.
- NFR-3: The system should process registration and scoring inputs in real time.

### 5.1.1 Normalization Check

The database design for the Fest Management System has been rigorously normalized to ensure data integrity, eliminate redundancy, and support efficient queries. The normalization process has been evaluated at multiple levels:

## 1. Colleges Table

**Attributes:**

- College_ID (Primary Key)
- Name
- Location
- Website
- Created_At
- Updated_At

**Functional Dependencies:**

- College_ID → Name, Location, Website, Created_At, Updated_At
- Name → College_ID (Assuming college names are unique)

---

# 2. Participants Table

**Attributes:**

- ID (Primary Key)
- Name
- College_ID (Foreign Key)
- Contact
- Email
- Created_At
- Updated_At
- Login_ID

**Functional Dependencies:**

- ID → Name, College_ID, Contact, Email, Created_At, Updated_At, Login_ID
- Email → ID, Name, College_ID, Contact, Created_At, Updated_At, Login_ID (Assuming emails are unique)
- Contact → ID, Name, College_ID, Email, Created_At, Updated_At, Login_ID (Assuming contacts are unique)

---

# 3. Judges Table

**Attributes:**

- Judge_ID (Primary Key)
- Name
- Contact
- Email
- Created_At
- Updated_At
- Login_ID

**Functional Dependencies:**

- Judge_ID → Name, Contact, Email, Created_At, Updated_At, Login_ID
- Email → Judge_ID, Name, Contact, Created_At, Updated_At, Login_ID
- Contact → Judge_ID, Name, Email, Created_At, Updated_At, Login_ID (If contact is unique)

---

# 4. Events Table

**Attributes:**

- Event_ID (Primary Key)
- Name
- Description
- Category
- Time

- Venue
- Participation_Limit
- Created_At
- Updated_At

**Functional Dependencies:**

- Event_ID → Name, Description, Category, Time, Venue, Participation_Limit, Created_At, Updated_At
- (Event_ID, Time) → Venue (An event at a particular time can have only one venue)
- Name → Event_ID (Assuming event names are unique)

---

# 5. Teams Table

**Attributes:**

- Team_ID (Primary Key)
- Event_ID (Foreign Key)
- Team_Name
- Created_At
- Updated_At

**Functional Dependencies:**

- Team_ID → Event_ID, Team_Name, Created_At, Updated_At
- (Event_ID, Team_Name) → Team_ID (Each team name is unique within an event)

---

# 6. TeamMembers Table

**Attributes:**

- Team_ID (Foreign Key)
- Participant_ID (Foreign Key)
- Role

**Functional Dependencies:**

- (Team_ID, Participant_ID) → Role
- Team_ID → Event_ID (Through foreign key relationship with Teams)

---

# 7. Registrations Table

**Attributes:**

- Registration_ID (Primary Key)
- Participant_ID (Foreign Key)
- Event_ID (Foreign Key)
- Team_ID (Foreign Key)
- Status
- Registration_Date
- Updated_At

**Functional Dependencies:**

- Registration_ID → Participant_ID, Event_ID, Team_ID, Status, Registration_Date, Updated_At

- (Participant_ID, Event_ID) → Registration_ID, Team_ID, Status, Registration_Date, Updated_At (Each participant can register only once for an event)
- Team_ID → Event_ID (Since a team belongs to an event)

---

# 8. Judging Table

**Attributes:**

- Judging_ID (Primary Key)
- Judge_ID (Foreign Key)
- Event_ID (Foreign Key)
- Score
- Comments
- Judging_Date
- Updated_At

**Functional Dependencies:**

- Judging_ID → Judge_ID, Event_ID, Score, Comments, Judging_Date, Updated_At
- (Judge_ID, Event_ID) → Score, Comments, Judging_Date, Updated_At (Each judge gives one score per event)

---

# 9. Committees Table

**Attributes:**

- Committee_ID (Primary Key)

- Name
- Role
- Contact
- Email
- Created_At
- Updated_At
- Login_ID

**Functional Dependencies:**

- Committee_ID → Name, Role, Contact, Email, Created_At, Updated_At, Login_ID
- Email → Committee_ID, Name, Role, Contact, Created_At, Updated_At, Login_ID
- Contact → Committee_ID, Name, Role, Email, Created_At, Updated_At, Login_ID (If contacts are unique)

---

# 10. Schedules Table

**Attributes:**

- Schedule_ID (Primary Key)
- Event_ID (Foreign Key)
- Start_Time
- End_Time
- Venue
- Created_At

- Updated_At

**Functional Dependencies:**

- Schedule_ID → Event_ID, Start_Time, End_Time, Venue, Created_At, Updated_At
- (Event_ID, Start_Time, End_Time) → Venue (A specific event time slot can have only one venue)
- (Venue, Start_Time, End_Time) → Event_ID (Each venue at a given time can have only one event)

---

# 11. Logins Table

**Attributes:**

- User_ID (Primary Key)
- Email
- Password_Hash
- Role
- Created_At
- Updated_At

**Functional Dependencies:**

- User_ID → Email, Password_Hash, Role, Created_At, Updated_At
- Email → User_ID, Password_Hash, Role, Created_At, Updated_At (Each email must be unique)

---

# Normalization Steps

## First Normal Form (1NF)

A table is in 1NF if:

- All attributes have atomic values.
- Each column contains values of a single type.
- Each row is uniquely identifiable (has a primary key).

✅ All tables satisfy 1NF as they contain atomic attributes and have primary keys.

---

## Second Normal Form (2NF)

A table is in 2NF if:

- It is already in 1NF.
- Every non-key attribute is fully functionally dependent on the whole primary key.

✅ All tables satisfy 2NF as there are no partial dependencies.

---

## Third Normal Form (3NF)

A table is in 3NF if:

- It is already in 2NF.
- There are no transitive dependencies.

✅ All tables satisfy 3NF as there are no transitive dependencies.

---

## Boyce-Codd Normal Form (BCNF)

A table is in BCNF if:

- It is already in 3NF.
- Every determinant is a superkey.

✅ All tables satisfy BCNF as there are no non-trivial functional dependencies where a non-superkey determines another attribute.

---

This structured schema ensures a well-normalized database with efficient data organization and minimal redundancy.