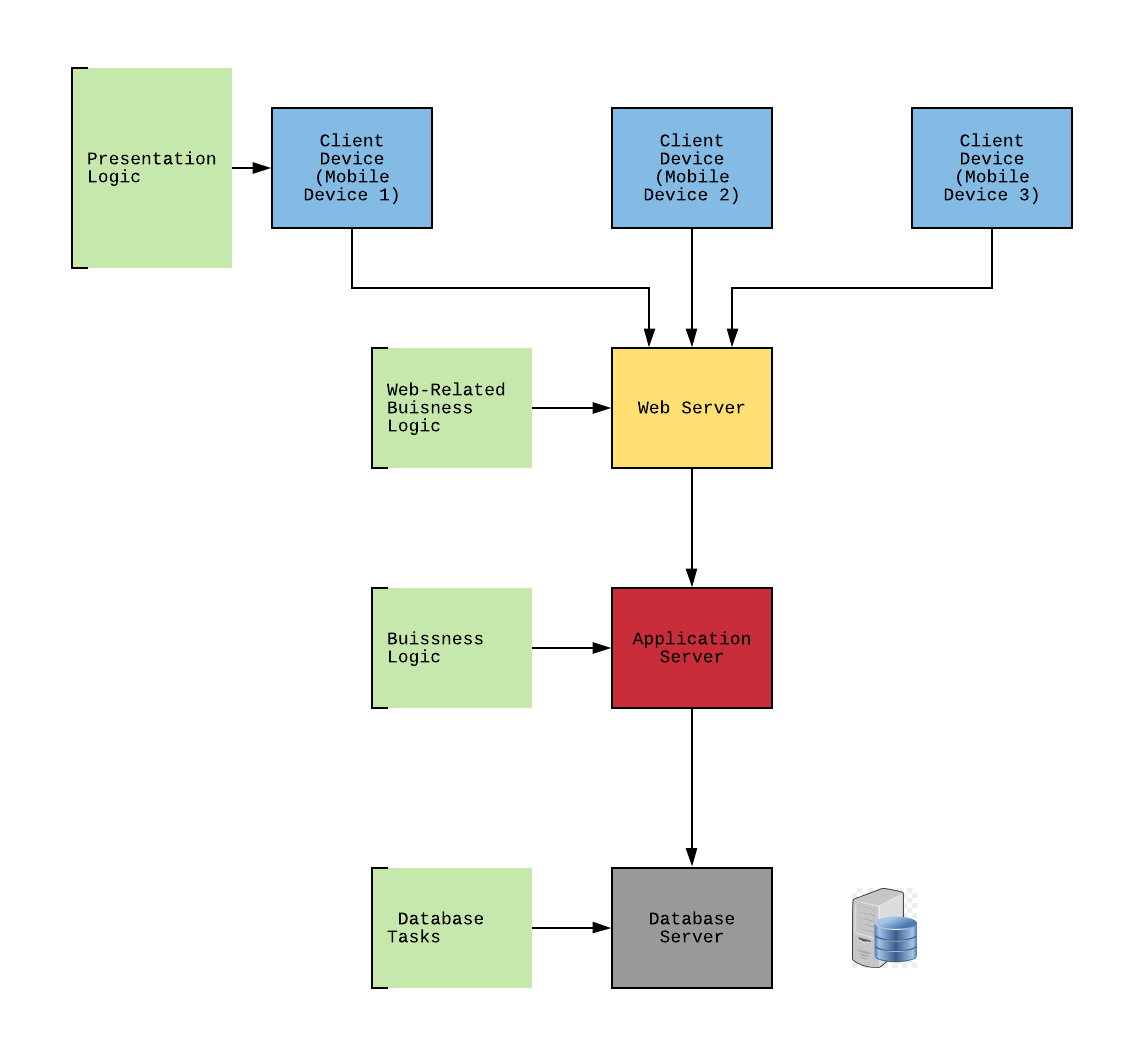
Chapter 8

Produce the client / server architecture for your system-

* Identify where the business logic will reside
* Identify where the presentation logic will reside
* Identify where the data logic will reside



The client / server architecture system is pictured above. This shows FullStock’s n-tiered architecture design. This structure was chosen because it best accommodates our team's need for potential scalability and e-commerce platform. Later in this section there will be a more in depth explanation of why the n-teired architecture design was chosen.

