Hotel Reservation System

Group 3

- Joshua Espana
- Griffin Graham
- Noah Gumm
- Zachary Marrs

Project Overview

- Our team has been tasked with the creation of a modern hotel reservation system.
- The goals of the project are:
  - Streamline reservation process by allowing customers to manage their bookings independently.
  - Empower hotel staff management capabilities by providing them with administrative functionality over the reservation process.
  - Enhance customer experience with a modern approach to the application, which will include mobile-friendly interfaces.

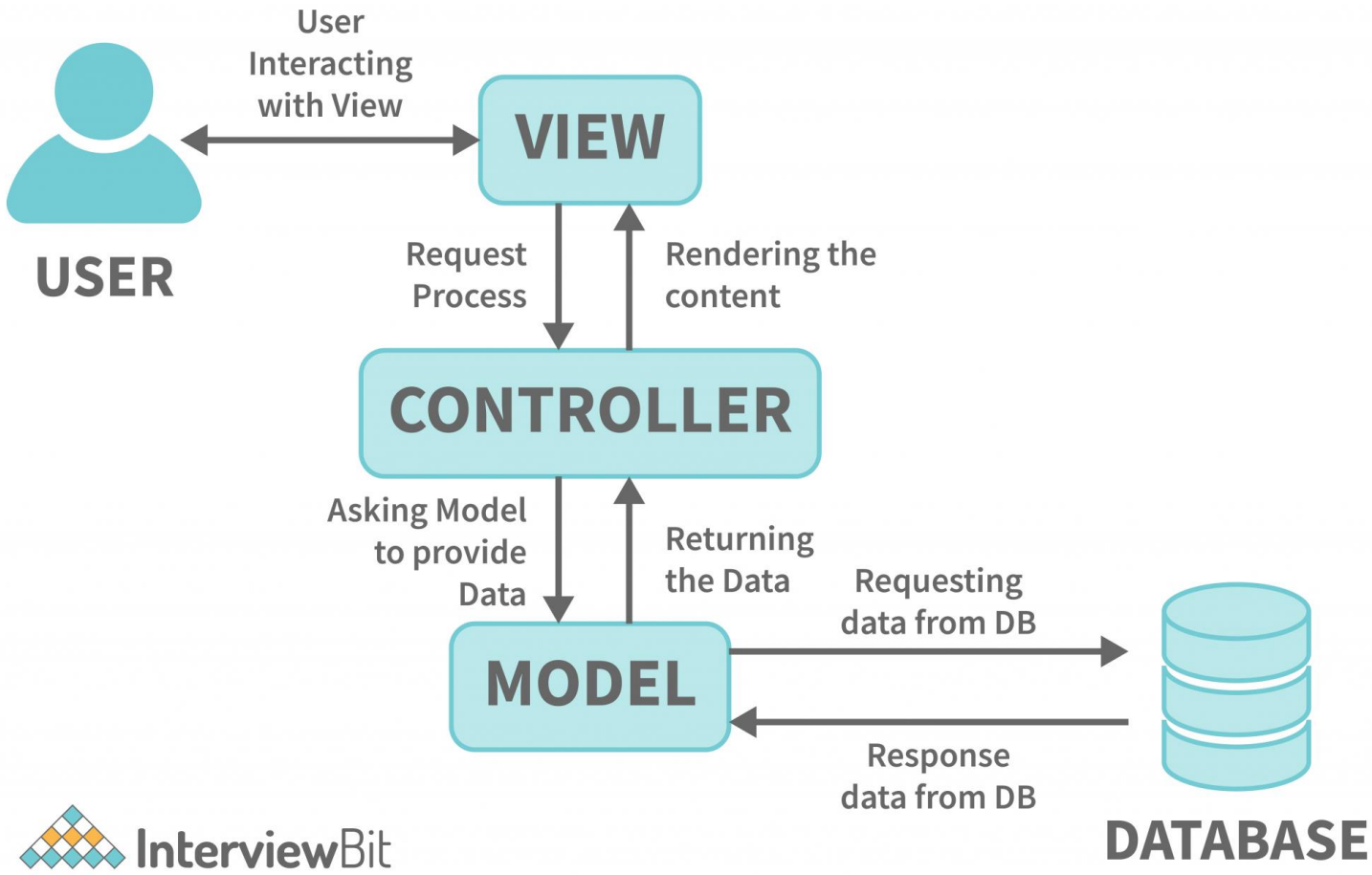
Key Architectural Drivers

- Scalability: The new system must accommodate the current number of users and be scalable to handle future growth.
- User Experience: Providing a seamless and intuitive user experience to enhance customer satisfaction and staff efficiencies.
- Modular Development: The new system should support modular development to enable efficient team collaboration and maintenance.

Architectural Style Choices

- Candidate Architectural Styles:
  - Client-Server Architecture
    - Pros
      - Centralized control over data and business logic.
      - Scalability by distributing the load across multiple servers.
    - Cons
      - Increased latency due to network communication.
      - Heavy reliance on server resources.
  - Model-View-Controller (MVC) Architecture: Offers separation of concerns, modularity, and flexibility but requires more upfront design effort.
    - Pros
      - Separation of concerns for better maintainability.
      - Flexibility to change UI without affecting the underlying logic.
    - Cons
      - Initial setup complexity.
      - Proper communication between components can be a challenge.
- Chosen Architectural Style: Model-View-Controller (MVC)
  - We chose MVC architecture for its ability to separate concerns, facilitate modular development, and provide flexibility for future enhancements.

The Model View Controller Architecture



Conclusion

- Chosen Architectural Style: Model-View-Controller (MVC)
- Open Issues/Risks:
  - Ensuring proper separation of concerns and maintaining modularity throughout development.
  - Designing efficient communication between the model, view, and controller components.
  - Managing complexity and dependencies between different modules within the MVC architecture.
  - Addressing scalability concerns as the system grows and evolves.