Project team #27 - Event Planner Management System(EPMS)

Team Members
Sathyaram Venkatesan
Vikas Siva Ravindran
Abineshwar Angamuthu Matheswaran

PROJECT PROPOSAL

Content:

Event Planning management system provides an efficient access to the database details for a scheduled conference event. The EPMS consists of a wide range of data ranging from the list of attendees, available dates, accommodation details, resource persons, sponsors.
The basic content of the project involves designing, storing, organizing and retrieving details pertaining to all the events in a conference and ensuring a foolproof database management system.

Entities: Admin, User, Attendee, Accommodation, Volunteer, Payment, Sponsor, Event, Organizer, Notification.

Scope :
The scope of the projects extends to :

- Hassle-free Administrative access.
- Registration for the conference.
- Easy communication between the attendees and organizers.
- Ensuring successful coordination and execution of the conference events.

Objective:
The goal is to design an EPMS that provides a holistic approach to design, develop and manage the database system.

PROJECT ENVIRONMENT

The development environment consists of MySQL Workbench (8.0.12) for database management system and HTML, CSS, JavaScript (ES6) for the User Interface along with the Spring Data JPA for the server side data retrieval. Operational environment involves various users registering and participating in various events. Event planner management system (EPMS) assures efficient management of the event details resulting in a user-friendly system. The system holds the data in a centralized manner which is available to all the event managers, attendees, administrators, and Sponsors. The event manager can keep records of participants and system can easily communicate with them. It is socially feasible as it is time and cost-saving compared to traditional file management system.

HIGH-LEVEL REQUIREMENTS

Initial user roles

User Role	**Description**
Admin	Access to the list of events, organizers for specific events, attendees registered for a particular event, views the list of sponsors and payment providers.
Organizer	Access to the list of attendees for a event, manages the accommodation details of the attendees, views the list of sponsors, volunteers for an event.
User	Views the list of Events, able to register for a single event.
Sponsor	Sponsors able to view the list of Events in the Conference.
Volunteers	Volunteers able to contact the organizer of a specific event.

Initial user story descriptions

Story ID	Story description
Admin	As an admin, I want to access the events at the conference. As an admin, I need to view the list of sponsors for the event. As an admin, I need to assign the organizer for an event.
Organizer	As an organizer, I want to view the list of attendees for a specific event. As an organizer, I want to manage the accommodation details of the attendees. As an organizer, I want to assign volunteer for a specific event. As an organizer, I want to contact attendees, volunteers, and sponsors.
User	As a user, I want to view the list of events. As a user, I want to register for an event.
Attendee	As an attendee, I want to view registered event details. As an attendee, I want to contact the organizer of an event.
Sponsors	As a sponsor, I want to view the events at the conference. As a sponsor, I want to contact the organizer of a specific event.
Volunteer	As a volunteer, I want to contact the organizer of a specific event.

HIGH-LEVEL CONCEPTUAL DESIGN

Entities:

1. Admin
2. Organizer
3. User
4. Attendee
5. Sponsor
6. Volunteer
7. Accomodation
8. Payment
9. Notification
10. Event

Relationships:

1. Admin creates/modifies/removes Event.
2. Admin assigns Organizer for an Event.
3. Organizer views list of attendees for an event.
4. Organizer assigns volunteers for an event.
5. User views the list of events.
6. User registers for the event.
7. User makes payment.
8. Attendee receives notification.
9. Attendee views accommodation details
10. Attendee contacts Organizer.
11. Sponsor views the list of events.
12. Sponsor contacts the Organizer.

Sprint 1

REQUIREMENTS

Story ID	Story description
US1	As an admin, I want to create and modify the events of the conference.
US2	As an admin, I need to view the list of sponsors for the event.
US3	As an admin, I need to assign the organizer for an event.
US4	As an organizer, I want to view the details of specific event.
US5	As an organizer, I want to manage the accommodation details of the attendees.
US6	As an organizer, I want to assign volunteer for a specific event.
US7	As an organizer, I want to contact attendees.
US8	As a user, I want to view and register the list of events.
US9	As an attendee, I want to view details of registered event
US10	As a sponsor, I want to view the events and contact the organizer at the conference.
US11	As a volunteer, I want to view list of events and contact the organizer of a specific event.

