SOFTWARE ARCHITECTURE PROJECT

Online Hotel Booking Project

Toygar Ateş

July 26, 2024

Table of Contents

List of Figures 5

List of Tables 6

1 Introduction 8
- 1.1 Purpose and Scope of the System . . . 8
- 1.2 Stakeholders and their Characteristics . . . 8
- 1.3 System Overview . . . 9
  - 1.3.1 System Perspective . . . 9
  - 1.3.2 System Primary Functions . . . 10
  - 1.3.3 Architecturally Significant Requirements . . . 12
  - 1.3.4 Constraints and Concerns . . . 12

2 References 15

3 Glossary 16

4 Tactics and Patterns on Selected Quality Attributes 17
- 4.1 Deployability . . . 17
  - 4.1.1 Step 1: Review Inputs . . . 17
    - 4.1.1.1 The purpose of the design round: . . . 17
    - 4.1.1.2 The primary functional requirements: . . . 17
    - 4.1.1.3 The primary quality attribute (QA) scenarios: . . . 17
    - 4.1.1.4 Constraints & Concerns . . . 18
  - 4.1.2 Step 2: Establish Iteration Goal by Selecting Drivers . . . 18
  - 4.1.3 Step 3: Choose One or More Elements of the System to Refine . . . 18
  - 4.1.4 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers . . . 18
    - 4.1.4.1 Script deployment commands . . . 18
    - 4.1.4.2 Package dependencies . . . 18
    - 4.1.4.3 Roll back . . . 18
    - 4.1.4.4 AB Testing . . . 19
  - 4.1.5 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces . . . 19
  - 4.1.6 Step 6: Sketch Views and Record Design Decisions . . . 19
  - 4.1.7 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose . . . 20

4.2 Availability . . . 20
4.2.1 Step 1: Review Inputs . . . 20
4.2.1.1 The purpose of the design round: . . . 20
4.2.1.2 The primary functional requirements: . . . 20
4.2.1.3 The primary quality attribute (QA) scenarios: . . . 20
4.2.1.4 Constraints & Concerns . . . 21
4.2.2 Step 2: Establish Iteration Goal by Selecting Drivers . . . 21
4.2.3 Step 3: Choose One or More Elements of the System to Refine . . . 21
4.2.4 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers . . . 21
4.2.4.1 Monitor . . . 21
4.2.4.2 Redundant Spare . . . 21
4.2.4.3 Transactions . . . 21
4.2.4.4 Circuit breaker . . . 22
4.2.5 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces . . . 22
4.2.6 Step 6: Sketch Views and Record Design Decisions . . . 22
4.2.7 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose . . . 23
4.3 Usability . . . 23
4.3.1 Step 1: Review Inputs . . . 23
4.3.1.1 The purpose of the design round: . . . 23
4.3.1.2 The primary functional requirements: . . . 24
4.3.1.3 The primary quality attribute (QA) scenarios: . . . 24
4.3.1.4 Constraints & Concerns . . . 24
4.3.2 Step 2: Establish Iteration Goal by Selecting Drivers . . . 24
4.3.3 Step 3: Choose One or More Elements of the System to Refine . . . 24
4.3.4 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers . . . 25
4.3.4.1 MVC Pattern . . . 25
4.3.4.2 Maintain Task Model . . . 25
4.3.4.3 Maintain User Model . . . 25
4.3.4.4 Maintain System Model . . . 25
4.3.5 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces . . . 25
4.3.6 Step 6: Sketch Views and Record Design Decisions . . . 26
4.3.7 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose . . . 26
4.4 Security . . . 27
4.4.1 Step 1: Review Inputs . . . 27
4.4.1.1 The purpose of the design round: . . . 27
4.4.1.2 The primary functional requirements: . . . 27
4.4.1.3 The primary quality attribute (QA) scenarios: . . . 27
4.4.1.4 Constraints . . . 27

4.4.1.5 Concerns . . . 27
4.4.2 Step 2: Establish Iteration Goal by Selecting Drivers . . . 27
4.4.3 Step 3: Choose One or More Elements of the System to Refine . . . 28
4.4.4 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers . . . 28
4.4.4.1 Authenticate actors . . . 28
4.4.4.2 Authorize actors . . . 28
4.4.4.3 Encrypt data . . . 28
4.4.4.4 Intercepting Validator . . . 28
4.4.5 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces . . . 28
4.4.6 Step 6: Sketch Views and Record Design Decisions . . . 29
4.4.7 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose . . . 29

