**SVKM’s NMIMS**

**School of Technology Management & Engineering (Indore Campus)**

**Computer Engineering Department (B Tech/MBATech CE and B Tech AIDS Sem IV)**

**Database Management System**

**Project Report**

|  |  |  |
| --- | --- | --- |
| Program | BTech CE | |
| Semester | IV | |
| Name of the Project: | Urban Solve : community solution platform | |
|  | | |
| Details of Project Members |  |  |
| Batch :A1 | Roll No. D075 | Name: Sahil Makhija |
|  | Roll No. D027 | Name: Dhruv Goyal |
|  | Roll No. D030 | Name: Garv Gupta |
|  |  |  |
| Date of Submission: 16/24/25 | | |

**Github link of your project:**

**Note:**

1. Create a readme file if you have multiple files
2. All files must be properly named (Example:R004\_DBMSProject)
3. Submit all relevant files of your work ( Report, all SQL files, Any other files)
4. **Plagiarism is highly discouraged (Your report will be checked for plagiarism)**

**Rubrics for the Project evaluation:**

|  |  |
| --- | --- |
| First phase of evaluation:  Innovative Ideas (5 Marks)  Design and Partial implementation (5 Marks) | 10 marks |
| Final phase of evaluation  Implementation, presentation and viva, Self-Learning and Learning Beyond classroom | 10 marks |

**Project Report**

**Urban Solve :Community Solutions Platform**

**By**

**Sahil Makhija, Roll number: D075**

**Dhruv Goyal, Roll number: D027**

**Garv Gupta, Roll number: D030**

**Course: DBMS**

**AY: 2024-25**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Topic** | **Page no.** |
| **1** | Storyline |  |
| **2** | Components of Database Design |  |
| **3** | Entity Relationship Diagram |  |
| **4** | Relational Model |  |
| **5** | Normalization |  |
| **6** | SQL Queries |  |
| **7** | Learning from the Project |  |
| **8** | Project Demonstration |  |
| **9** | Self-learning beyond classroom |  |
| **10** | Learning from the project |  |
| **8** | Challenges faced |  |
| **9** | Conclusion |  |

1. **Storyline:**

Urban Solve is a web-based civic-tech platform inspired by apps like 311. It allows citizens to report problems in their locality, such as garbage accumulation, broken streetlights, water contamination, etc.

Users can upload images, provide a description, and add the location of the problem. These reports are stored in a MySQL database and accessed by admins who review and assign the issue to appropriate volunteer teams.  
The system ensures transparent issue tracking and quicker problem resolution using digital means.

1. **Components of Database Design:**

### A screenshot of a computer AI-generated content may be incorrect.🔹 ****Entities and Their Attributes:****

🔹 **Relationships, Cardinality, and Participation:**

**A black screen with white text

AI-generated content may be incorrect.**

🔹 **Summary of Keys and Foreign Keys:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**III. Entity Relationship Diagram:**

A diagram of a software company

AI-generated content may be incorrect.

**IV. Relational Model**

**1. User:**

A screenshot of a computer

AI-generated content may be incorrect. Stores information about registered users who report problems.

**2. Report:**

Stores user-submitted reports about community issues.

A screenshot of a computer

AI-generated content may be incorrect.

**3. Admin:**

A screenshot of a black screen

AI-generated content may be incorrect. Stores admin data who assign tasks to volunteers.

**4. Volunteer\_Team:**

Stores information about volunteer teams that resolve the issues.

A screenshot of a black screen

AI-generated content may be incorrect.

**5. Assignment:**

Stores assignments made by admins to volunteer teams.

A screenshot of a computer

AI-generated content may be incorrect.

🔗 **Key Relationships:**

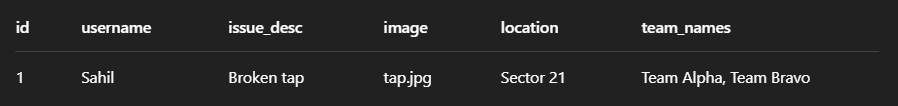
A screenshot of a black screen

AI-generated content may be incorrect.

**V. Normalization:**

**🚧 Unnormalized Form (UNF)**

Example of how data might originally look (with repeating groups):



**1NF – Atomic Values**

**A screenshot of a computer

AI-generated content may be incorrect.**

Repeating group team\_names is split into multiple rows.

**2NF – Full Functional Dependency**

break out repeating or partially dependent data:

A black background with white text

AI-generated content may be incorrect.

A screenshot of a black screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black background with white text

AI-generated content may be incorrect.

A screenshot of a black screen

AI-generated content may be incorrect.

✅ Every non-key attribute is fully functionally dependent on the whole key in each table.