CONCEPTUAL DESIGN

1) Entity: **Admin**
   Attributes:
   - Admin_Id
   - First_Name
   - Last_Name
   - Password

   Entity: **Event**
   Attributes:
   - Event_Id
   - Event_Name
   - Event_Details
   - Organizer_Id
   - Volunteer_Id
   - Sponsor_Id

Relationship: **Admin** creates the **Events** of the conference.
Cardinality: One to Many
Participation:
- Admin has Partial participation
- Event has Total participation

2) Entity: **Organizer**
   Attributes:
   - Organizer_Id
   - First_Name
   - Last_Name
   - Email [Multivalued]
   - Password
   - PhoneNumber [Multivalued]

   Entity: **Attendee**
   Attributes:
   Attendee_Id

First_Name
Last_Name
Email [Multivalued]
Password
Phone_Number [Multivalued]
Event_Id

Relationship: **Organizer** contacts **Attendee**
Cardinality: One to Many
Participation:
Organizer has partial participation
Attendee has total participation

3) Entity: **Attendee**
Attributes:
Attendee_Id
First_Name
Last_Name
Email [Multivalued]
Password
Phone_Number [Multivalued]
Event_id

Entity: **Event**
Attributes:
Event_Id
Event_Name
Event_Details
Organizer_Id
Volunteer_Id
Sponsor_Id

Relationship: **Attendee** views **Event** details.
Cardinality: One to One
Participation:
Attendee has total participation
Event has partial participation

LOGICAL DESIGN

Table: **Admin**
Columns:
- Admin_Id
- First_Name
- Last_Name
- Password

Table: **Event**
Columns:
- Event_Id
- Event_Name
- Event_Details
- Organizer_Id [foreign key; references **Organizer_Id** of **Organizer**]
- Volunteer_Id [foreign key; references **Volunteer_Id** of **Volunteer**]
- Sponsor_Id [foreign key; references **Sponsor_Id** of **Sponsor**]

Each event has only one Organizer and Sponsor.

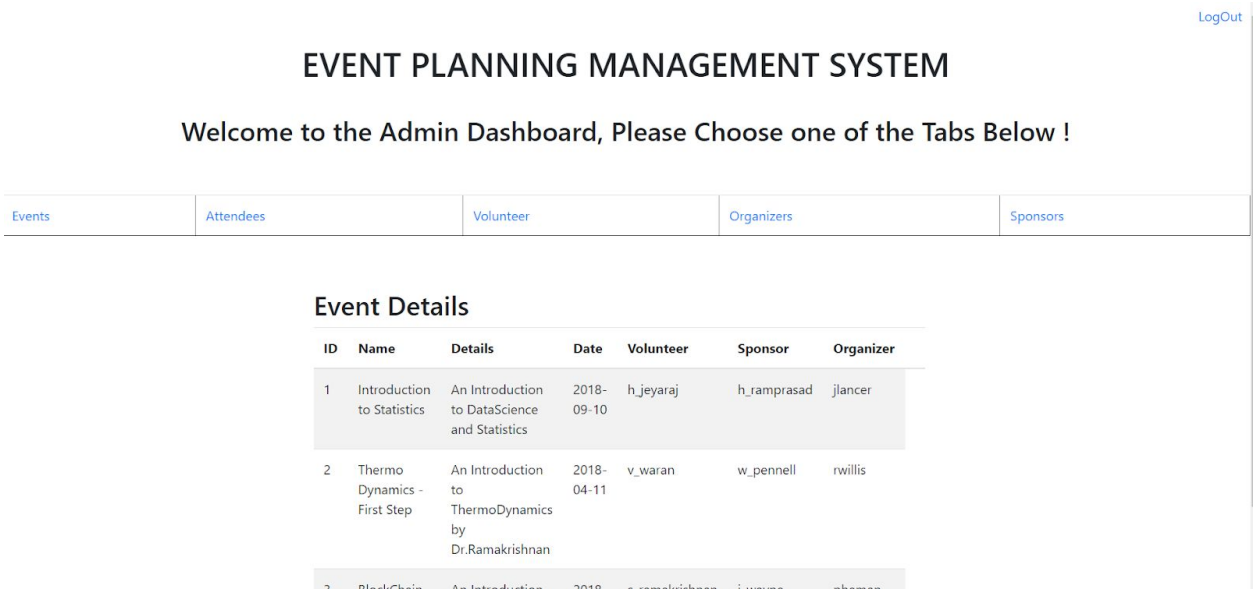
Table: **Organizer**
Columns:
- Organizer_Id
- First_Name
- Last_Name
- Email
- Password
- Phone_Number

Table: **Attendee**
Columns:
- Attendee_Id
- Event_Id[foreign key; references **Event_Id** of **Event**]
- First_Name
- Last_Name
- Email
- Password
- Phone_Number