5 Architecture Design **30**
5.1 Module View . . . 30
5.1.1 Stakeholders' uses of this view . . . 30
5.1.2 UML Diagrams for Module View . . . 30
5.2 C&C View . . . 33
5.2.1 Stakeholders' uses of this view . . . 33
5.2.2 UML Components Diagram for C&C View . . . 33
5.3 Allocation View . . . 34
5.3.1 Stakeholders' uses of this view . . . 34
5.3.2 Deployment Diagram for Allocation View . . . 35
5.4 Behavior . . . 35
5.4.1 Stakeholders' uses of this view . . . 35
5.4.2 UML Diagrams for Behavior . . . 36
5.5 Rationale . . . 37

List of Figures

1.1 System Context Diagram for Online Booking Project 9

4.1 Diagram Sketch of Deployment ADD 19
4.2 Deployment Diagram Sketch of Availability 22
4.3 Component Diagram Sketch of Availability 23
4.4 Component Diagram Sketch for Usability 26
4.5 Package Diagram Sketch for Usability 26
4.6 Component Diagram Sketch for Security 29

5.1 Use Case Diagram For Online Booking System 31
5.2 Class Diagram For Online Booking System 32
5.3 Component Diagram for CC View 34
5.4 Deployment Diagram for Online Booking System Allocation View 35
5.5 Activity Diagram to Represent the Most Architecturally Critical Function . . . 36

List of Tables

1.1 Utility Tree for Software Quality Attributes 14

5.1 Example Table for Decisions Design Decisions 37

Revision History

Revision	Date	Author(s)	Description
1.0.0	03.12.2023	Ateş, T. , Koşaroğlu, A.İ.	SAP v1
2.0.0	18.12.2023	Ateş, T. , Koşaroğlu, A.İ.	SAP v2

1 Introduction

1.1 Purpose and Scope of the System

The purpose of the system is to enhance the booking experience for travelers.

The primary aim is to provide users easy access to up-to-date information, empowering them to make informed decisions about their accommodations. The website has rates and comments section for both the users to share their experiences and hotels to use as a service quality control mechanism. Users can express their experiences on the website by adding pictures and comments to ensure they receive exactly what they expect. Online hotel booking website also suggests nearby experiences to increase the holiday experience for customers.

For the scope of the website, the project is created for profit, by adding a commission to prices of hotel books, flight tickets, car rentals and experiences in many destinations worldwide. The website offers a wide range of options for users to choose from, including cozy bed-breakfasts to 5 star luxury hotels. The website provides users with a user-friendly interface to search and compare low prices on hotels, homes, flights, and car rentals and tours, and is designed to make it easy for users to take advantage of discounts and promotions by employing a loyalty program.

To summarize the scope of the Online Hotel Booking project:

- The platform is a global platform, providing booking services from all around the world.
- The online system covers a wide range of properties; hotels, resorts, rent-a-cars, touristic spots and suggested experiences.
- Customers can access real time information about the availability, prices, special offers and comments of other users.
- The Booking platform provides flexible reservation options which include free cancellation, prepayments, special requests, also gives customers the opportunity to plan their visas for abroad countries.
- The Booking project offers reward programs to encourage people to repeat bookings and provides additional benefits to regular users.

1.2 Stakeholders and their Characteristics

There are five main stakeholders for the online booking project which are Customers, Hotel Providers (Hosts), Transportation Partners, Employees and Investors.

Customers: Who are looking for travel hotel reservations directly. They may need to browse through available restaurants, rent-a-car services, touristic activities or may inquire information

about hotel and places.
Hotel Providers (Hosts): They are the individuals or organization officials which provide properties to be browsed by the users. They have to conform to certain quality standards because of customer reviews and ratings.
Transportation Partners: Who have partnerships with travel services such as car rentals and airport transfers. They can also choose to provide API's to the Booking system to employ collaboration, and is applicable for airlines and other travel related businesses.
Employees: A diversified employee set, who are to develop and maintain the software, or call-centers for direct phone calls for customer services. They should be continuously learning and developing the project.
Investors: Who are focused on returning on their investments on the project, by the means of profitability and sustainability of the platform, employing growth strategies and transparency by communicating with the system owners for financial returns.

