International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Dynamic Webpage Management System

Joel A[1]*,Dr. M.Jaithoon Bibi* [2]

[1] B.Sc Computer Science With Cognitive Systems, Sri Ramakrishna College of Arts & Science ,Coimbatore
[2] Assistant Professor Department of Computer Science with Cognitive Systems (B.Sc.CsCs) Sri Ramakrishna College of Arts & Science ,Coimbatore

ABSTRACT :

This report describes the design, development, and integration of a dynamic self-managed webpage management system based on the concept of Google Forms deployed on a Linux based web server. The objective of this system is to allow users to build, distribute and manage generic forms including business forms, quiz forms while ensuring that data is securely and swiftly managed through the robust MySQL database. Django frameworks and Apache2 for server deployment have been utilized, thereby providing a practical use case methodology for hosting of dynamic web applications in a Linux server.

Keywords: Dynamic Webpage, Django, MySQL, Linux Hosting, Apache2, Custom Form Builder , Ubuntu Server

INTRODUCTION :

This project focuses on developing and hosting a *self-managed form submission system* on an *Ubuntu Linux server*, providing an independent, customizable, and secure alternative to third-party form services. The objective is to create a *web-based form application* that allows users to submit data, which is stored in a *structured database* while enabling *real-time monitoring and administrative control* By implementing this project, individuals and organizations can *host their own form system* without relying on cloud-based solutions, ensuring *complete control over data privacy, security, and customization*. This self-hosted approach is particularly beneficial for businesses, educational institutions, healthcare organizations, and government agencies that require *secure and private* data collection methods. Additionally, it provides *scalability*, as the server infrastructure can be expanded based on the volume of form submissions and user interactions.

Ultimately, this project offers a *cost-effective, secure, and highly customizable* web-based form system that serves as a viable alternative to commercial form services. It empowers users with full *data ownership, enhanced security, and complete flexibility*, making it an ideal solution for handling sensitive or large-scale data collection needs. Platforms like Google Forms, Type form, and Microsoft Forms provide user-friendly solutions for form creation and submission. However, these cloud-based services come with limitations, including restricted customization, lack of full data ownership, and concerns regarding security and privacy. Organizations that handle sensitive or large-scale user data often prefer self-hosted solutions to maintain control over data storage, security configurations, and user access.

This project aims to develop and host a *self-managed online form submission system* on an *Ubuntu Linux server*, providing an alternative to cloud-based solutions like Google Forms. The system enables users to create and submit forms, with responses securely stored in a *structured database* while ensuring data privacy, security, and real-time monitoring. The *frontend* is built using *HTML, CSS, JavaScript, or React*, offering a user-friendly interface for form submissions. The *backend* is developed using *PHP or Python (Flask/Django)* to handle form processing and data storage in *MySQL/PostgreSQL*. The server runs on *Apache or Nginx*, with *SSL encryption* configured for secure communication. A *domain name* is linked to the server, making the form system accessible via a web address. Security measures such as *firewall protection, user activity tracking, and data validation* ensure the system i**s** *safe and efficient*. This project provides a *cost-effective, scalable, and customizable* solution for organizations and individuals who need *full control over their data*, making it ideal for businesses, educational institutions, and private users handling sensitive information.

EXISTING SYSTEM :

Currently, most users rely on cloud-based form submission platforms such as *Google Forms, Type form, Microsoft Forms, and JotForm*. These platforms offer a *ready-to-use* interface for data collection without requiring technical expertise. Users can create forms, share them with respondents, and collect responses in a structured manner. These services also provide basic analytics and integration with third-party tools.

1. **Limited Customization** – Predefined form templates and limited design flexibility restrict user-specific requirements.
2. **Data Privacy Concerns** – Data is stored on third-party cloud servers, which can pose security risks and compliance issues.
3. **No Full Data Ownership** – Users do not have complete control over collected data; service providers may have access to stored responses.
4. **Internet Dependency** – Requires a constant internet connection to access and manage form submissions.
5. **Subscription Costs** – Many advanced features (such as conditional logic, analytics, or integrations) require paid plans.
6. **Limited Scalability** – Handling a high volume of responses or integrating with external databases is often restricted by pricing models.

PROPOSED SYSTEM :

To overcome these limitations, this project proposes a *self-hosted form submission system* on an *Ubuntu Linux server***,** allowing organizations and individuals to create, manage, and store form submissions independently. The system will use *PHP or Python (Flask/Django) for backend development, MySQL/PostgreSQL for database management, and Apache/Nginx for web hosting***,** ensuring a *fully customizable and scalable* platform.