SQL QUERIES

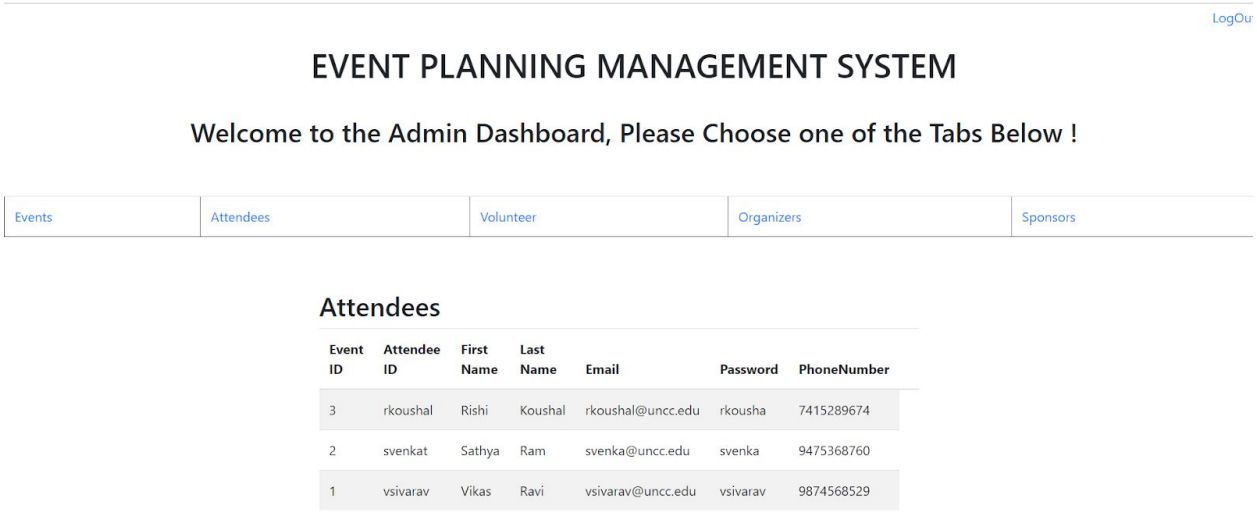
1) As an admin, I want to modify the events of the conference :

SELECT Event_Id, Event_Name, Event_Details,Event_Date,Volunteer_ID,Sponsor_Id, Organizer_Id from Event;,



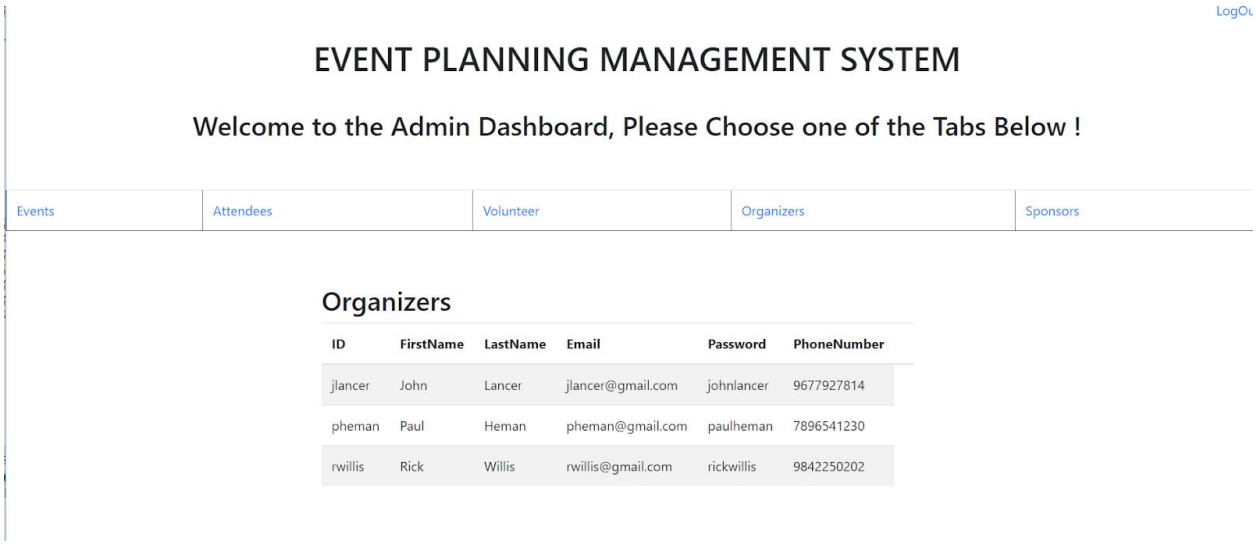
2) As an admin, I want to modify the list of attendees.