1.3 System Overview

1.3.1 System Perspective

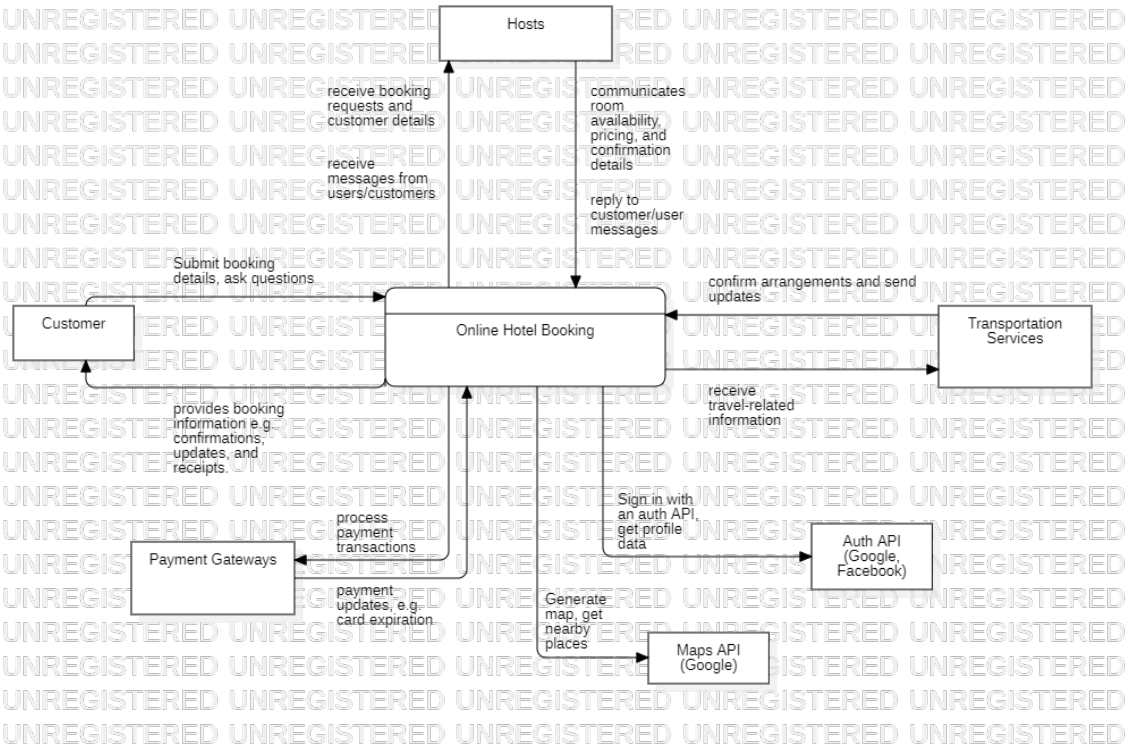

Figure 1.1: System Context Diagram for Online Booking Project

1.3.2 System Primary Functions

1. **Comprehensive Search Functionality:**
   - 1.1 Search by location, date, number of adults and children, and room type.
   - 1.2 Remember and display last searches.
2. **Registering New Properties:**
   - 2.1 Allows hosts to add new properties for users to find and book.
   - 2.2 Should be able to add rooms, explanations, tags, amenities, and photographs of the property.
3. **Property Page:**
   - 3.1 A page specific to a chosen property. It should include explanations and photographs prepared by the host.
   - 3.2 Users should be able to see the prices and rooms and book the property here.
4. **User Authentication:**
   - 4.1 Allows users to log in or sign up.
   - 4.2 Auth integration with Google and Facebook.
   - 4.3 Special partner accounts for hosts.
5. **Loyalty Program:**
   - 5.1 Implement a loyalty program (e.g., "Genius") that offers many benefits such as discounts to people who use the software regularly.
6. **Notifications:**
   - 6.1 Allow sending and receiving notifications from a hotel or a booked service.
7. **Google Maps Integration:**
   - 7.1 Use Google Maps to show hotel locations and nearby places.
   - 7.2 Use Google Maps to identify key location distances to the booked place.
8. **Tag System:**
   - 8.1 Tags for activities and aspects of the booked service (e.g., breakfast included, dinner included, fun activities).
   - 8.2 Hotel types (e.g., hotel, house, apartment, hostel, pension).
   - 8.3 Tags for specific needs or interests. (e.g., vegan availability)
9. **Search Filters:**
   - 9.1 Filters for various features (e.g., parking availability, fitness facilities, room).
   - 9.2 Distance from city center (e.g., within 1, 3, 5 km).