- **Complete Data Ownership** – Since the system is self-hosted, users have full control over data storage, security, and access.
- **Enhanced Security** – Implementation of SSL encryption, firewall protection, and data validation ensures a secure platform.
- **Customization & Flexibility** – The system allows full customization of form structure, UI design, and response handling.
- **No Subscription Fees** – Eliminates the need for paid plans, making it a cost-effective solution for long-term use.
- **Scalability** – Can be optimized to handle a large number of responses efficiently with database and server optimizations.
- **Integration with Other Systems** – Supports integration with institutional databases, business applications, and analytics tools.

OBJECTIVE :

1. **Educational Institutions** – Universities and colleges can use it for *student registration, examination forms, feedback collection, and research surveys***.**
2. **Healthcare Sector** – Hospitals and clinics can use it for *patient surveys, appointment booking, and confidential medical data collection***.**
3. **Corporate Organizations –** Businesses can implement it for *employee feedback, customer surveys, internal audits, and project tracking***.**
4. **Event Management –** Organizers can use it for *conference registrations, ticketing, and attendee feedback collection*.
5. **Banking & Finance –** Can be used for *loan applications, customer satisfaction surveys, and employee compliance reporting***.**

METHODOLOGY OF THE PROJECT :

The development of the self-hosted form submission system follows a structured methodology to ensure efficiency, scalability, and security. The process is divided into several phases, each focusing on specific aspects of the project.

1.Requirement Analysis

In this phase, the limitations of existing form submission platforms, such as Google Forms, are identified. The functional and non-functional requirements of the system are defined to ensure it meets user needs. The technology stack is carefully selected, including the frontend (HTML, CSS, JavaScript/React.js), backend (PHP/Flask/Django), database (MySQL/PostgreSQL), and server (Apache/Nginx). Additionally, user roles like Admin and User are determined, along with their respective permissions to ensure proper access control.

2.System Design

This phase focuses on designing the system architecture using a client-server model. The database schema is created to efficiently store form responses, ensuring data is organized and easily retrievable. UI/UX wireframes are developed to provide an interactive and user-friendly experience, making the system intuitive for all users.

3.Deployment & Implementation

Once the design is complete, the system is deployed on a Linux server using Apache or Nginx. Server settings are optimized to enhance performance and security. System logs are continuously monitored to detect and resolve any errors or vulnerabilities, ensuring the system runs smoothly and securely.

4.Future Enhancement

The methodology also includes plans for future improvements. These enhancements may involve implementing AI-based analytics to gain insights from the collected data, developing a mobile app for better accessibility, enabling multi-language support to cater to a global audience, and introducing blockchain-based security to ensure tamper-proof data storage. These future upgrades aim to keep the system scalable, secure, and adaptable to evolving user needs.

FUTURE ENHANCEMENT :

The *self-hosted form submission system* has a strong foundation, but several enhancements can be implemented to improve *functionality, security, and user experience* in the future. Some key areas for improvement include:

1. AI-Powered Data Analysis

- Implement *machine learning algorithms* to analyse form responses and detect patterns.
- Provide *predictive insights* based on user-submitted data.

2. Advanced Authentication and Security

- Introduce *OAuth, two-factor authentication (2FA), or biometric authentication* for secure user login.
- Implement *role-based access control (RBAC)* for different user permissions.

3. Mobile App Integration

- Develop an *Android and iOS app* for easier access and submission on mobile devices.
- Enable *offline submission* where users can fill forms without an internet connection and sync data later.

4. Multi-Language Support

- Add a *language translation feature* to make the system accessible to a global audience.
- Provide *customizable UI themes* for improved user experience.

5. **Cloud and Hybrid Deployment**

- Allow hybrid hosting, where organizations can choose between *self-hosted and cloud-based deployment*.
- Integrate with cloud storage options like *AWS S3, Google Drive, or Dropbox* for secure backups.

6. Integration with External APIs

- Provide *API support* for businesses and institutions to integrate with *CRM, ERP, and analytics tools*.
- Enable *webhooks* for real-time data synchronization with external applications.

7. Blockchain-Based Data Security

- Implement *blockchain technology* for an immutable and transparent audit trail of submissions.
- Ensure *tamper-proof data storage* to enhance trust in sensitive data collection.

SYSTEM TESTING AND IMPLEMENTATION :

System Testing:

System testing ensures that the form submission system operates smoothly across different environments. The testing process begins with **Unit Testing**, where individual modules such as form creation, submission, and database storage are tested. This phase focuses on verifying form validation rules, such as required fields and correct data types, to ensure they function as intended. After unit testing, **Integration Testing** is conducted to verify the interactions between the frontend, backend, and database. This ensures that submitted data is correctly processed and stored without any discrepancies. Following this, **Functional Testing** is performed to check if all features, such as login, form submission, and data retrieval, work as expected.