Select Event_Id,Attendee_ID,First_Name,Last_Name,Email,Password,Phone_Number from attendee;

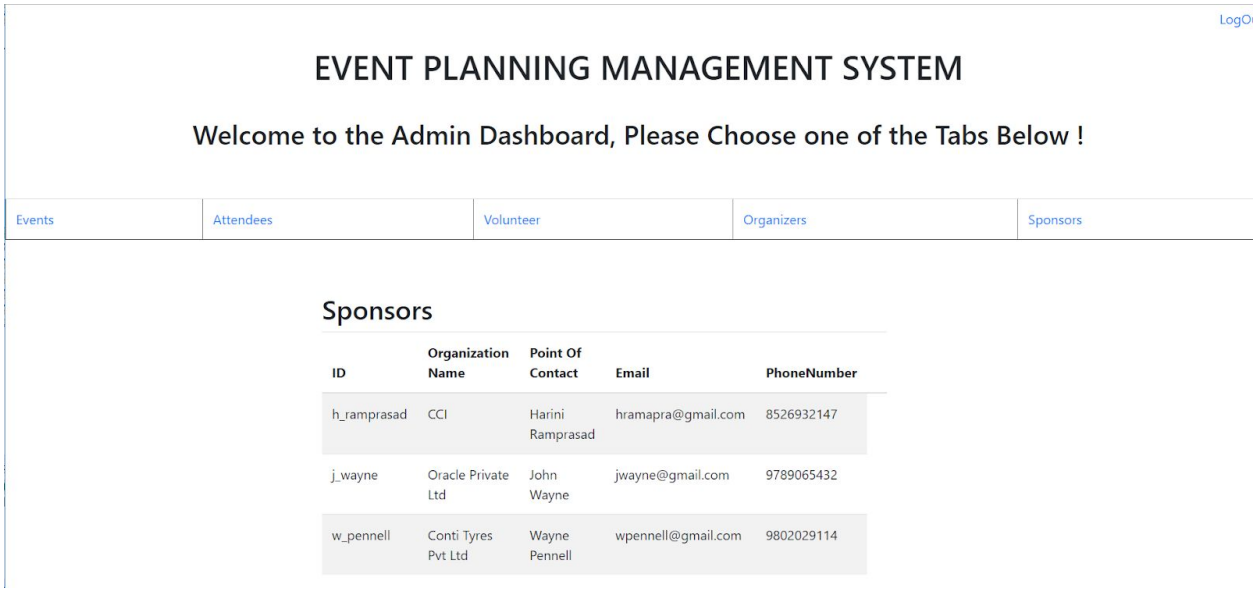


3) As an admin, I want to view list of organizers.

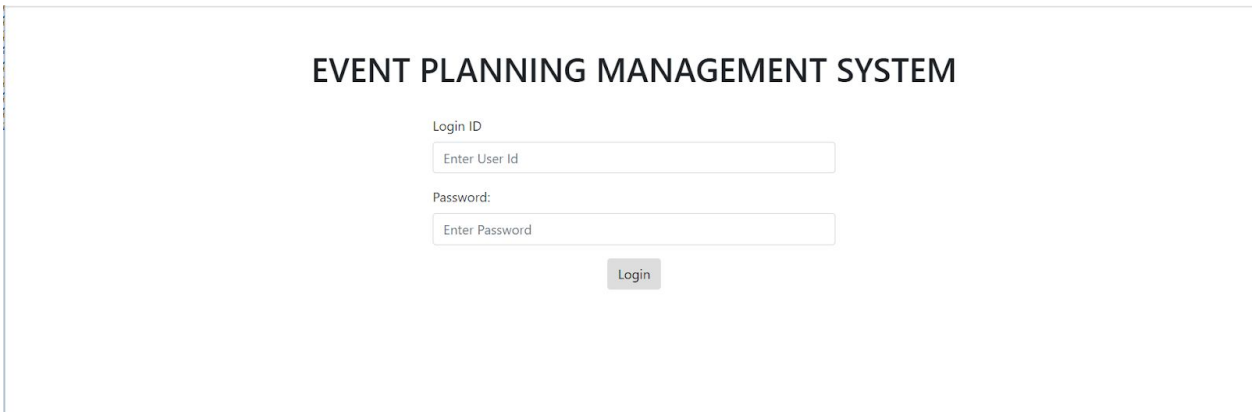
SELECT Organizer_Id, First_Name ,Last_Name ,Email, Password ,Phone_Number FROM Organizer;



Additional Screenshots:



EVENT PLANNING MANAGEMENT SYSTEM



Sprint 2

REQUIREMENTS

Story ID	Story description
US1	As an admin, I need to view registration details.
US2	As an organizer, I want to send notification.
US3	As an organizer, I want to assign accomodation for attendees.
US4	As an admin, I want to create and modify the events of the conference.
US5	As an organizer, I want to view the details of specific event.
US6	As a volunteer, I want to view list of events and contact the organizer of a specific event.
US7	As an attendee, I want to view details of registered event

CONCEPTUAL DESIGN

1) Entity: **Admin**
Attributes:
- Admin_Id
- First_Name
- Last_Name
- Password

Entity: **Registration_details**
Attributes:
- Registration_id
- Event_Id
- Attendee_Id

Relationship: **Admin** views **Registration_details**
Cardinality: One to Many
Participation:
- Admin has partial participation
- Registration_details has total participation

2) Entity: **Organizer**
Attributes:
- Organizer_Id
- First_Name
- Last_Name
- Email
- Password
- Phone Number

Entity: **Notification**
Attributes:
- notification_id
- Registration_Id

Relationship : **Organizer** sends **Notification**
Cardinality: One to Many
Participation:
Organizer has total participation
Notification has total participation

3) Entity: **Organizer**
Attributes:
Organizer_Id
First_Name
Last_Name
Email
Password
Phone Number

Entity: **Accommodation**
Attributes:
Registration_Id
Accomodation_details
Check_In_Date
Check_Out_Date
Check_In_time
Check_Out_time

Relationship : **Organizer** assigns **Accommodation**.
Cardinality: One to Many
Participation:
Organizer has total participation
Accommodation has total participation

LOGICAL DESIGN WITH NORMAL FORM IDENTIFICATION

Table: **Admin**
Columns:
- Admin_Id
- First_Name
- Last_Name
- Password

Highest normalization level: 4NF

Table: **Registration_details**
Columns:
- Event_Id [foreign key; references **Event_Id** of **Event**]
- Attendee_Id [foreign key; references **Attendee_Id** of **Attendee**]
- Registration_id

Primary Key Justification: Every attendee has Registration _details with unique Event_Id and Attendee_Id[composite key]

Highest normalization level: 4NF

Table: **Organizer**

Columns:

Organizer_Id
First_Name
Last_Name
Email
Password
Phone_Number

Primary Key Justification: Every Organizer has unique Organizer_Id

Highest normalization level: 4NF

Table: **Accomodation**

Columns:

Registration_Id[Foreign key; references **Reg_id** of **Registration_details]**
Accomodation_details
Check_In_Date
Check_Out_Date
Check_In_time
Check_Out_time

Primary key justification: Every attendee has a unique Registration_Id.

Highest normalization level: 4NF

Table: **Notification**

Columns:

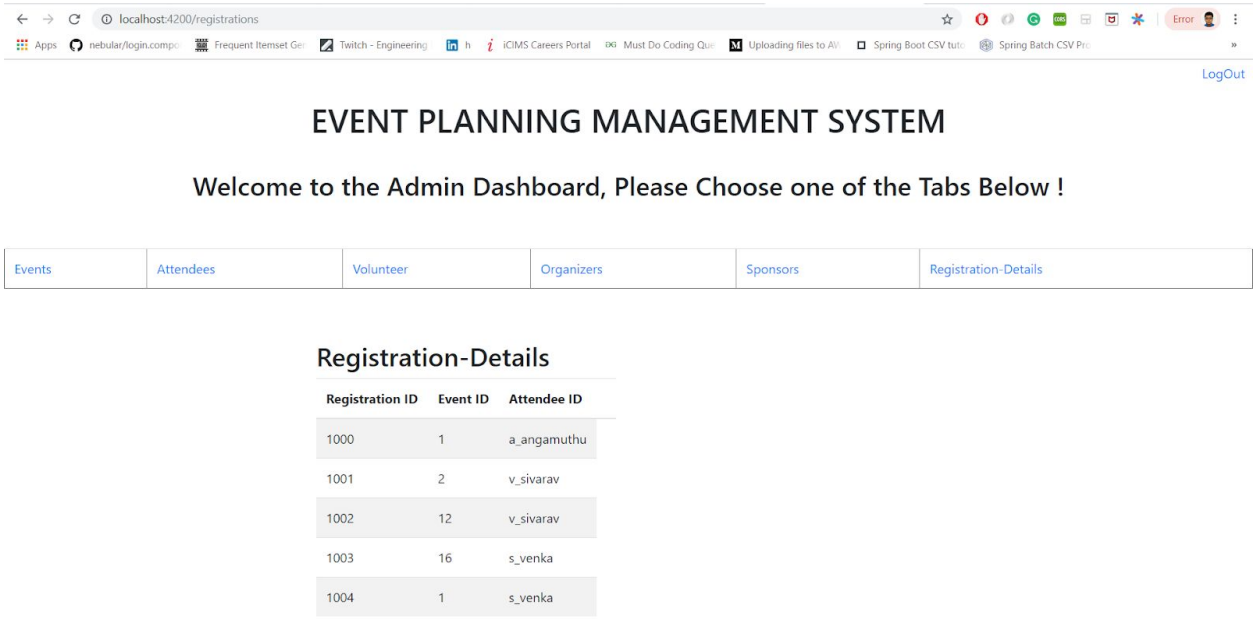
notification_id
Registration_Id[Foreign key ;references **Reg_id** of **Registration_details**]

