Candidate Software Architectures for Beverage Bookers

First, software architectures are how a project like our Beverage Bookers idea is implemented into an actual software product. This allows programmers to understand how a program should be built and how the interface should be implemented to interact throughout all the different components in the software.

Types of architectures we can use for Beverage Bookers:

The Client / Server Architecture:

This architecture is a system where there are 2 separate classes called the client and the server. The client is usually a user interface that the user would send a request from to the server. The server can be any service that processes that request from the client and responses back with appropriate action through the interface which it controls. With this in mind, a device or service can be both a client who sends a request to a server device or service and also be a server which processes and responds to the client request.

Advantages:

- The server has all the files and allocates resources making the client lightweight.
- Server doesn't use resources heavily unless it is processing a request from the user.
- Client remains unaffected if the server requires maintenance or upgrading.

Disadvantages:

- The server can become overloaded if too many clients are trying to request actions.
- If the server has a fatal error it will leave client requests unaccomplished.

The Three-Layer Architecture:

Three-Layer architecture is similar to how client / server architecture functions but specifically this architecture has 3 layers instead of 2 classes interacting. The system works by having a request climb through the layers. Mostly all requests interact with all layers but some specific interactions might be able to skip one of the layers. The layers are; the view layer (top), the domain layer (middle) and the data layer (bottom). The view layer is the user interface where the user can input data and receives results of that data back to them. The domain layer applies rules to the request that follow the business rules we laid down for our system. The data layer is the final layer and manages data stored in the system.

Advantages:

- Has better security due to the client not having direct access with the data layer where all the sensitive data is being held.
- Can be modified easily on servers and not have to redistribute clients.
- Data corruption from clients doesn't matter as the middle layer would detect anomalies and prevent the bottom layer from receiving bad data.

Disadvantages:

- Complex architecture due to it having the request access each layer with there being 3 layers.
- Reduced performance from having multiple layers of communication.
- Implementation takes a little

The Microservices Architecture:

The microservices architecture is mainly used in web-based applications but is about separating modules from one another keeping them independent and just loosely coupling them together so they can still interact in larger programs.
Advantages:

- Isolated failures from classes not relying on other classes allow for easier fixes and do not interrupt the entire program or cause it to fail as a whole.
- When required to scale important modules it doesn't impact the entire system and can be more efficient.
- If the program is small it makes it really quick to deploy.

Disadvantages:

- Communication between each service is more complex due to having a large number of modules.
- Debugging can take a while where each module will create its own logs which will need to be looked through to figure out the issue instead of a bunch of modules only having a single log.
- Useful in big teams but harder to implement and time consuming for smaller teams.

Architecture analysis and comparison:

Each architecture has both positives and negatives but the best fit for Beverage Bookers would probably be Three-Layer Architecture. The top layer and bottom layer are requirements for our project as we require the storage of transactions that occur on the system and the user interface should request specific actions that the server takes care of to make the client easy to use and understand while not making a bulky application that runs poorly. If we were to use the microservices architecture it could take a long time to complete due to having too many separate modules and classes to take care of but the architecture could have been useful for how quick deployment of the program is in that architecture. However with client / server architecture it is similar to the three-layer architecture but it is missing the important middle layer preventing interactions of unauthorized access to sensitive information which is one of our nfrs as we deal with transactions and need to keep information safe.