USE CASE STUDY REPORT

Group No: Group 1

Student Names: Jayasurya Sangeeth Someswaran & Varsha Balasubramaniam

Executive Summary:

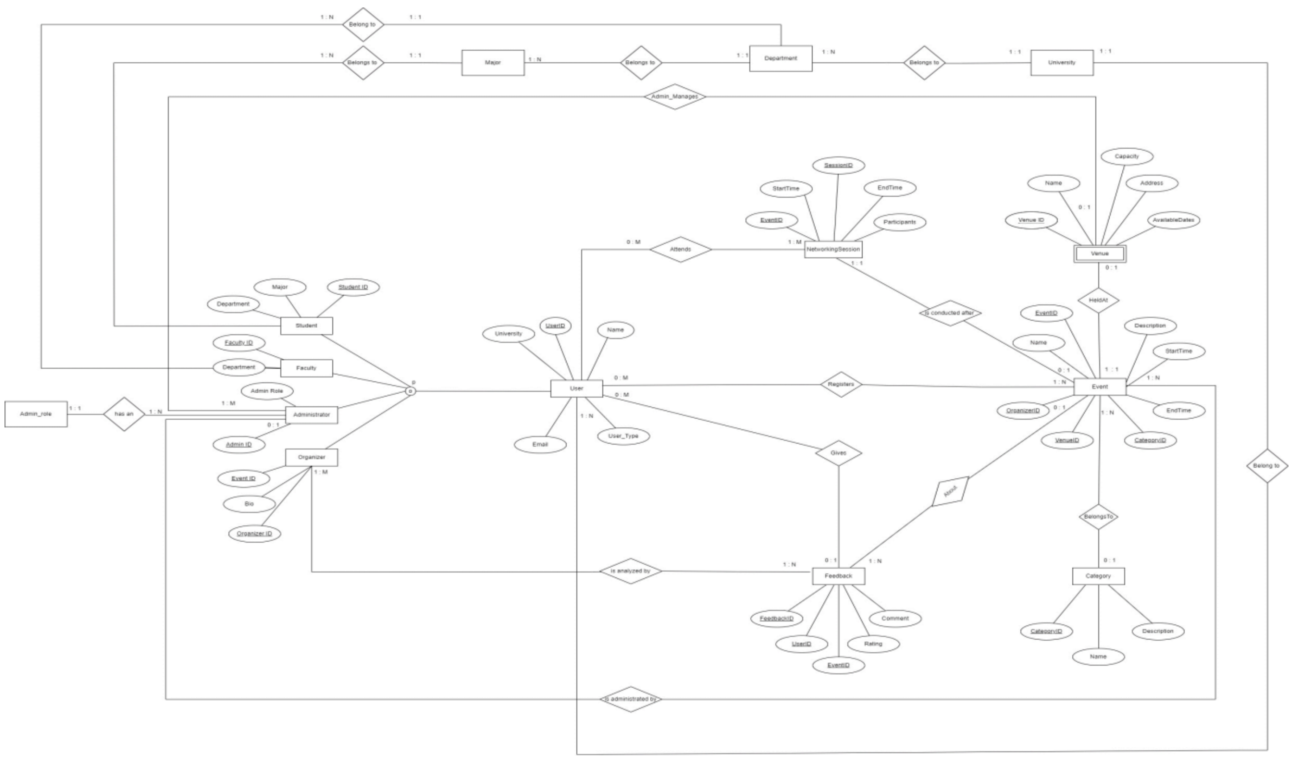
Uni-Vents presents a groundbreaking solution to the prevailing challenges in university event management. Developed by Jayasurya Sangeeth Someswaran and Varsha Balasubramaniam, our integrated events management platform aims to streamline the entire process, benefiting students, faculty, organizers, administrators, and venues alike. With a commitment to enhancing the overall educational environment, Uni-Vents addresses the issues of missed opportunities, resource wastage, reduced engagement, and administrative overhead. By centralizing event-related activities into one cohesive platform, Uni-Vents promises a transformative experience, fostering a vibrant and dynamic campus life. The platform offers a highly customizable approach, incorporating user profiles, organizer toolsets, venue management, feedback collection, event classification, and networking features. Uni-Vents is poised to revolutionize how events are organized and experienced within university communities, ultimately contributing to the holistic development, and learning experiences of students.

I. Introduction:

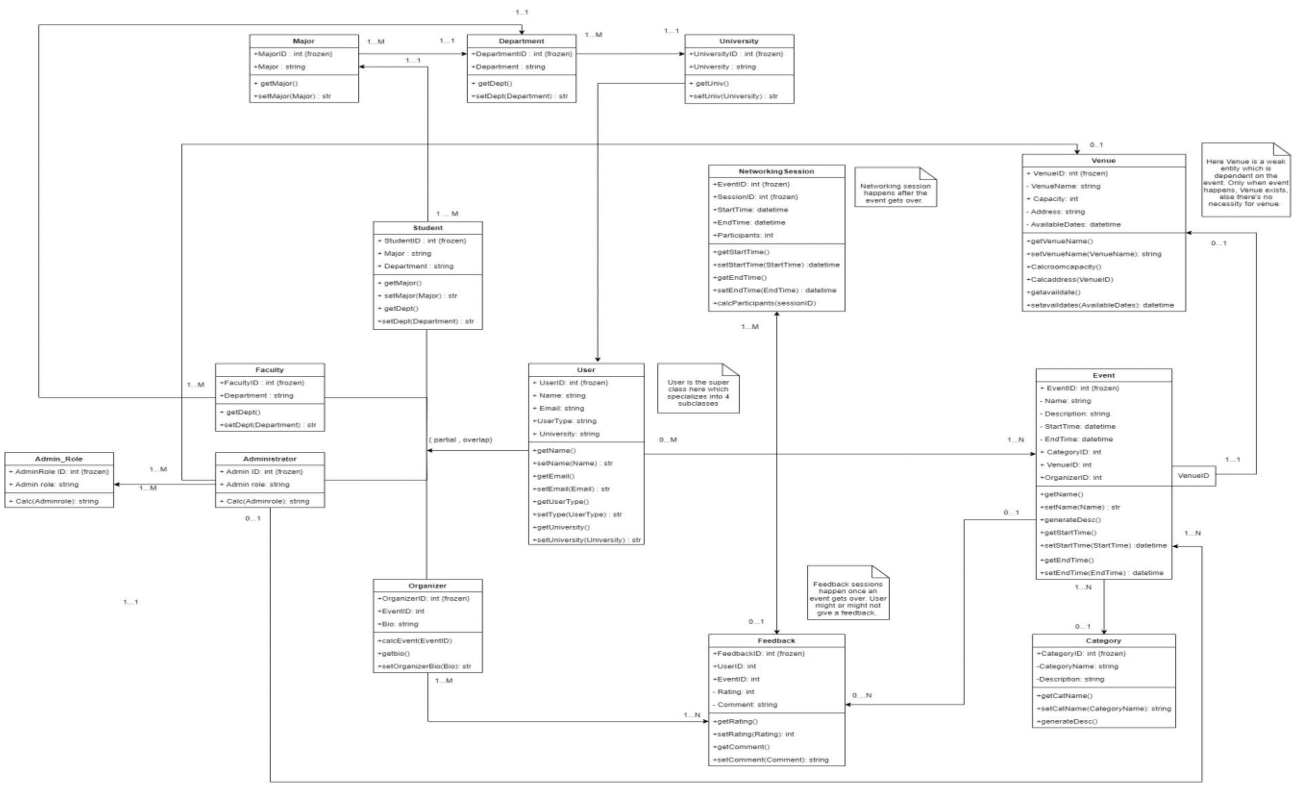
In the dynamic landscape of higher education, Uni-Vents emerges as a beacon of innovation in response to the fragmented state of event management within universities. The brainchild of Jayasurya Sangeeth Someswaran and Varsha Balasubramaniam, this platform seeks to address the multifaceted challenges stemming from the lack of a centralized system. As universities teem with diverse events, the absence of a unified platform results in missed opportunities, resource wastage, reduced engagement, and administrative overhead. Uni-Vents aims to rectify these issues by providing a comprehensive solution that integrates user profiles, efficient event management tools, robust venue capabilities, feedback collection mechanisms, event classification features, and networking opportunities. This introduction sets the stage for Uni-Vents, a revolutionary platform designed to bring about a paradigm shift in how universities approach event organization, fostering a more connected, engaged, and vibrant campus community.

