**What software architecture are you using? e.g. Layered/How many layers? Repository? Something else?**

I used a layered architecture with 4 layers.

Presentation Layer: Handles user interface components and user interactions.

Business Logic Layer: Contains core functionality and processing logic, managing user requests, transaction processing, and decision-making.

Data Access Layer: Manages database operations, making sure data is stored and retrieved as needed.

Database Layer: Stores data and is managed by a DBMS like MySQL.

**Please answer the following: discuss the pros/cons of the environments: VMs, docker, plain old computer. Things to consider: development vs production, working in teams, the OS you need, cost, licensing etc...**

VM pros

* Provides a complete operating system instance, ensuring full isolation and security.
* Allows multiple OS to run on one server.
* Good for production due to its isolation.

VM cons

* Takes more resources.
* Slower than containers due to hardware emulation
* Higher costs and more complex licensing.

Docker pros

* Shares the host OS so it is more resource efficient than a VM.
* Easier to scale and deploy with microservices architectures.
* Good for development due to its ease to set up and destroy.
* Easy to work with teams

Docker cons

* Less isolated compared to a VM. Having a breach in one container can potentially compromise other containers on the same host.
* Depends on the host OS compatibility.

Plain old computer pros

* Full control over hardware and software configurations.
* No overhead created from visualizations.

Plain old computer cons

* Expensive maintenance costs
* Inflexible. Not as agile as a VM or a container.