**3NF – Remove Transitive Dependencies**

Already satisfied:

* users: no attribute depends on anything other than user\_id
* reports: all attributes depend directly on report\_id, not transitively
* No derived data like admin\_name in assignments

**VI. SQL Queries**

Get all reports:

SELECT \* FROM reports;

Get all users:

SELECT \* FROM users;

**Get reports with 'Pending' status**

SELECT \* FROM reports WHERE status = 'Pending';

Get reports in a specific location:

SELECT \* FROM reports WHERE location LIKE '%Sector 21%';

**List all reports submitted in April 2024:**

SELECT \* FROM reports WHERE MONTH(created\_at) = 4 AND YEAR(created\_at) = 2024;

Count number of reports per category:

SELECT category, COUNT(\*) AS total\_reports

FROM reports

GROUP BY category;

Join reports with users:

SELECT r.report\_id, r.title, u.fullname

FROM reports r

JOIN users u ON r.user\_id = u.user\_id;

List users who submitted more than 3 reports:

SELECT u.fullname, COUNT(r.report\_id) AS total

FROM users u

JOIN reports r ON u.user\_id = r.user\_id

GROUP BY u.user\_id

HAVING COUNT(r.report\_id) > 3;

Reports ordered by most recent:

SELECT \* FROM reports ORDER BY created\_at DESC;

Search reports that mention 'water':

SELECT \* FROM reports WHERE description LIKE '%water%';

**Get reports assigned to a vendor**

SELECT \* FROM reports WHERE vendor IS NOT NULL;

Reports that haven't been assigned yet:

SELECT \* FROM reports WHERE vendor IS NULL OR status = 'Pending';

**Get top 5 users with most reports**

SELECT u.fullname, COUNT(r.report\_id) AS total\_reports

FROM users u

JOIN reports r ON u.user\_id = r.user\_id

GROUP BY u.user\_id

ORDER BY total\_reports DESC

LIMIT 5;

**Update status of a report to 'Resolved'**

UPDATE reports SET status = 'Resolved' WHERE report\_id = 5;

Assign a vendor to a report:

UPDATE reports SET vendor = 'ABC Solutions', status = 'Assigned' WHERE report\_id = 8;

Delete a test report:

DELETE FROM reports WHERE title = 'Test Report';

Show users who never submitted a report:

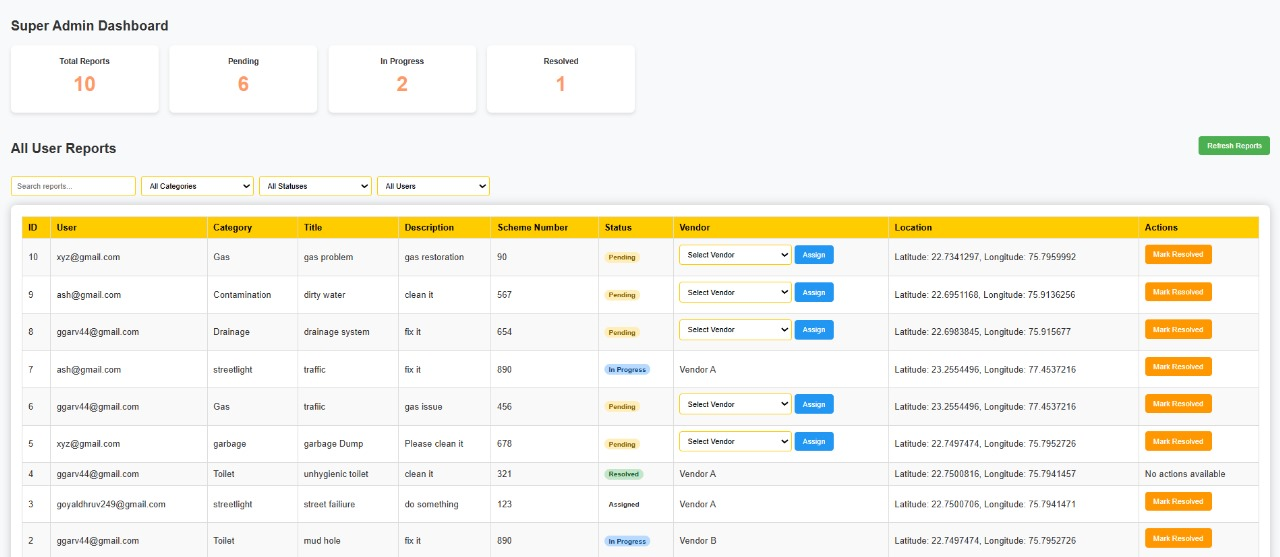
SELECT \* FROM users

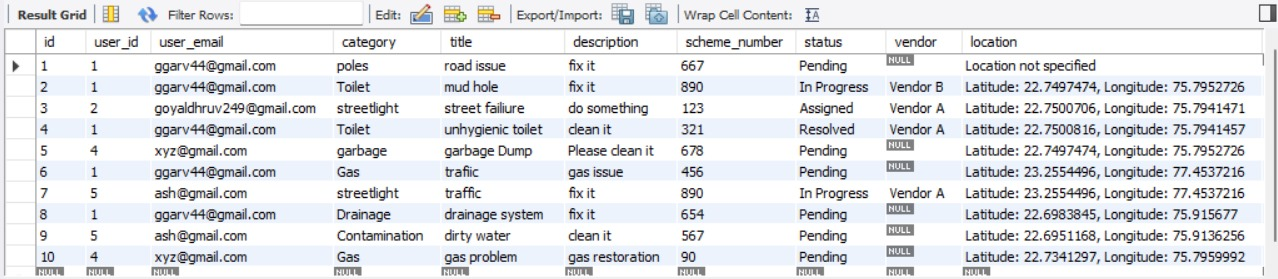
WHERE user\_id NOT IN (SELECT DISTINCT user\_id FROM reports);

**VI. Project demonstration**

**✅ Tools / Software / Libraries Used:**

| **Category** | **Tool / Library** | **Purpose** |
| --- | --- | --- |
| **Frontend** | HTML, CSS, JavaScript | Building user-facing forms and pages for report submission |
| **Backend** | Node.js + Express.js | Server-side logic and handling API requests |
| **Database** | MySQL (via mysql2 package) | Storing users, reports, and related metadata |
| **Session Management** | express-session | For login session handling |
| **Password Security** | bcrypt | For encrypting user passwords |
| **File Handling** | Multer (if used) | For image upload functionality |
| **Testing & Debugging** | Postman | To test API endpoints like login, report submission, etc. |
| **Description of the Demonstration of the Project:**  **User Interface:**   * **A simple report submission form built using HTML/CSS.** * **Fields include: title, category, description, location, image, etc.** * **On form submission, data is sent to the server using AJAX/Fetch API.**   **Report Workflow Demonstration:**   1. **User registers or logs in using email and password. ↪ The credentials are securely stored with hashed passwords in the users table.** 2. **User submits a report through the form: ↪ The form data is stored in the reports table with fields like category, location, lat/lng, and image path.** 3. **Admin Panel (backend or via Postman): ↪ Admin views all reports using /api/admin/reports endpoint. ↪ Admin can assign vendors or mark status as "Assigned" or "Resolved".** 4. **All changes are stored and updated in real-time in the MySQL database.** |  |  |



****

**VII. Self -Learning beyond classroom**:

This project helped bridge the gap between theoretical knowledge and practical application of full-stack development integrated with DBMS. By researching documentation, Stack Overflow discussions, and official library guides, I learned to implement real-world backend solutions and improve problem-solving skills beyond the syllabus.

**VIII. Learning from the Project**

The Urban Solve project helped me understand how to design and normalize a real-world database. I learned to write efficient SQL queries, connect a MySQL database with a Node.js backend, and manage user sessions securely.  
It gave me hands-on experience in full-stack development and showed how DBMS concepts are applied in practical civic-tech applications.  
Overall, it improved my problem-solving, backend logic, and database integration skills.

**IX. Challenges Faced :**

**1. Backend and Database Integration**

* Connecting Node.js with MySQL and managing async query responses required troubleshooting unfamiliar errors.
* Ensuring secure data insertion and retrieval through parameterized queries was a new challenge.

**2. Handling Image Uploads**

* Managing file uploads and storing image paths in the database involved learning how to use middleware like Multer.

**3. User Authentication and Session Control**

* Implementing proper login sessions and protecting admin-only routes required careful logic and testing.

**4. Designing a Scalable Schema**

* Structuring tables to avoid redundancy while maintaining flexibility for future features (like vendor management or admin dashboards) required multiple iterations.

**5. Debugging SQL Queries**

* Writing JOINs, subqueries, and grouped queries for reporting features sometimes led to logical or syntax errors that took time to debug.

**X. Conclusion**

The Urban Solve project successfully demonstrated how database systems play a critical role in solving real-world problems. By integrating MySQL with a Node.js backend and designing a clean, normalized database schema, I was able to create a functional and scalable community issue reporting system.

This project helped me apply classroom knowledge to a real-life use case—right from user registration to issue tracking and assignment. It improved my understanding of relational models, SQL queries, backend integration, and overall system design.

In conclusion, this project not only strengthened my DBMS skills but also gave me the confidence to work on full-stack applications with real-world impact.