II. Conceptual Data Modelling

1. EER Diagram



2. UML Diagram



III. Mapping Conceptual Model to Relational Model

Bold and underline – Primary key, Double underline – Foreign key

1) User (**UserID**, First_Name, Last_Name , University, User_type, Email)

This table captures the details of users. UserID is the primary key of the relation.

2) Student (**StudentID**, Major, Department)

Derived from User superclass where StudentID is the derived UserID for student subclass.

3) Faculty (**FacultyID**, Department)

Derived from User superclass where FacultyID is the derived UserID for the faculty subclass

4) Administrator (**AdminID**, AdminRole)

Derived from User superclass where AdminID is the derived UserID for the admin subclass

5) Organizer (**OrganizerID**, Bio)

Derived from User superclass where OrganizerID is derived UserID for organizer subclass.

6) EventCategory (**CategoryID**, Name)

This table classifies events. Each category has a unique identifier, name, and description.

7) Venue (**VenueID**, Name, University, Capacity, Address, Available_dates)

Contains details about venues where events can take place.

8) Event (**EventID**, Name, Description, Date, Start_time, End_time, CategoryID, VenueID)

CategoryID - foreign key from EventCategory – can be NULL, VenueID - foreign key from Venue - should be NOT NULL.

9) Organisedby (**OrganizerID**, EventID)

EventID is the foreign key added from the Event relation which should be NOT NULL.

10) NetworkingSession (**SessionID**, EventID, Starttime, Endtime, Participants)

EventID is the foreign key added from the Event relation which should be NOT NULL.

11) Attends (UserID, SessionID, EventID)

Primary key of the relation is the combination of all 3 foreign keys where their combination should be NOT NULL.

12) Registers (UserID, EventID)

This relation consists of 2 foreign keys UserID, EventID which together as a combination function as the primary key of this relation, which should not be NULL.

13) Feedback (**FeedbackID**, Comment, UserID, Rating, EventID)

Feedback - primary key of Feedback relation- should be NOT NULL, UserID - foreign key from User -can be NULL.

14) Analysedby (<u>OrganiserID</u>, <u>EventID</u>, <u>FeedbackID</u>, <u>UserID</u>)

This relation consists of 4 foreign keys OrganiserID, EventID, FeedbackID, UserID which together as a combination function as the primary key of this relation.

15) Admin_manage (<u>EventID</u>, <u>VenueID</u>, <u>AdminID</u>, University)

This relation consists of 3 foreign keys EventID, VenueID, AdminID which together as a combination function as the primary key of this relation.

16) Admin_role (**<u>AdminRoleID</u>**, AdminRole)

AdminroleID is the primary key.

17) Department (**<u>DepartmentID</u>**, Department)

DepartmentID is the primary key.

18) Major (**<u>MajorID</u>**, Major)

MajorID is the primary key.

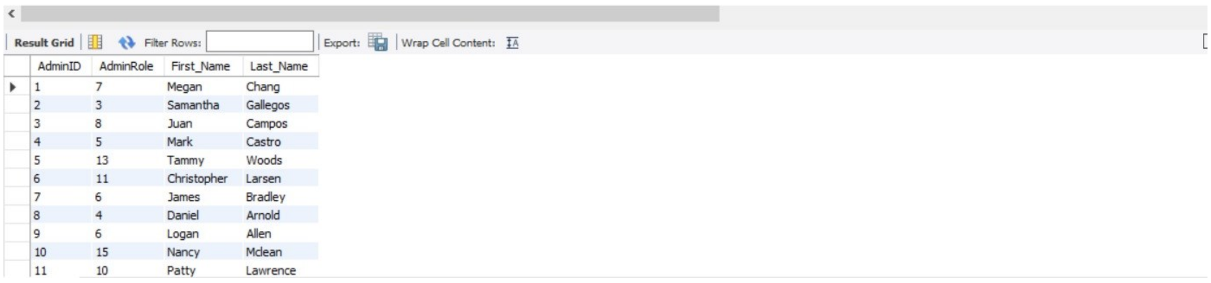
IV. Implementation of Relation Model via MySQL and NoSQL

MySQL Implementation:

The database was created in MySQL and the following queries were performed:

Query 1: List active administrators and their roles

```
SELECT a.AdminID, a.AdminRole, u.First_Name, u.Last_Name

FROM univents.administrator a

INNER JOIN univents.user u ON a.AdminID = u.UserID;
```

