**Two-Tier Architecture Report**

**Architecture Overview**

The **Two-Tier Architecture** consists of two layers:

1. **Client Layer (Application Layer)** – The front-end or business logic layer that interacts directly with the database. In this case, a Python script (**Twotier.py**) handles user input and database operations.
2. **Database Layer** – A MySQL database (**notes\_db**) stores and manages structured data.

This architecture allows the application to execute database queries directly without an intermediary server or service.

**Implementation Steps**

1. **Database Setup:**
   * The **notes\_db.sql** script creates a MySQL database (notes\_db) with a notes table containing id, title, and content fields.
   * Uses MySQL with InnoDB storage engine for transaction support.
2. **Python Script (Twotier.py) Development:**
   * Establishes a connection to the MySQL database using mysql.connector.
   * Implements functions such as add\_note(title, content) to insert data into the notes table.
   * Executes SQL queries directly through Python without an additional middleware layer.
3. **Execution and Testing:**
   * Run Twotier.py to add and retrieve notes.
   * Use MySQL tools (phpMyAdmin or MySQL Workbench) to verify data insertion.

**Advantages and Challenges**

**Advantages:**

* **Simple and Fast:** Direct communication between application and database minimizes latency.
* **Easy to Implement:** Requires minimal setup, making it ideal for small-scale applications.
* **Better Performance (for small applications):** No need for an intermediary, reducing processing overhead.

**Challenges:**

* **Scalability Issues:** As the number of users increases, direct database access can become a bottleneck.
* **Security Risks:** Direct connections to the database expose it to potential threats.
* **Limited Flexibility:** No dedicated middleware for business logic, making enhancements harder.

**Comparison with Other Architectures**

| **Feature** | **Single-Tier Architecture** | **Two-Tier Architecture** | **Three-Tier Architecture** |
| --- | --- | --- | --- |
| **Scalability** | Very Low | Low | High |
| **Performance** | High (local only) | High (small scale) | Moderate |
| **Security** | Very Low | Low | Higher |
| **Complexity** | Very Low | Low | Medium |
| **Use Case** | Personal applications | Small applications | Enterprise systems |

In contrast to **Single-Tier Architecture** (where all functions reside on a single system) and **Three-Tier Architecture** (which includes a middleware layer for business logic), **Two-Tier Architecture** is ideal for small-scale applications where direct database access is sufficient.

**Conclusion**

The **Two-Tier Architecture** is suitable for lightweight applications with limited users. While it offers simplicity and speed, it lacks the scalability and security needed for larger applications. If the system grows, transitioning to a **Three-Tier Architecture** would be beneficial.