9.3 Ratings filter (e.g., 9+).

10. **Dynamic Pricing:**
    10.1 Prices should be displayed in local currency (e.g., TRY), with a system to fetch and update exchange rates.
11. **Payment System:**
    11.1 Supports prepayments and other payment methods, preferably using an API.
    11.2 Supports transferring payment to the host with an added commission.
12. **Real-Time Room Availability Tracking:**
    12.1 The system should not allow booking if the service is already full.
    12.2 The system should be able to reduce the number of available space after a successful book.
13. **Ratings and Reviews:**
    13.1 Rate using a star system
    13.2 Add a public comment to a previous stay at the property, which includes text and optionally photographs.
    13.3 Categories for ratings (e.g., staff, facilities, cleanliness, comfort, value for money, location, Wi-Fi).
14. **Direct Communication with Property Owners:**
    14.1 Publicly available "Ask the owner" section for all users to message the host directly, but without booking.
    14.2 Private messaging system after booking the property, for questions e.g. entry & exit times.
15. **Recommendations System:**
    15.1 Suggest other hotels based on user behavior ("Customers who viewed this also viewed...").
16. **Travel Articles/Blog Posts:**
    16.1 Provide content and blog posts related to travel guides, tips, experiences.
17. **Experience Booking:**
    17.1 Offer booking for additional experiences (e.g., tours).
    17.2 Options for different languages in guided tours.
    17.3 Detailed information on what's included (e.g., hotel pickup, equipment) and what's not (e.g., photographs and videos).
18. **User Page:**

18.1 A Page where users can see their past and current bookings.

18.2 Users can also manage their preferences, and make changes to their bookings using this page.

1.3.3 Architecturally Significant Requirements

Utility tree has shown with table 1.1

1.3.4 Constraints and Concerns

There are several constraints and concerns that should be followed by Online Hotel Booking such as:

Constraints:

- Budget Constraints: Limitations in funding affecting the technology choices, development power and maintenance costs limit.
- Technology: Constraints imposed by the chosen technology stack, both for the users and developers.
- Time Constraints: Limited time for development and deployment, impacting project scope, especially in usability and testing.
- Resource Availability: Availability and expertise of the development team, and limitations in terms of hardware and software resources.
- Competitors: When in a competition with rivals, the extent of discounts and promotions that can be offered will have a limit because maintaining profitability is also needed.
- Legal: Compliance to national and international laws and regulations, such as KVKK in Turkey.

Concerns:

- For data security and privacy, managing the risk of data or unauthorized access to sensitive user data.
- For payment security, addressing vulnerabilities in the payment system that could lead to unauthorized transactions.
- For continuous connectivity, designing a platform that can operate efficiently with limited internet connectivity, challenges in delivering real-time updates and maintaining a responsive user interface.
- For competition with rivals, developing different bunch of strategies and being able to test them quickly to attract users in a competitive environment.

- For information accuracy, reducing the risk of displaying inaccurate information to users during the booking process.
- For localization requirements; language support for users of each country, and cultural specialization of the system should be kept in mind.

Quality Attribute	Attribute Refinement	ASR Scenario
Security	Access control	(1) A user attempts to access the page of another user by its URL and is not allowed access. (H, M)
	Encryption	(2) A user sends a message containing sensitive information and the message is encrypted before being sent. (M, M) (3) User books a property and this data is saved in the database as encrypted. (M, L)
Usability	Accessibility	(4) A user with a visual impairment uses a new version of the software using a screen reader to navigate the application without encountering any issues. (M, H)
	User feedback	(5) A user submits feedback on the application's usability and the feedback is recorded and analyzed within 1 week. (H, M)
	User Experience	(6) A new user is accessing the system for the first time but is able to search for the property using auto fill and they find it in less than 10 seconds because they find it intuitive. (H, H)
Availability	Reliability	(7) The application is able to handle a large number of requests, 3x of the peak access, without crashing or slowing down more than 50% . (H, H)
	Redundancy	(8) A server hosting the application fails, but the application is still available because it is hosted on multiple servers, achieving 99.9% uptime. (M, H)
Deployability	New Update	(9) A new feature or update is added to the application without requiring more than 30 minutes to take affect to all users without any downtime. (M, H) (10) A tester of the staging environment is testing the software and the system is able to rollback to the previous version when the tester finds a bug within 30 minutes. (H, M)
	Preference Testing	(11) A new feature is being A/B tested and it is deployed only for a 100 people, and is then reverted or applied to the production environment within 3 days if found to be desirable. (M, H)

2 References

This SAP file is prepared with the light of information from Software Architecture in Practice:

Bass, L., Clements, P. and Kazman, R. (2022). Software Architecture in Practice (4th ed.). Addison-Wesley.