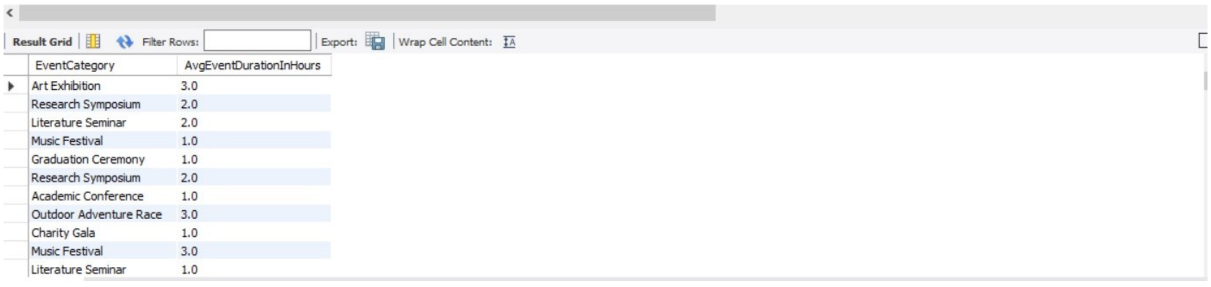


AdminID	AdminRole	First_Name	Last_Name
1	7	Megan	Chang
2	3	Samantha	Gallegos
3	8	Juan	Campos
4	5	Mark	Castro
5	13	Tammy	Woods
6	11	Christopher	Larsen
7	6	James	Bradley
8	4	Daniel	Arnold
9	6	Logan	Allen
10	15	Nancy	Mclean
11	10	Patty	Lawrence

Query 2: Average duration of events in various event category

```
SELECT

        eg.Name AS EventCategory,

        CAST(AVG(TIME_TO_SEC(TIMEDIFF(e.End_Time, e.Start_Time)) / 3600) AS
        DECIMAL(16,1)) AS AvgEventDurationInHours
```

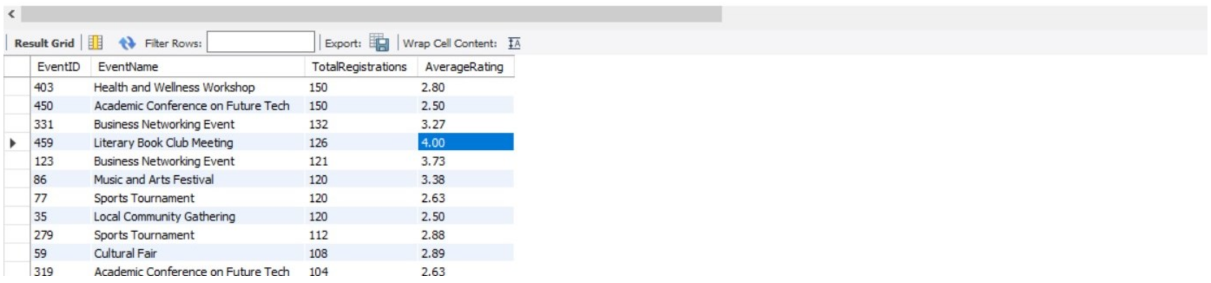
```
    FROM event e
    INNER JOIN eventcategory eg ON e.CategoryID = eg.CategoryID
    WHERE e.Start_Time IS NOT NULL AND e.End_Time IS NOT NULL
    GROUP BY EventCategory, e.EventID;
```


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EventCategory	AvgEventDurationInHours
Art Exhibition	3.0
Research Symposium	2.0
Literature Seminar	2.0
Music Festival	1.0
Graduation Ceremony	1.0
Research Symposium	2.0
Academic Conference	1.0
Outdoor Adventure Race	3.0
Charity Gala	1.0
Music Festival	3.0
Literature Seminar	1.0

Query 3: Calculate the total registrations and average rating for each event

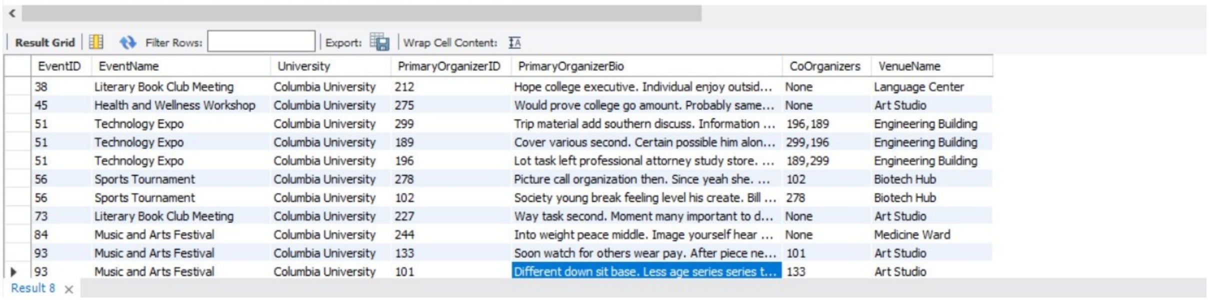
```
SELECT
        e.EventID,
         e.Name AS EventName,
        COUNT(r.UserID) AS TotalRegistrations,
        CAST( AVG(f.Rating) AS DECIMAL (16,2)) AS AverageRating
FROM event e
LEFT JOIN registers r ON e.EventID = r.EventID
LEFT JOIN feedback f ON e.EventID = f.EventID
GROUP BY e.EventID, e.Name
ORDER BY TotalRegistrations DESC;
```


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EventID	EventName	TotalRegistrations	AverageRating
403	Health and Wellness Workshop	150	2.80
450	Academic Conference on Future Tech	150	2.50
331	Business Networking Event	132	3.27
459	Literary Book Club Meeting	126	4.00
123	Business Networking Event	121	3.73
86	Music and Arts Festival	120	3.38
77	Sports Tournament	120	2.63
35	Local Community Gathering	120	2.50
279	Sports Tournament	112	2.88
59	Cultural Fair	108	2.89
319	Academic Conference on Future Tech	104	2.63

Query 4: List events along with their organizers, venues, and the earliest registration date

```
SELECT
        e.EventID,
        e.Name AS EventName,
        v.University,
        o.OrganizerID AS PrimaryOrganizerID,
        o.Bio AS PrimaryOrganizerBio,
        COALESCE(GROUP_CONCAT(co.OrganizerID), 'None') AS CoOrganizers,
        v.Name AS VenueName
FROM
   univents.event e
   INNER JOIN univents.organisedby ob ON e.EventID = ob.EventID
   INNER JOIN univents.organizer o ON ob.OrganizerID = o.OrganizerID
   LEFT JOIN univents.organisedby coob ON e.EventID = coob.EventID AND
coob.OrganizerID != o.OrganizerID
   LEFT JOIN univents.organizer co ON coob.OrganizerID = co.OrganizerID
   LEFT JOIN univents.venue v ON e.VenueID = v.VenueID
GROUP BY e.EventID, e.Name, v.University, o.OrganizerID, o.Bio, v.Name
ORDER BY University, EventID;
```