Highest normalization level: 4NF

SQL QUERIES

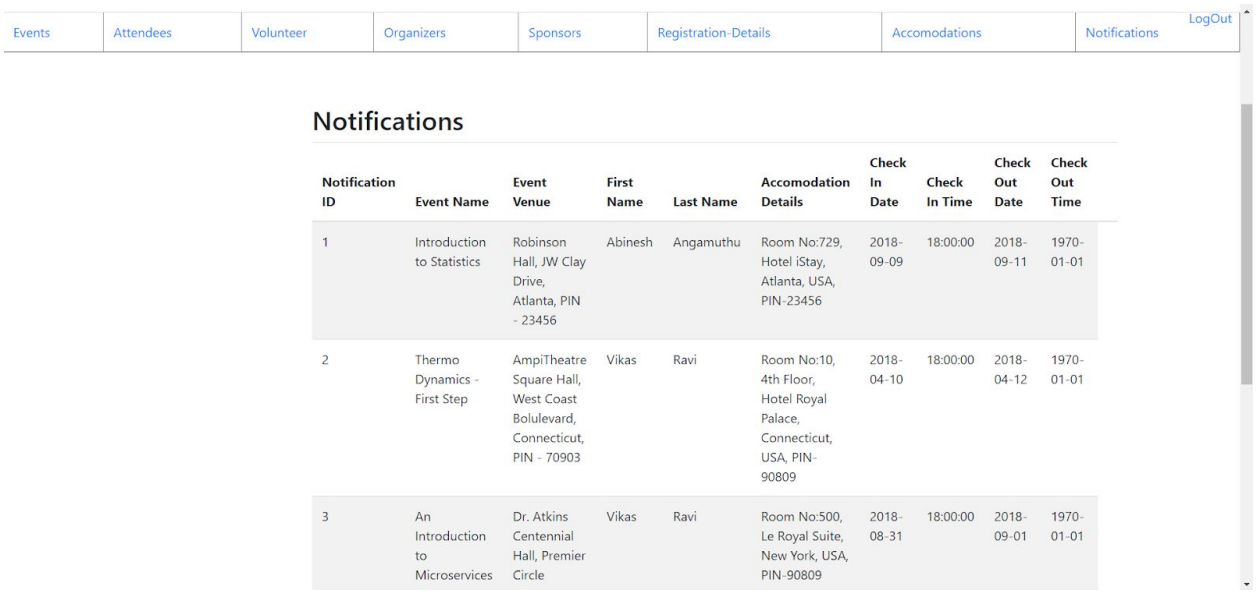
1) As an admin, I need to view registration details :

SELECT Registration_Id, Event_Id, Attendee_Id from Registration_details;