Booking.com. (2023). Booking.com: The largest selection of hotels, homes, and vacation rentals. Booking.com. https://www.booking.com

3 Glossary

- API: Application Programming Interface.
- CDN: Content Delivery Network.
- Host: People or organizations who open properties for booking.
- Property: Places that can be booked.
- Customers: People who want to book properties online.
- HTTP/HTTPS: Hypertext Transfer Protocol/ Hypertext Transfer Protocol Secure.
- Investors: People who are partner of a certain part of a company.
- Kubernetes: Kubernetes is an open-source container orchestration system for automating software deployment, scaling, and management. Originally designed by Google, the project is now maintained by the Cloud Native Computing Foundation.
- Docker: Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers.

4 Tactics and Patterns on Selected Quality Attributes

4.1 Deployability

4.1.1 Step 1: Review Inputs

4.1.1.1 The purpose of the design round:

- To design a deployment system to ensure new updates are tested with users and continuous improvement takes place.
- To be able to have the system restartable and scalable with the deployment strategies taken place.
- To be able to serve the new updates to the system without downtime, keeping availability.
- Making sure that the deployment strategy supports keeping the data of users persistent, i.e. data loss is not incurred between deployments.

4.1.1.2 The primary functional requirements:

- Developers should be able to publish new updates and rollback when necessary.
- User data integrity should be kept between deployments.
- New UI tests can be deployed in the system with a small set of users.

4.1.1.3 The primary quality attribute (QA) scenarios:

- ASR Scenario (7) - Availability through Deployment Scaling
- ASR Scenario (8) - Availability through Deployment Redundancy
- ASR Scenario (9) - Deployment Speed of the New Update
- ASR Scenario (10) - Deployment Rollback Speed
- ASR Scenario (11) - Deployability For AB Testing

4.1.1.4 Constraints & Concerns

- Budget Constraints - Deployment should keep costs within reasonable limits.
- Resource Availability - Deployment complexity should be kept reasonable for the team to create and maintain the system.
- Information Accuracy Concern - Displaying inaccurate information should be avoided by keeping the data during a new deploy safe and sound.

The purpose of the first round of deployability will be to create a system for deployability, making sure that the deployment system supports all the necessary requirements mentioned above.

4.1.2 Step 2: Establish Iteration Goal by Selecting Drivers

The goal of the first iteration is to try to establish the base system for deployment. Thus, the driver here is the ability to create testing deployments and constructing the initial system that will support the aforementioned attributes.

4.1.3 Step 3: Choose One or More Elements of the System to Refine

Since this iteration of the ADD is the first, the element picked here is the whole system. The initial elements will be chosen and constructed. However, since the booking system is a website, it will have to consist of clients and servers, so the deployment will mainly aim to improve the server here.

4.1.4 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

4.1.4.1 Script deployment commands

To be able to deal with new deployments, creating a continuous deployment environment is very beneficial.

4.1.4.2 Package dependencies

Instead of using a microservices pattern and dealing with service versions, packaging dependencies in the deployment is found to be better since it has less overhead both in deploying the system and in maintaining it.

4.1.4.3 Roll back

When an error occurs, the deployment is to be reverted back to a working version. This is made easier with script deployment commands.

4.1.4.4 AB Testing

To increasingly improve the user experience and usability and to test the efficiency of new features, an ab testing environment will be created, and it will be made possible with the script deployment commands and rollbacks. Packaging dependencies will make sure that the AB tested versions will work correctly.

4.1.5 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

Scripting the deployment will be made possible using virtualization technologies. One such virtualization is a docker environment, which can containerize the software and make sure that the deployments are always behaving in similar manner. Kubernetes will also be used to manage the containers and to be able to scale the system whenever needed, establishing the basis for the QA's mentioned above.

To be able to maintain the data integrity, only one service will access the database at one time. The deployment config shall convey that information. The database is also, by nature, persistent in that service.

Creating the necessary configurations, kubernetes will also handle the load balancing of client connections, distributing IP's to certain containers.

NoSQL database is a strong preference here because of the flexibility of the structure which is needed in the AB testing. For people that use different versions, NoSQL can easily keep up with different structured data.

Microservices pattern is not employed because of the increased overhead of the version control and deployment complexities.