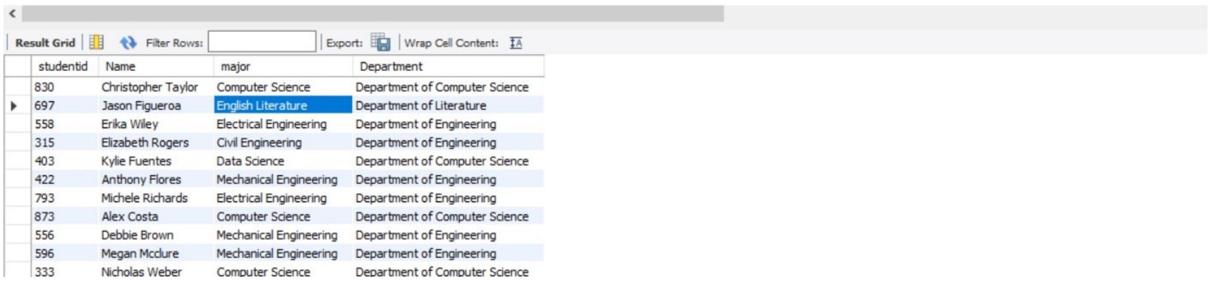
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EventID	EventName	University	PrimaryOrganizerID	PrimaryOrganizerBio	CoOrganizers	VenueName
38	Literary Book Club Meeting	Columbia University	212	Hope college executive. Individual enjoy outsid...	None	Language Center
45	Health and Wellness Workshop	Columbia University	275	Would prove college go amount. Probably same...	None	Art Studio
51	Technology Expo	Columbia University	299	Trip material add southern discuss. Information ...	196,189	Engineering Building
51	Technology Expo	Columbia University	189	Cover various second. Certain possible him alon...	299,196	Engineering Building
51	Technology Expo	Columbia University	196	Lot task left professional attorney study store. ...	189,299	Engineering Building
56	Sports Tournament	Columbia University	278	Picture call organization then. Since yeah she. ...	102	Biotech Hub
56	Sports Tournament	Columbia University	102	Society young break feeling level his create. Bill ...	278	Biotech Hub
73	Literary Book Club Meeting	Columbia University	227	Way task second. Moment many important to d...	None	Art Studio
84	Music and Arts Festival	Columbia University	244	Into weight peace middle. Image yourself hear ...	None	Medicine Ward
93	Music and Arts Festival	Columbia University	133	Soon watch for others wear pay. After piece ne...	101	Art Studio
93	Music and Arts Festival	Columbia University	101	Different down sit base. Less age series series t...	133	Art Studio

Result 8

Query 5: Students Who Organized Events at Massachusetts Institute of Technology

```
SELECT s.studentid, CONCAT(u.first_name, ' ', u.last_name) AS Name, s.major,
s.Department
FROM univents.student s
INNER JOIN univents.user u ON s.studentid = u.userid
WHERE studentid IN (
   SELECT ob.OrganizerID
   FROM univents.organisedby ob
   INNER JOIN univents.event e ON ob.EventID = e.EventID
   INNER JOIN univents.venue v ON e.VenueID = v.VenueID
   WHERE v.University = 'Massachusetts Institute of Technology');
```

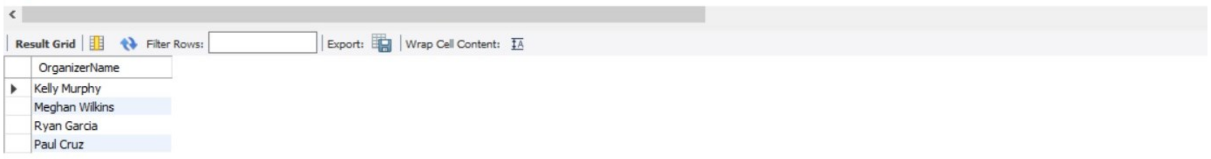


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

studentid	Name	major	Department
830	Christopher Taylor	Computer Science	Department of Computer Science
697	Jason Figueroa	English Literature	Department of Literature
558	Erika Wiley	Electrical Engineering	Department of Engineering
315	Elizabeth Rogers	Civil Engineering	Department of Engineering
403	Kylie Fuentes	Data Science	Department of Computer Science
422	Anthony Flores	Mechanical Engineering	Department of Engineering
793	Michele Richards	Electrical Engineering	Department of Engineering
873	Alex Costa	Computer Science	Department of Computer Science
556	Debbie Brown	Mechanical Engineering	Department of Engineering
596	Megan Mcclure	Mechanical Engineering	Department of Engineering
333	Nicholas Weber	Computer Science	Department of Computer Science

Query 6: List of Organizers from Northeastern University for Research Symposium Events

```
SELECT DISTINCT CONCAT(u.First_Name, ' ', u.Last_Name) AS OrganizerName

FROM univents.user u

INNER JOIN univents.organizer o ON u.UserID = o.OrganizerID

WHERE EXISTS (

    SELECT *

    FROM univents.organisedby ob

    INNER JOIN univents.event e ON ob.EventID = e.EventID

    WHERE ob.OrganizerID = u.UserID

        AND e.CategoryID IN (

            SELECT ec.CategoryID

            FROM univents.eventcategory ec

            WHERE ec.Name = 'Research Symposium' ))

AND University = 'Northeastern University'
```

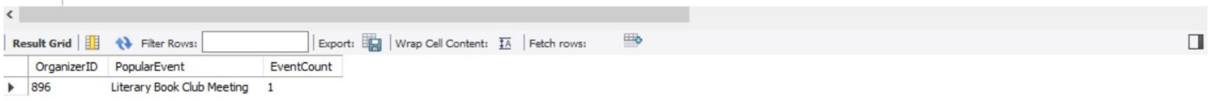


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

OrganizerName
Kelly Murphy
Meghan Wilkins
Ryan Garcia
Paul Cruz

Query 7: Retrieve the most commonly organized event by the same group across all organizers

```
SELECT

    ob.OrganizerID,

    MAX(pe.PopularEvent) AS PopularEvent,

    MAX(pe.EventCount) AS EventCount

FROM organisedby ob

JOIN (
    SELECT
        ob.OrganizerID,
        e.EventID,
        e.Name AS PopularEvent,
        COUNT(DISTINCT e.EventID) AS EventCount
    FROM event e
    INNER JOIN organisedby ob ON e.EventID = ob.EventID
    WHERE
        EXISTS (
            SELECT 1
            FROM organisedby otherOb
            WHERE otherOb.EventID = e.EventID
                AND otherOb.OrganizerID <> ob.OrganizerID)
    GROUP BY ob.OrganizerID, e.EventID, e.Name
) AS pe ON ob.OrganizerID = pe.OrganizerID
GROUP BY ob.OrganizerID
ORDER BY MAX(pe.EventCount) DESC
LIMIT 1;
```