This phase also validates form responses and ensures proper error handling to provide a smooth user experience. **Security Testing** is then carried out to identify vulnerabilities such as SQL injection, cross-site scripting (XSS), and unauthorized access attempts. SSL encryption is also tested to ensure secure data transmission. Next, **Performance Testing** measures the system's response time and its ability to handle server load, especially when simulating multiple users submitting forms simultaneously. Finally, **User Acceptance Testing (UAT)** is conducted, where users and admins validate the system to ensure it meets their expectations and identify any usability improvements before deployment. This comprehensive testing process ensures the system is reliable, secure, and ready for real-world use.

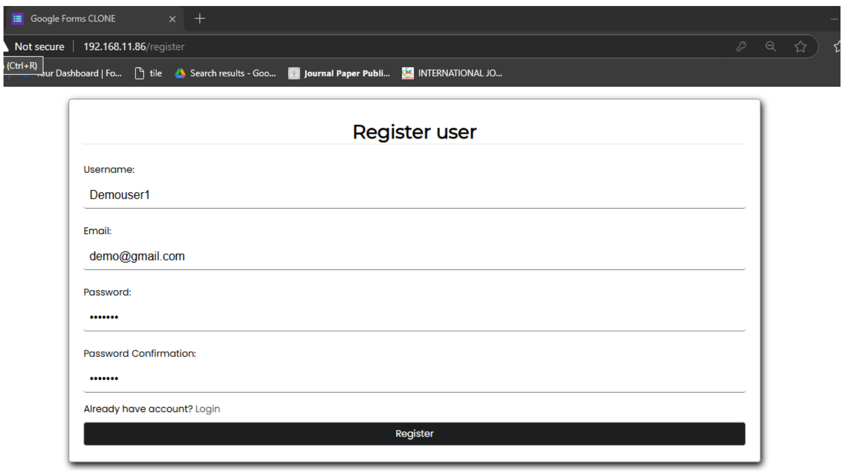
System Implementation :

The implementation process involves a series of carefully planned steps to ensure the system is properly set up, secure, and ready for use. It begins with **Server Setup**, where Ubuntu Linux is installed as the operating system, and Apache or Nginx is configured as the web server. MySQL or PostgreSQL is set up as the database, and SSL encryption (using Let's Encrypt) is configured to secure data transmission. Once the server is ready, **Database Deployment** takes place, where the required tables for storing form responses are created. Backup strategies are also implemented to prevent data loss and ensure data integrity.

The next step is **Code Deployment**, where backend scripts (PHP/Python) and frontend files are uploaded to the server. Version control using Git is set up to manage code changes efficiently and track updates. After the code is deployed, **Domain Configuration** is carried out, where a custom domain name is assigned to the system for easy access. DNS records and web server settings are configured to ensure the domain points to the correct server. Following this, **Testing and Debugging** is performed to conduct final tests, resolve any remaining issues, and optimize the system for fast response times and improved performance. Finally, **User Training and Documentation** is provided, where user manuals and admin guides are created to help users and administrators understand the system. Training sessions are also conducted to ensure administrators can effectively manage and maintain the system. This structured implementation process ensures the system is secure, functional, and ready for real-world use

WORK FLOW OF THE PROJECT :