4.1.6 Step 6: Sketch Views and Record Design Decisions

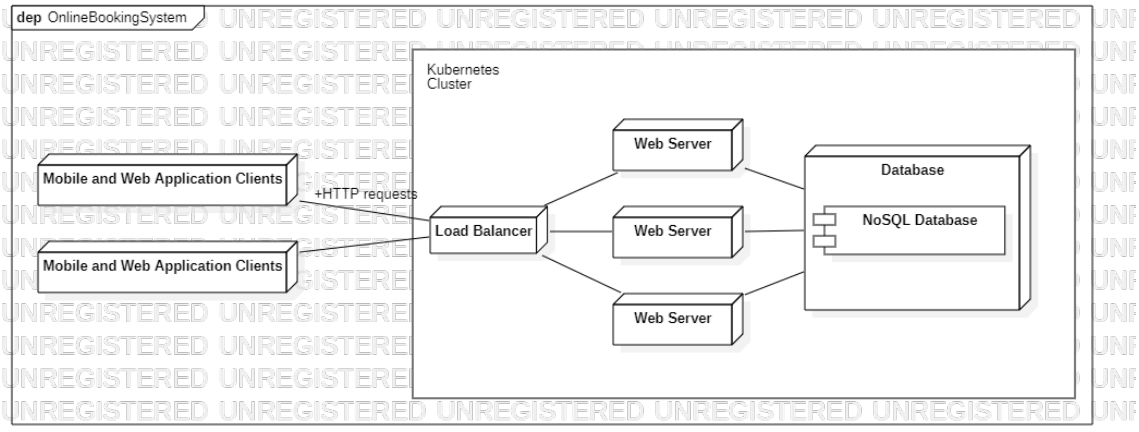

Figure 4.1: Diagram Sketch of Deployment ADD

The Figure above captures the ideas mentioned in the step 5 of this ADD cycle. As can be seen on the diagram, the server is isolated from the clients within a kubernetes cluster. There is only one instance of the database node which contains a NoSQL database to aid on the AB testing.

4.1.7 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

The iteration goal is satisfied here because we now have an initial system that gives an idea on the general deployment structures. The database type, i.e. NoSQL is selected, or very least strongly suggested as a side effect of this iteration.

Continuous improvement purpose is considered and realized under AB testing and continuous deployment and thanks to Kubernetes, downtime is also taken under control. Keeping the data inside one node, data integrity is also kept which was one of the main goals of the iteration. Overall, it can be said that the iteration is successful and sets the basis on which future improvements can take place.

4.2 Availability

4.2.1 Step 1: Review Inputs

4.2.1.1 The purpose of the design round:

To improve on the deployment plan of the previous ADD iteration to make sure the availability requirements are met.

4.2.1.2 The primary functional requirements:

- To make sure that the user actions are transactional, and the system is reliable even on certain failures. E.g. payments are not lost.
- To make sure that users can always reach property hosts through the messaging system with high availability.
- To make sure that the most up to date currency info is available to the users and fallback mechanisms are present when its not the case.

4.2.1.3 The primary quality attribute (QA) scenarios:

- ASR Scenario (7) - Availability through Scaling
- ASR Scenario (8) - Availability through Redundancy

4.2.1.4 Constraints & Concerns

- Budget Constraints - Availability should keep costs within reasonable limits.
- Resource Availability - Availability should not add extra development complexity to the system, but keep it as minimal as possible within reason.
- Information Accuracy - Making the system more available should not impair data integrity but improve on it.

The purpose of the first round of availability will be to increase the availability of the general system through selected patterns and tactics, making sure that users can mostly access the features of the system. We will extend on the system after the previous ADD iteration cycle, the deployment round.

4.2.2 Step 2: Establish Iteration Goal by Selecting Drivers

The goal of this iteration will be to accomplish high availability through reliability and redundancy. General patterns to increase availability will be employed to create the initial fault detection and tolerance mechanisms. The main goal will be to increase the availability for users, or to be more specific, client connections.

4.2.3 Step 3: Choose One or More Elements of the System to Refine

The refine will primarily aim the web server element, since it can hold the components needed to increase availability. Also, the cluster config is also selected to be refined since it highly affects availability.

4.2.4 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

4.2.4.1 Monitor

A monitor element should be initialized because the system needs a way to understand the faults in the system so that other parts of the architecture can take action. Luckily, kubernetes provides a mechanism for adding monitors and heartbeats into the system using configs, so that will be utilized.

4.2.4.2 Redundant Spare

Redundant spares should be configured to make sure that the availability is high enough. Deployment iteration goes hand in hand with this pattern.

4.2.4.3 Transactions

To be able to prevent faults in the first place, transactions will be used. This will make sure that the payments are final and booking events are not lost.

4.2.4.4 Circuit breaker

Currencies will be constantly fetched from outside sources, and some kind of circuit breaker mechanism is needed in the event of failure of those API's. This will make sure that the system stays alive when such an error occurs.

4.2.5 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

We will add two monitors in the system so that the processes can be monitor from inside and outside. Also, we can configure the Web Server count here to be at least 3, so that we have also satisfied the Redundancy tactic that we have discussed. In order to satisfy the ASR scenarios, we shall also set the config to auto scale when the monitor senses clogging in the system.