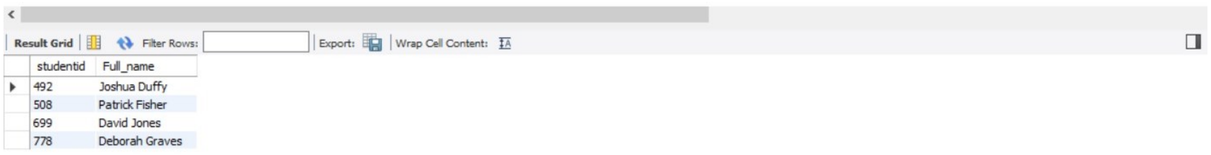


OrganizerID	PopularEvent	EventCount
896	Literary Book Club Meeting	1

Query 8: Students who have not registered for any events

```
SELECT s.studentid, CONCAT(u.First_Name, ' ', u.Last_Name) AS Full_name
FROM student s
INNER JOIN user u ON s.studentid = u.userid
WHERE u.university = 'Northeastern University'
EXCEPT
SELECT u.UserID, CONCAT(u.First_Name, ' ', u.Last_Name) AS Full_name
FROM registers r
INNER JOIN user u ON r.userid = u.userid
```

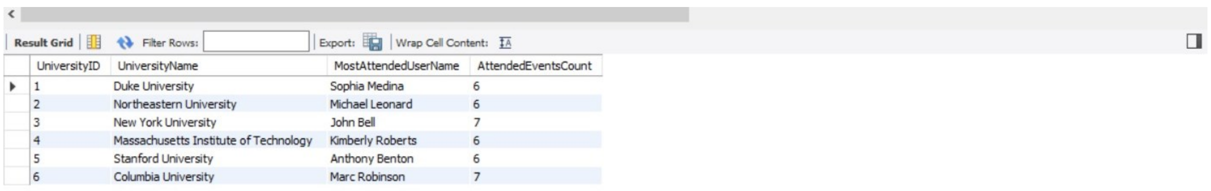
```
WHERE u.university = 'Northeastern University'
```



studentid	Full_name
492	Joshua Duffy
508	Patrick Fisher
699	David Jones
778	Deborah Graves

Query 9: Retrieve Most Attended User Details and Events Count by University

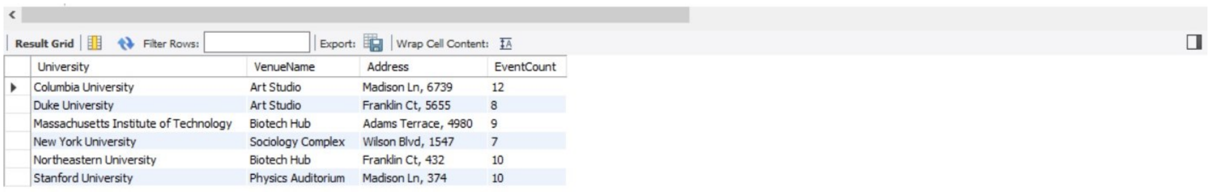
```
SELECT
    university.UniversityID,
    university.UniversityName,
    (SELECT CONCAT(user.First_Name, ' ', user.Last_Name) AS MostAttendedUserName
        FROM attends
        INNER JOIN user ON attends.UserID = user.UserID
        WHERE user.University = university.UniversityName
        GROUP BY user.UserID
        ORDER BY COUNT(DISTINCT attends.EventID) DESC
        LIMIT 1) AS MostAttendedUserName,
    (SELECT COUNT(DISTINCT attends.EventID) AS AttendedEventsCount
        FROM attends
        INNER JOIN user ON attends.UserID = user.UserID
        WHERE user.University = university.UniversityName
        GROUP BY user.UserID
        ORDER BY AttendedEventsCount DESC
        LIMIT 1 ) AS AttendedEventsCount
FROM university;
```



UniversityID	UniversityName	MostAttendedUserName	AttendedEventsCount
1	Duke University	Sophia Medina	6
2	Northeastern University	Michael Leonard	6
3	New York University	John Bell	7
4	Massachusetts Institute of Technology	Kimberly Roberts	6
5	Stanford University	Anthony Benton	6
6	Columbia University	Marc Robinson	7

