**Three-Tier Architecture Report**

**1. Architecture Overview**

The three-tier architecture consists of three layers:

1. **Presentation Layer (Frontend):** Handles user interactions via a web interface.
2. **Application Layer (Backend):** Processes logic and communicates with the database.
3. **Database Layer:** Stores and retrieves data using MySQL.

This architecture separates concerns, improving scalability, security, and maintainability.

**2. Implementation Steps**

**Step 1: Setting Up MySQL Database**

* Created a database notes3\_db using phpMyAdmin.
* Defined a t3 table with id, title, and content fields.

**Step 2: Developing the Flask Backend**

* Installed Flask and MySQL connector (pip install flask mysql-connector-python).
* Created a Flask application to handle routes for adding, viewing, and deleting notes.
* Established a connection to the MySQL database.

**Step 3: Building the Frontend**

* Created an index.html file to display notes and a form to add new notes.
* Used basic CSS for styling.
* Integrated Flask with HTML templates using Jinja2.

**Step 4: Running the Application**

* Started the Flask server (python app.py).
* Accessed the application via html
* Tested CRUD operations (Create, Read, Update, Delete) via the web interface.

**3. Advantages and Challenges**

**Advantages:**

**Scalability:** Each layer can be scaled independently.

**Security:** The database is isolated from direct user access.

**Maintainability:** Code is modular, making it easier to update.

**Challenges:**

**Increased Complexity:** Requires more setup and configuration.

**Performance Overhead:** Communication between layers may introduce latency.

**Deployment Considerations:** Requires proper management of backend and database services.

**4. Comparison with Other Architectures**

|  |  |  |  |
| --- | --- | --- | --- |
| **Architecture** | **Description** | **Advantages** | **Disadvantages** |
|  |  |  |  |
| **Single-Tier** | Direct interaction between app and database. | Simple & fast | Poor scalability & security |
| **Single-Tier** | Direct interaction between app and database. | Simple & fast | Poor scalability & security |
| **Two-Tier** | Client interacts with database via backend. | Better security & flexibility | Limited scalability |
| **Three-Tier** | Separate frontend, backend, and database layers. | High security & scalability | Complex setup |

**5. Conclusion**

The **three-tier architecture** enhances scalability, security, and maintainability, making it ideal for enterprise applications. While the initial setup may be complex, the long-term benefits outweigh the challenges.