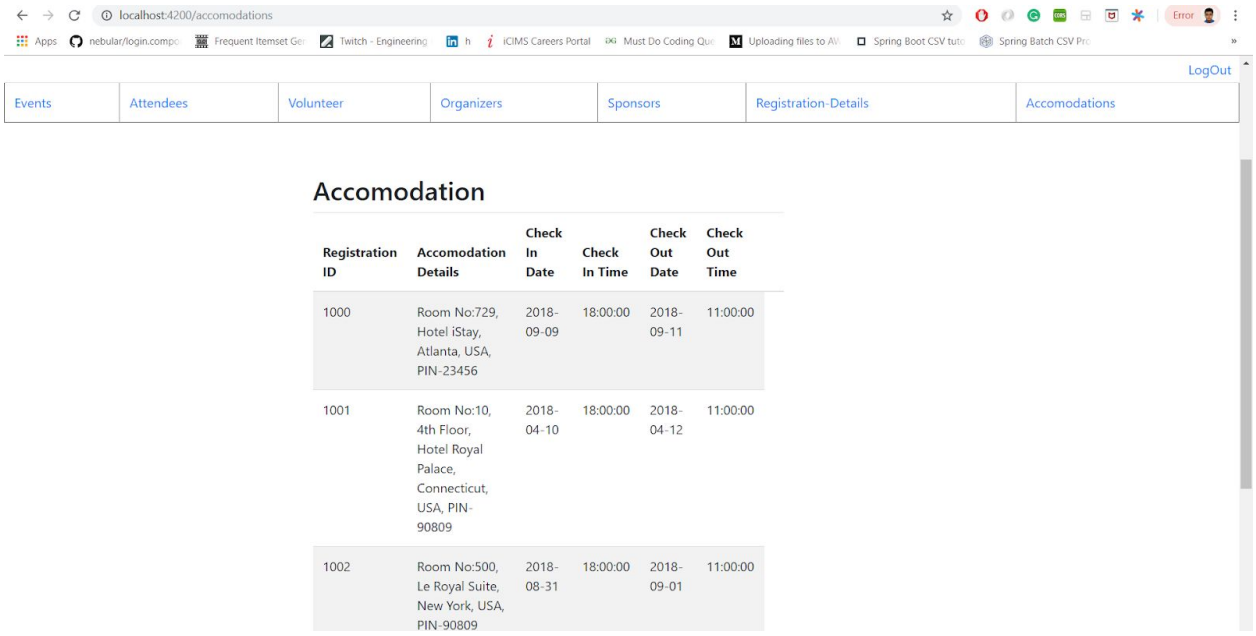
2) As an organizer, I want to send notification:

SELECT n1.notification_id, e1.Event_Name, e1.Event_Venue,b1.First_Name, b1.Last_Name, a1.Accomodation_Details, a1.Check_In_Date, a1.Check_In_Time, a1.Check_Out_Date, a1.Check_Out_Time from notification n1
INNER JOIN registration_details r1 ON n1.Registration_Id=r1.Registration_Id
INNER JOIN accommodation a1 ON a1.Registration_Id=r1.Registration_Id
INNER JOIN attendee b1 ON r1.Attendee_Id=b1.Attendee_Id
INNER JOIN event e1 ON r1.Event_id=e1.Event_id
ORDER BY notification_id asc;



3) As an organizer, I want to assign accomodation for attendees:

SELECT Registration_Id, Accomodation_Id, Check_In_Date,Check_In_Time, Check_In_Date, Check_out_time FROM Accomodation;



Sprint 3

REQUIREMENTS

Story ID	Story description
US1	Organizer assigns certificates to attendees.
US2	As an admin , I need to view list of volunteers & sponsors.
US3	As an admin, I need to view registration details.
US4	As an organizer, I want to send notification.
US5	As an organizer, I want to assign accomodation for attendees.
US6	As an admin, I want to create and modify the events of the conference.
US7	As an organizer, I want to view the details of specific event.
US8	As an attendee, I want to view details of registered event

CONCEPTUAL DESIGN

1) Entity:**Organizer**
Columns:
- Organizer_Id
- First_Name
- Last_Name
- Email
- Password
- Phone_Number

Entity: **Certificate**
Columns:
- certification_id
- Registration_Id

Relationship: **Organizer** assigns **certificate** to attendees.
Cardinality: One to Many
Participation:
- Organizer has partial participation
- Certificate has total participation

2) Entity: **Admin**
Attributes:
- Admin_Id
- First_Name
- Last_Name
- Password

Entity: **Volunteers**
Attributes**:**
- Volunteer_Id
- First_Name
- Last_Name
- Email
- Phone_number

Relationship : As an **admin**, I need to view list of **Volunteers**
Cardinality: One to Many
Participation:
Admin has partial participation
Volunteers has total participation

3) Entity: **Admin**
Attributes:
Admin_Id
First_Name
Last_Name
Password

Entity: **Sponsor**
Attributes:
Sponsor_Id
Organization_Name
Point_of_Contact
Email
Phone_Number

Relationship : As an **admin**, I need to view list of **Sponsors**.
Cardinality: One to Many
Participation:
Admin has partial participation.
Sponsors has total participation.

LOGICAL DESIGN WITH HIGHEST NORMAL FORMS AND INDEXES

Table: **Accomodation**
Columns:
Registration_Id[Foreign key; references **Reg_id** of **Registration_details]**
Accomodation_details
Check_In_Date
Check_Out_Date
Check_In_time
Check_Out_time

Highest normalization level:4NF

Indexes:
Index #1: clustered
Columns: Registration_id
Justification: Registration_id is a primary key, it identifies all columns uniquely.It is used to retrieve specific row in the table.