Query 10: Famous venue per university

```
SELECT University, VenueName, Address, EventCount
FROM (
 SELECT
        v.University,
        v.Name AS VenueName,
        v.Address,
        COUNT(e.EventID) AS EventCount,
        ROW_NUMBER() OVER (PARTITION BY v.University ORDER BY
COUNT(e.EventID) DESC) AS RowNum
    FROM event e
    JOIN venue v ON e.VenueID = v.VenueID
    GROUP BY v.University, v.Name, v.Address) AS RankedVenues
WHERE RowNum = 1;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

University	VenueName	Address	EventCount
Columbia University	Art Studio	Madison Ln, 6739	12
Duke University	Art Studio	Franklin Ct, 5655	8
Massachusetts Institute of Technology	Biotech Hub	Adams Terrace, 4980	9
New York University	Sociology Complex	Wilson Blvd, 1547	7
Northeastern University	Biotech Hub	Franklin Ct, 432	10
Stanford University	Physics Auditorium	Madison Ln, 374	10

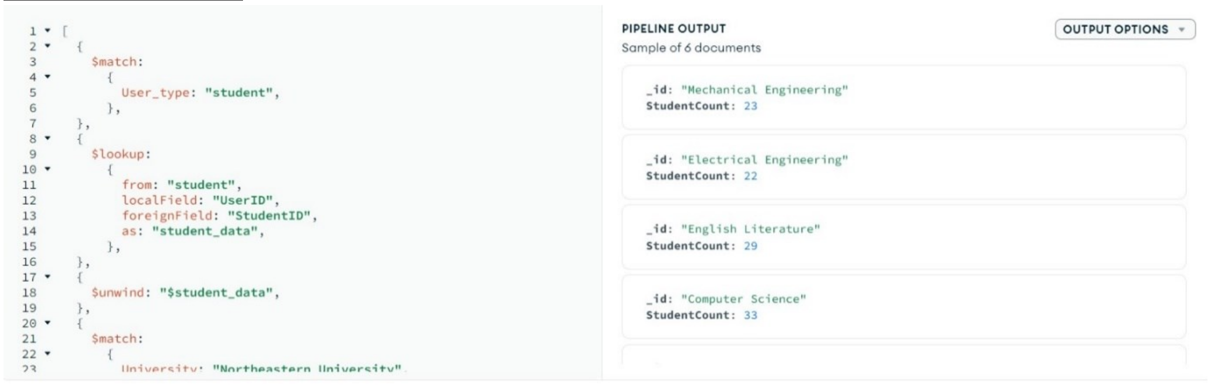
NoSQL Implementation:

The tables are created in MongoDB. The following NoSQL queries were done:

Query 1: To fetch student count in each major :

result = client['univents_nosql']['user'].aggregate([[{ $match: { User_type: "student", }, }, { $lookup:{ from: "student", localField: "UserID", foreignField: "StudentID", as: "student_data", }, }, { $unwind: "$student_data", }, { $match: { University: "Northeastern University", }, }, { $group: { _id: "$student_data.Major", StudentCount: { $sum: 1, }, }, },]

Result Screenshot:



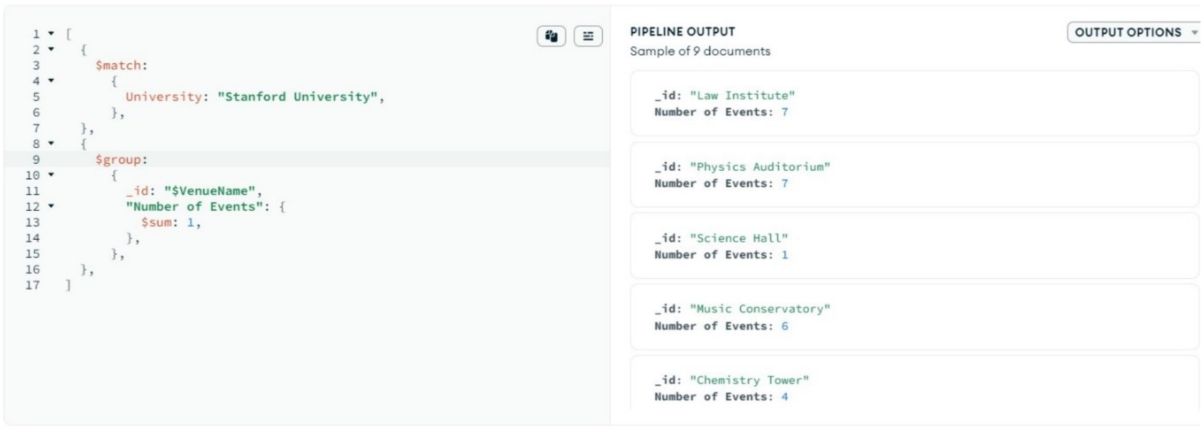
```
[
  {
    $match:
      {
        User_type: "student",
      },
  },
  {
    $lookup:
      {
        from: "student",
        localField: "UserID",
        foreignField: "StudentID",
        as: "student_data",
      },
  },
  {
    $unwind: "$student_data",
  },
  {
    $match:
      {
        University: "Northeastern University",
```

PIPELINE OUTPUT
Sample of 6 documents

```
_id: "Mechanical Engineering"
StudentCount: 23

_id: "Electrical Engineering"
StudentCount: 22

_id: "English Literature"
StudentCount: 29

_id: "Computer Science"
StudentCount: 33
```

Query 2: To find the number of events that are happening in a specific venue in a specific university:

result = client['univents_nosql']['event_venue'].aggregate([[{ $match: { University: "Stanford University", }, }, { $group: { _id: "$VenueName", "Number of Events": { $sum: 1, }, }, },] Result Screenshot:



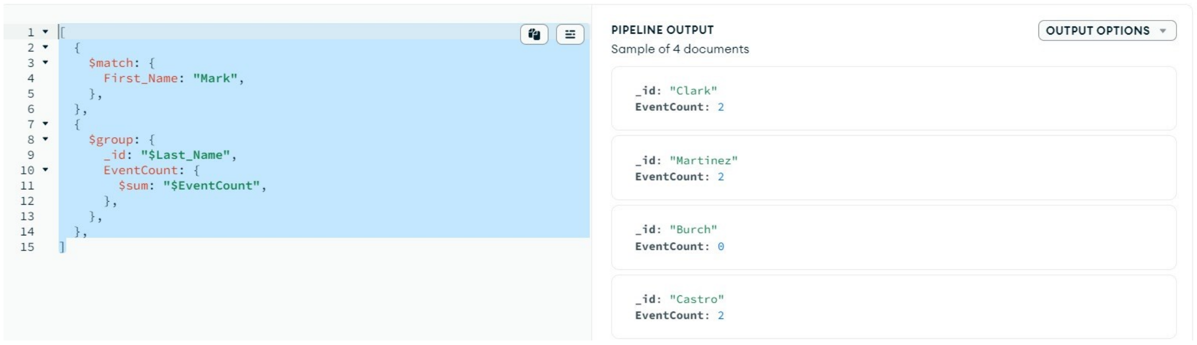
```
[
  {
    $match:
      {
        University: "Stanford University",
      },
  },
  {
    $group:
      {
        _id: "$VenueName",
        "Number of Events": {
          $sum: 1,
        },
      },
  },
]
```

```
PIPELINE OUTPUT
Sample of 9 documents

_id: "Law Institute"
Number of Events: 7

_id: "Physics Auditorium"
Number of Events: 7

_id: "Science Hall"
Number of Events: 1

_id: "Music Conservatory"
Number of Events: 6

_id: "Chemistry Tower"
Number of Events: 4
```

Query 3: To fetch number of events attended by the person with first name being "Mark"

result = client['univents_nosql']['userdetailswitheventcount'].aggregate([[{ $match: { First_Name: "Mark", }, }, { $group: { _id: "$Last_Name", EventCount: { $sum: "$EventCount", }, }, },]

Result Screenshot:



```
[
  {
    $match: {
      First_Name: "Mark",
    },
  },
  {
    $group: {
      _id: "$Last_Name",
      EventCount: {
        $sum: "$EventCount",
      },
    },
  },
]
```