Fig 1 :User Registration



Google Forms CLONE

Not secure | 192.168.11.86/register

Register user

Username: Demouser1

Email: demo@gmail.com

Password: •••••••

Password Confirmation: •••••••

Already have account? Login

Register

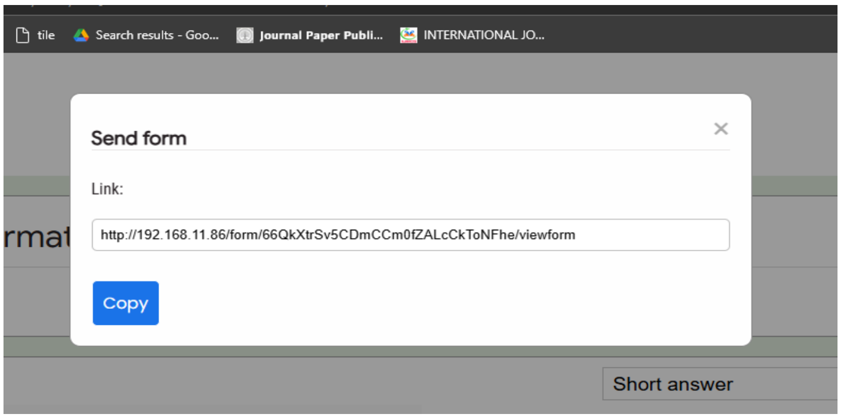
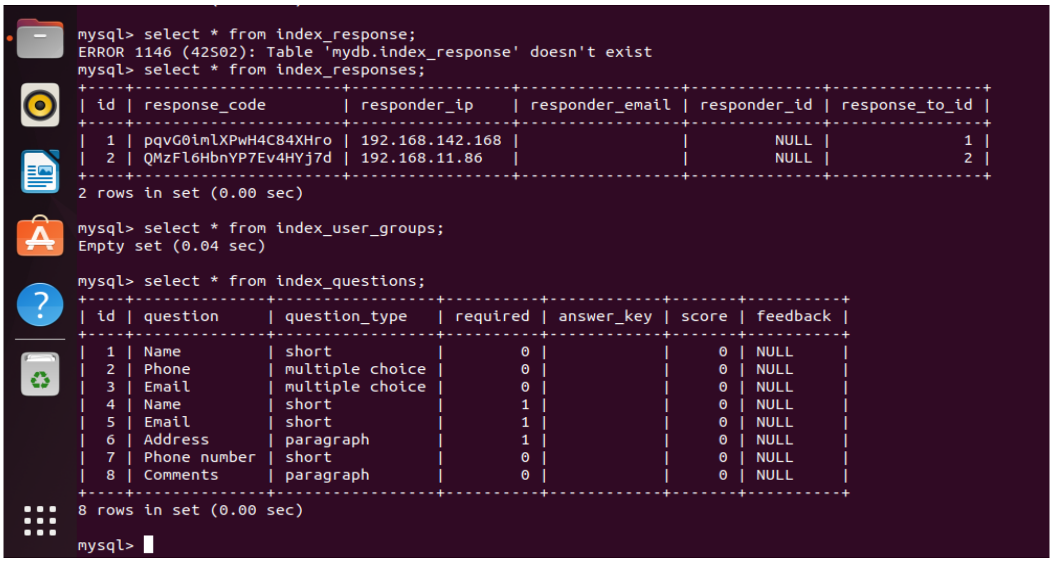


Fig: 1.2 Url To Share Forms



```
mysql> select * from index_response;
ERROR 1146 (42S02): Table 'mydb.index_response' doesn't exist
mysql> select * from index_responses;
+----+----------------------+-----------------+-----------------+--------------+----------------+
| id | response_code        | responder_ip    | responder_email | responder_id | response_to_id |
+----+----------------------+-----------------+-----------------+--------------+----------------+
|  1 | pqvG0imlXPwH4C84XHro | 192.168.142.168 |                 |         NULL |              1 |
|  2 | QMzFl6HbnYP7Ev4HYj7d | 192.168.11.86   |                 |         NULL |              2 |
+----+----------------------+-----------------+-----------------+--------------+----------------+
2 rows in set (0.00 sec)

mysql> select * from index_user_groups;
Empty set (0.04 sec)

mysql> select * from index_questions;
+----+--------------+-----------------+----------+------------+-------+----------+
| id | question     | question_type   | required | answer_key | score | feedback |
+----+--------------+-----------------+----------+------------+-------+----------+
|  1 | Name         | short           |        0 |            |     0 | NULL     |
|  2 | Phone        | multiple choice |        0 |            |     0 | NULL     |
|  3 | Email        | multiple choice |        0 |            |     0 | NULL     |
|  4 | Name         | short           |        1 |            |     0 | NULL     |
|  5 | Email        | short           |        1 |            |     0 | NULL     |
|  6 | Address      | paragraph       |        1 |            |     0 | NULL     |
|  7 | Phone number | short           |        0 |            |     0 | NULL     |
|  8 | Comments     | paragraph       |        0 |            |     0 | NULL     |
+----+--------------+-----------------+----------+------------+-------+----------+
8 rows in set (0.00 sec)

mysql> 
```

Fig: 1.3 Form Response Stores In Databses

CONCLUSION :

This project successfully delivers a *self-hosted form submission system* that provides a *secure, customizable, and scalable* alternative to cloud-based solutions. By implementing this system, organizations can maintain *full control over data*, ensuring *privacy, security, and flexibility* in form creation and submission management. The system is ideal for *educational institutions, healthcare organizations, businesses, government agencies, and research sectors*, offering a *cost-effective, independent, and efficient* solution for collecting and managing form responses. Future enhancements can include *AI-based data analysis, advanced authentication (OAuth, biometrics), and mobile app integration*, making the system even more robust and versatile

REFERENCES :

1. docs.djangoproject.com/en/3.2/ref/databases/#mysql-notes
2. studygyaan.com/Django/how-to-setup-django-applications-with-apache-and-mod-wsgi-on-ubuntu
3. pypi.org/project/mysqlclient/
4. https://www.w3schools.com
5. https://docs.djangoproject.com
6. https://www.php.net/manual/en/
7. https://dev.mysql.com/doc/
8. https://httpd.apache.org/docs/
9. https://nginx.org/en/docs/
10. https://letsencrypt.org/