**Nonfunctional Requirements**

| **Requirements** | **Server-Based** | **Thin Client - Server** | **Thick Client - Server** |
| --- | --- | --- | --- |
| **Operational Requirements** |  |  |  |
| System Integration Requirements | Yes | Yes |  |
| Portability Requirements |  | Yes |  |
| Maintainability Requirements | Yes | Yes |  |
| **Performance Requirements** |  |  |  |
| Speed Requirements |  | Yes |  |
| Capacity Requirements |  | Yes |  |
| Availability/Reliability Requirements | Yes | Yes |  |
| **Security Requirements** |  |  |  |
| High System Value | Yes | Yes |  |
| Access Control | Yes |  |  |
| Encryption/Authentication |  | Yes |  |
| Virus Control | Yes |  |  |
| **Cultural/Political Requirements** |  |  |  |
| Multilingual Requirements |  | Yes |  |
| Customization Requirements |  | Yes |  |
| Making Unstated Norms Explicit |  | Yes |  |
| Legal Requirements | Yes | Yes |  |

The table above shows the breakdown of the different requirements and the relationship between the chosen thin client server architecture.

**Specific nonfunctional requirements for Full Stock**

| **Requirements** | **Number** | **Description** |
| --- | --- | --- |
| **Operational Requirements** |  |  |
| Technical | 1.1 | The system will work with the internet. |
|  | 1.2 | Consumers can use the application via their mobile device. |
| System Integration | 1.3 | The system needs to use MS to properly track the inventory quantities. |
|  | 1.4 | The system needs to work with the application to properly update the restaurants ordering system and inventory database. |
| Portability | 1.5 | The system will be able to work with multiple operating systems and mobile devices (iphone, android, etc.) |
| Maintainability | 1.6 | The system will be updated as needed. |
| **Performance Requirements** |  |  |
| Speed | 2.1 | Response time must be less than 5 seconds. |
| Capacity | 2.2 | Maximum of 50 users at a time when placing an order for food. |
| Availability/Reliability | 2.3 | System will be accessible during the restaurant's business hours. |
| **Security Requirements** |  |  |
| High System Value | 3.1 | No special system value requirements are anticipated. |
| Access Control | 3.2 | Consumers can access their accounts using their mobile app. Via username and password. |
| Encryption/Authentication | 3.3 | Payment is secure and information must be re-entered each time. |
| Virus Control | 3.4 | Virus threats are constantly monitored. The application cannot be downloaded if the user has a virus on their mobile device. |
| **Cultural/Political Requirements** |  |  |
| Multilingual | 4.1 | Restaurants have the option to switch the applications language to Spanish, English, and Chinese to accommodate different types of consumers. |
| Customization | 4.2 | Each application is customized to the particular restaurant. |
| Unstated Norms | 4.3 | The application is currently only in the US. There are currently no unstated norms. |
| Legal | 4.4 | No special legal requirements are anticipated. |

The table above shows a more in depth look at the requirements for Full Stock.

Discuss your architecture design. Why are you designing your system in this way?

Our team chose n-tiered architecture design for FullStock. A n-tiered design for use over a two or three-tiered because of its scalability. The separate servers provide the allowance of the shared load. Our team does not want the server to crash while consumers use it. This type of design is common for e-commerce. Our team decided on using a thin client-server architecture to reduce the overhead and maintenance. Restaurant industry can be busy and we did not want our clients to worry about maintaining the server. As seen in the non-functional requirements figure our team’s system requirements support a thin architecture. The following list are reasons for a thin client-server architecture:

Operational requirements - our application can be integrated with a desktop and have vast portability requirements. Data from the application is constantly being transferred to the restaurant’s own systems.

Performance - Client-server architecture is best suited for our application. This type of architecture is scalable meaning that companies can easily change the hardware to the speed requirements. Also if a server crashes, then another server can easily be used. Users will have an easier time using Full Stock.

Security - A client-based server provides more advanced authentication tools to ensure that the users and the restraunts personal information won’t be stolen.

Cultural and Political - Separating the presentation logic from the application logic and data allows more languages to be used and more customization. The restaurant industry attracts a variety of people from different backgrounds. It is important to accommodate all types of employees. Having a thin-server architecture allows an easier separation.