```
PIPELINE OUTPUT
Sample of 4 documents

_id: "Clark"
EventCount: 2

_id: "Martinez"
EventCount: 2

_id: "Burch"
EventCount: 0

_id: "Castro"
EventCount: 2
```

V. Database Access via Python

The database is connected to Python using MySQL connector and pymysql followed by a cursor to execute the queries and fetch records using execute. The list and matplotlib are used to store data and show it in a pie chart and bar chart as follows.

1. Python Code to plot a Bar graph between the event count from each university:

import pymysql import pandas AS pd import matplotlib.pyplot AS plt import seaborn AS sns host = 'localhost' user = 'root' password = 'Sribala22*' database = 'univents' connection = pymysql.connect(host=host,

user=user,

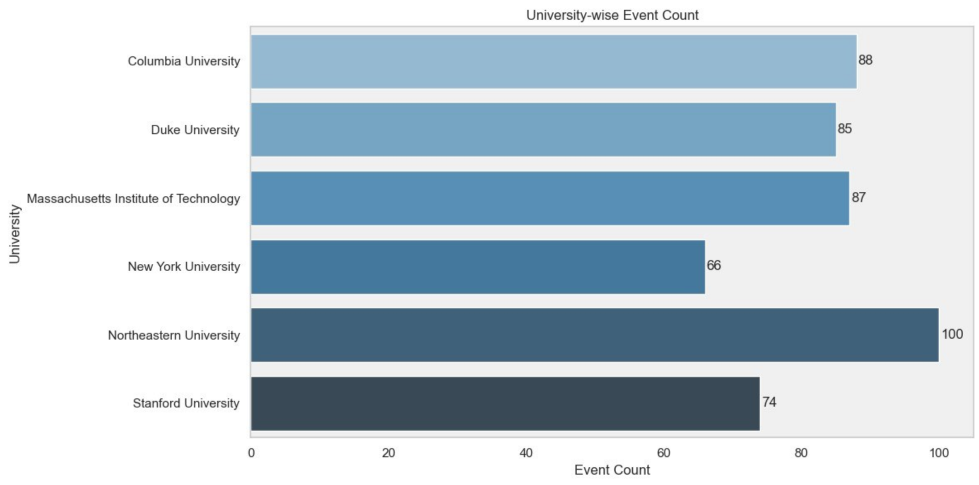
password=password,

database=database) cursor = connection.cursor() query = """SELECT v.University,

COUNT(DISTINCT e.EventID) AS EventCount

```
FROM event e

JOIN venue v

   ON e.VenueID = v.VenueID

GROUP BY  v.University; """ cursor.execute(query) result = cursor.fetchall()
university_events = pd.DataFrame(result, columns=['University', 'EventCount']) colors =
sns.color_palette("Blues_d", n_colors=len(university_events)) # Darker Blue
plt.figure(figsize=(12, 6)) plt.rcParams['axes.facecolor'] = '#f0f0f0' # Light Grey ax =
sns.barplot(x='EventCount', y='University', data=university_events, palette=colors) for p IN
ax.patches: ax.annotate(f'{p.get_width():.0f}', (p.get_width() + 0.2, p.get_y() + p.get_height()
/ 2), ha='left', va='center') plt.title('University-wise Event Count') plt.xlabel('Event Count')
plt.ylabel('University') plt.grid(False) plt.tight_layout() plt.show() cursor.close()
connection.close()
```

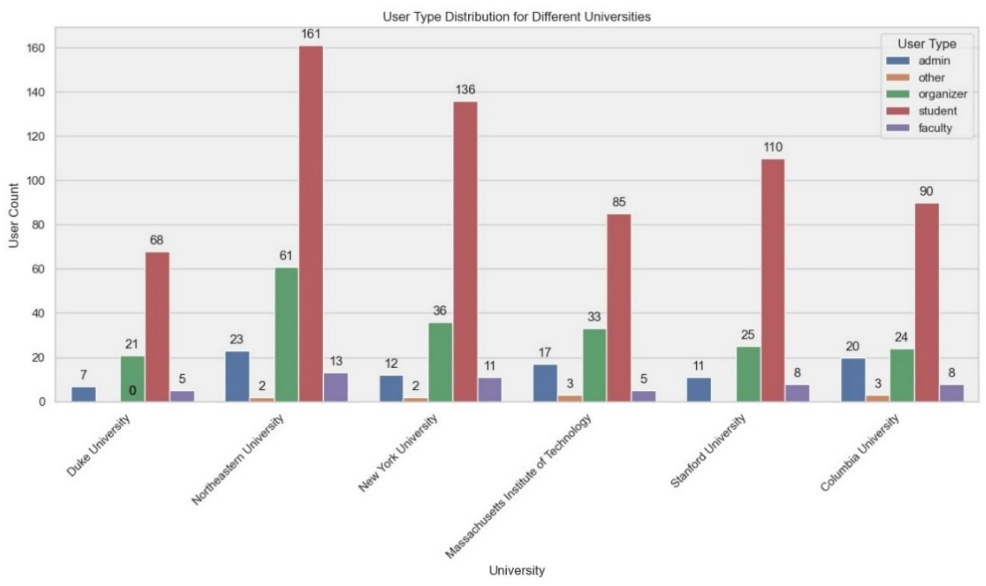
Screenshot of the result:



2. Python Code to find user type distribution in different universities:

```
import pymysql import pandas AS pd import matplotlib.pyplot AS plt import seaborn AS sns
host = 'localhost' user = 'root' password = 'Sribala22*' database = 'univents' connection =
pymysql.connect(host=host,