The second monitor will be inside the web server, so that the server application itself can monitor the errors and then using the network util, can tell the other monitor the required action to take, e.g. restarting the server. This way, we can make sure that web servers are alive and working and availability is maximized.

In order to actualize the transaction pattern, we have added the request handler component. This component transactionalizes the payments so that when a user books a property, it never gets finalized until everything is finished at the same time. DB Interface will get used here so that multiple orders of the same payment also never gets through.

Last but not least, in order to apply the circuit breaker pattern to increase the availability, we have defined a currency fetcher component. This component uses the network util to fetch the currencies regularly, and implements a circuit breaker inside so that when after not responding to more than 5 requests, it sets a timeout and then does not ask the currency again for 1 hour. It will also set a flag so that the users can see the prices in the default currency.

4.2.6 Step 6: Sketch Views and Record Design Decisions

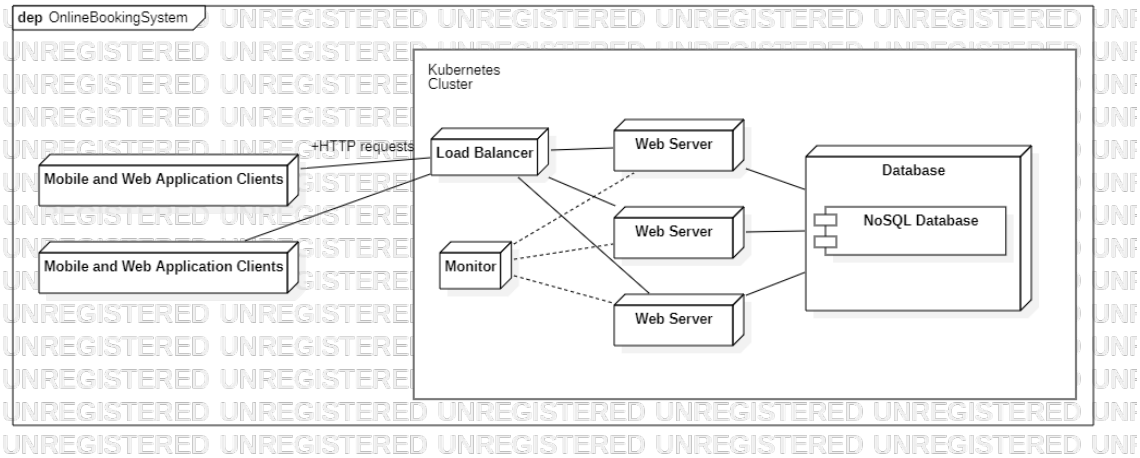

Figure 4.2: Deployment Diagram Sketch of Availability

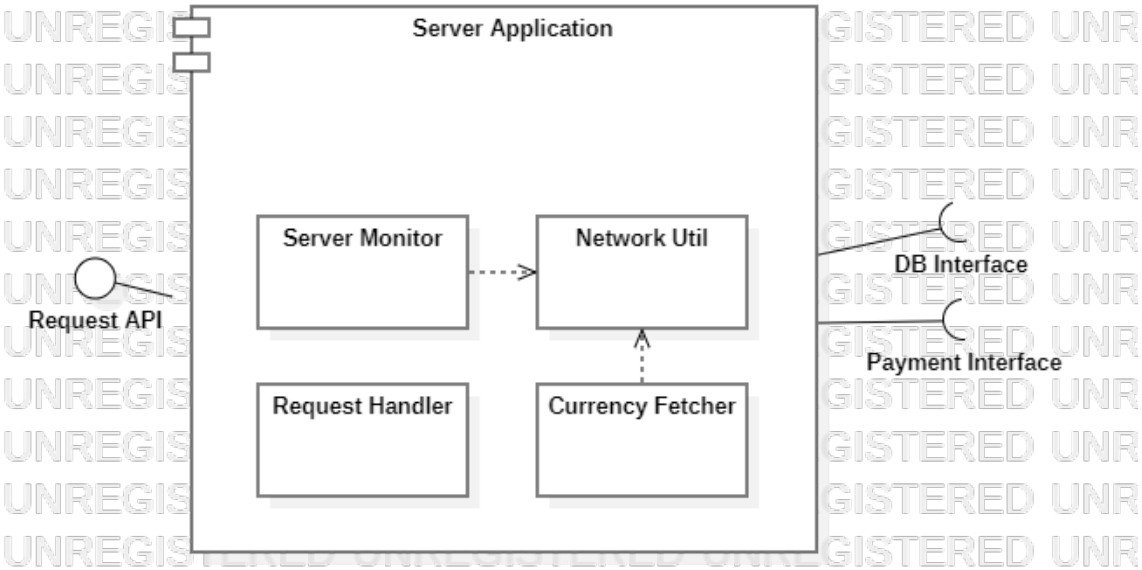

Figure 4.3: Component Diagram Sketch of Availability

Above you can see two diagrams, the first one is the deployment diagram. In this diagram, you can see that the Monitor instance is added in this iteration. Thus the first diagram shows the monitor and redundancy tactics discussed in the previous step in application.