Index #2: Non-clustered:
Columns: CheckInDate, CheckOutDate, Accomodation_details
Justification: Using this index in the accomodation table,we are able to fetch records quickly without going round about without an index.

Table: **Registration_details**
Columns:
Event_Id [foreign key; references **Event_Id** of **Event**]
Attendee_Id [foreign key; references **Attendee_Id** of **Attendee**]
Registration_id

Primary Key Justification: Every attendee has Registration _details with unique Event_id and Attendee_id[Composite]

Highest normalization level: 4NF

Indexes:
Index #1: Clustered
Columns: Event_id, attendee_id
Justification: Upon using Event_id and attendee_id in Registration_details table, we are able to search and fetch records quickly compared to same operation without an index.

Index #2: Non-Clustered
Columns: Registration_id
Justification: Registration_id is the primary key and identifies all columns in Registration_details uniquely.It is used to retrieve specific row in the table.

Table: **Organizer**

Columns:

Organizer_Id
First_Name
Last_Name
Email
Password
Phone_Number

Primary Key Justification: Every Organizer has unique Organizer_Id

Highest normalization level: 4NF
Indexes:
Index #1: Clustered
Columns: Organizer_id
Justification:Organizer_id is the primary key and identifies all the columns uniquely.It is used to retrieve specific row in the table.

Index #2: Non-Clustered
Columns: First_Name,Last_Name,Email, Phone_Number
Justification: These indexes upon using, in Organizer table, we are able to fetch records quickly when compared to same operation without an index.

Table: **Admin**

Columns:

Admin_Id
First_Name
Last_Name
Password

Highest normalization level: 4NF
Indexes:
Index #1: Clustered
Columns: Admin_id
Justification:Admin_id is the primary key and Identifies all record uniquely. It is used to retrieve specific row in the table.

Index #2: Non-Clustered
Columns: First_Name,Last_Name
Justification:Upon using these indexes in the Admin table, we are able to fetch the records quickly,when compared to same operation without an index.

Table: **Notification**

Columns:

notification_id

Registration_Id[Foreign key ;references **Reg_id** of **Registration_details**]

Highest normalization level: 4NF

Indexes:

Index #1: Clustered

Columns: notification_id

Justification: notification_id is the primary key and Identifies all record uniquely. It is used to retrieve specific row in the table.

Index #2: Non-Clustered

Columns: Registration_id

Justification:Upon using these indexes in the Notification table, we are able to fetch the records quickly,when compared to same operation without an index.

Table**: Certificate**

Columns:

certification_id

Registration_Id[Foreign key ;references **Reg_id** of **Registration_details**]

Highest normalization level: 4NF

Indexes:

Index #1: Clustered

Columns:certification_id

Justification: cerification_id is the primary key and Identifies all record uniquely. It is used to retrieve specific row in the table.

Index #2: Non-Clustered

Columns: Registration_id

Justification:Upon using Registration_id in the Certificate table, we are able to fetch the records quickly,when compared to same operation without an index.

VIEWS AND STORED PROGRAMS

View: Notification_details
Goal: The view contains registration_id, Accomodation_details & Notification_id,event_name,event_venue,First_Name,Last_Name,CheckIn_Date,CheckIn_time,Checkout_Date,Checkout_Time.Organizer could have an easy look of accomodation and registration details , that's to be sent as notification to attendees.

View: Certification_details
Goal : The view contains certification_id, Participant Name, Event_Date,Event_Name . This enables Organizer to easily look at the attendee details, to whom, the certificate is sent.

Stored Procedure: Accomodation_details_insert
Parameters:This stored procedure contains Registration_Id, Accomodation_details,Check_In_Date,Check_In_Time,Check_Out_Time, Check_Out_date.
Goal:This procedure helps to easily insert accommodation details.This enables easy access, provides improved performance, easy fetching and modularity in accessing notification table.

Stored procedure: notification_insert
Parameters: IN notification_id, IN Registration_Id
Goal:This procedure helps to easily insert notification details. This enables easy access, provides improved performance, easy fetching and modularity in accessing notification table.

Stored procedure: registration_details_insert
Parameters: IN Registration_Id , IN Event_Id, IN Attendee_Id
Goal:This procedure helps to easily insert registration details. This enables easy access, provide improved performance, easy fetching and brings modularity in accessing registration_details table

Trigger: registration_details_after_trigger on Registration_details
Goal: The trigger allows insertion of record from Registration_details to certificate table as soon as there's a new record on registration_details simultaneously.