        user=user,

        password=password,

        database=database) cursor = connection.cursor() query = """SELECT University,

        User_Type,

        COUNT(UserID) AS UserCount

FROM userdetailswitheventcount

GROUP BY  University, User_Type; """ cursor.execute(query) result = cursor.fetchall()
user_type_distribution = pd.DataFrame(result, columns=['University', 'User_Type',
'UserCount']) plt.figure(figsize=(16, 8)) ax = sns.barplot(x='University', y='UserCount',
hue='User_Type', data=user_type_distribution) plt.title('User Type Distribution for Different
Universities') plt.xlabel('University') plt.ylabel('User Count') plt.legend(title='User Type') for
p IN ax.patches: ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2.,
p.get_height()), ha='center', va='center', xytext=(0, 10), textcoords='offset points')
plt.xticks(rotation=45, ha='right') plt.show() cursor.close() connection.close()
```

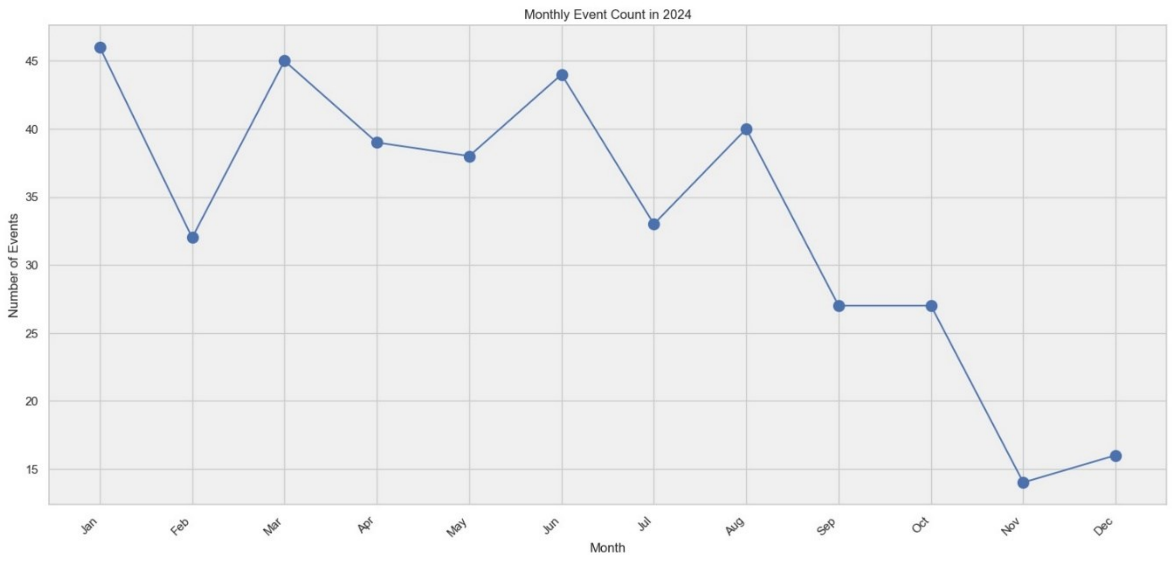
Screenshot of the result:



3. Python Code to plot a line chart for monthly event count in the year 2024

```
import pymysql import pandas AS pd import matplotlib.pyplot AS plt import calendar host =
'localhost' user = 'root' password = 'Sribala22*' database = 'univents' connection =
pymysql.connect(host=host,

        user=user,

        password=password,

        database=database) sql_query = """SELECT DATE_FORMAT(e.Date,

        '%Y-%m') AS Month, COUNT(e.EventID) AS EventCount

FROM event e

WHERE YEAR(e.Date) = 2024

GROUP BY  Month

ORDER BY  Month; """ df = pd.read_sql(sql_query, connection) df['Month'] =
pd.to_datetime(df['Month']) df['Month'] = df['Month'].dt.month.map(lambda x:
calendar.month_abbr[x]) plt.figure(figsize=(12, 6)) plt.plot(df['Month'], df['EventCount'],
marker='o', markersize=10, linestyle='-', color='b') plt.title('Monthly Event Count in 2024')
plt.xlabel('Month') plt.ylabel('Number of Events') plt.xticks(rotation=45, ha='right') # Rotate
x-axis labels for better readability plt.grid(True) plt.tight_layout() plt.show()
connection.close()
```

Screenshot of the result:

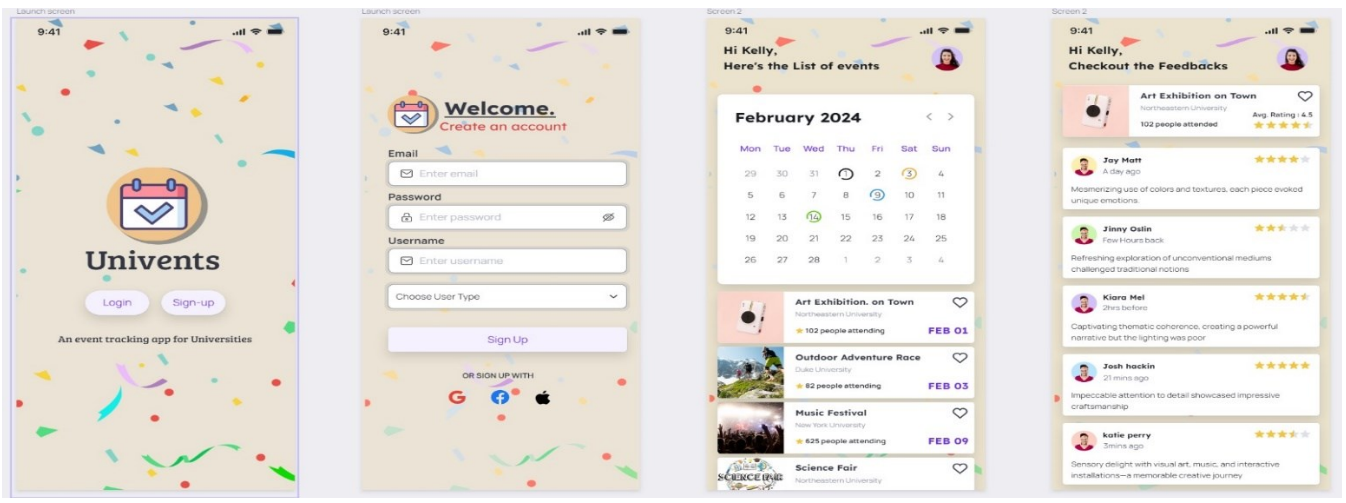


VI. Summary and Recommendation

Summary:

Uni-Vents, spearheaded by Jayasurya Sangeeth Someswaran and Varsha Balasubramaniam, offers a transformative solution to the challenges plaguing university event management. By centralizing event-related activities into an integrated platform, Uni-Vents addresses issues such as missed opportunities, resource wastage, reduced engagement, and administrative overhead. The platform's comprehensive features, including user profiles, organizer toolsets, venue management, feedback collection, event classification, and networking opportunities, promise to revolutionize how events are organized and experienced within university communities. Uni-Vents stands as a beacon of innovation, poised to enhance the holistic development and learning experiences of students while fostering a vibrant campus life.

Recommendation:



Mesmerizing use of colors and textures, each piece evoked unique emotions.

Refreshing exploration of unconventional mediums challenged traditional notions

Captivating thematic coherence, creating a powerful narrative but the lighting was poor

Impeccable attention to detail showcased impressive craftsmanship

Sensory delight with visual art, music, and interactive installations—a memorable creative journey

Considering the comprehensive features and benefits offered by Uni-Vents, it is highly recommended for universities and colleges to adopt this platform to streamline their event management processes. The integrated nature of Uni-Vents addresses critical pain points faced by students, organizers, and administrators, ensuring a more efficient, engaging, and error-free event planning and execution. The customizable approach and diverse toolsets cater to the unique needs of each stakeholder, promising a user-friendly experience. By investing in Uni-Vents, institutions can not only optimize resource utilization and enhance event attendance but also foster a sense of community and connection among students. The platform's potential to revolutionize the university event landscape makes it a valuable asset for institutions seeking to create a more dynamic and enriching campus experience.