The second diagram shows the Web Server or the server application's component diagram. Here you can see that we have the Server Monitor component which is the second way we have developed the monitor. It also includes the Request Handler which implements transactions, and the Currency Fetcher which implements the circuit breaker tactic.

4.2.7 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

We have successfully applied the tactics and patterns to achieve availability in this iteration. The availability is satisfied with avoiding the fault, and then monitoring for faults and taking actions based on those faults. Deployment config is also tweaked in this iteration so that availability and ASR scenarios are better satisfied.

4.3 Usability

4.3.1 Step 1: Review Inputs

4.3.1.1 The purpose of the design round:

To improve the usability of the application by keeping certain information about the users, tasks and the system.

4.3.1.2 The primary functional requirements:

- To make sure that the user actions are understood by the system in tasks such as searching, and in recommending users the places to book. (Functional Req. 1)
- To make sure that users with visual impairments can use the system. (ASR Scenario 4)
- Users should be able to view the properties in their own currencies. (Functional Req. 10)
- Users will get extra promotions based on their loyalty. (Functional Req. 5)
- Users should be able to get recommendations special to them. (Functional Req. 15)

4.3.1.3 The primary quality attribute (QA) scenarios:

- ASR Scenario (4) - Usability through different views
- ASR Scenario (6) - Usability through search

4.3.1.4 Constraints & Concerns

- Budget Constraints - Usability should not increase the budget more than the return on investment.
- Resource Availability - Usability should not add complex libraries to the system to require more developer expertise.

The purpose of this round of usability will be to add necessary components to create the base for maintaining user, task and system models to increase usability.

4.3.2 Step 2: Establish Iteration Goal by Selecting Drivers

Maintaining the models to infer certain information to enhance the features of searching, recommendation and loyalty system and seeing the pages in local currencies is the main driver. The second driver is to generate a system that can support multiple views of the same business logic so that the system can show different views for different people, e.g. people with visual impairments.

4.3.3 Step 3: Choose One or More Elements of the System to Refine

Two elements will be picked, one is the client, so that it supports multiple views of the same model, and the second is the server, which will maintain the users previous books and keep and process recommendations for them. Other models will also be maintained in the server except for the user model.

4.3.4 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

4.3.4.1 MVC Pattern

MVC pattern will be utilized so that the view, i.e. UI and the model and logic is separated. This way, we can implement a secondary view for accessibility and then switch to that if users client asks for that. MVC pattern also will help other aspects of development so MVC is chosen. In rationale, we have also explained the differences between the observer pattern.

4.3.4.2 Maintain Task Model

When a user searches for a place, the server will recommend an auto fill for the user so that the user's frustrations are minimized. This is especially important for regular users that are not power users.

4.3.4.3 Maintain User Model

A user model will be kept in the client so that the software can understand some of the intentions of the user, such as the currencies used or the places the user will likely search. The client will keep these in the local database.

4.3.4.4 Maintain System Model

The system will maintain a model of itself so that it can increase the revenues of the software by increasing the number of places booked. In order to accomplish this, properties will be analyzed in a network fashion, so that when viewing a place, others will be recommended to the user. Even though the end goal is the user, user model is not maintained because property recommendation is not specific to a certain user but the properties themselves.

4.3.5 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

The client will get separated to three folders, Model, View, and Controller. The view will have two folders, regular and voice accessibility. Model and Controller will keep the data and the business logic but it will be out of scope for this iteration, since we will focus on the view part.

Task model will be kept in the server in the Search Auto Fill component, whereas the system model tactic will be applied in the Property Recommendation component. They both will run regularly and use DB Interface to read through the data, and write to it after processing the model.

On the other hand, User Preferences component will be in the client application, and it will be used to actualize user model tactic, to understand and analyze the users preferences and intentions, such as if they ask for accessibility page, or they prefer certain currencies. It will use the Local DB Interface.

4.3.6 Step 6: Sketch Views and Record Design Decisions

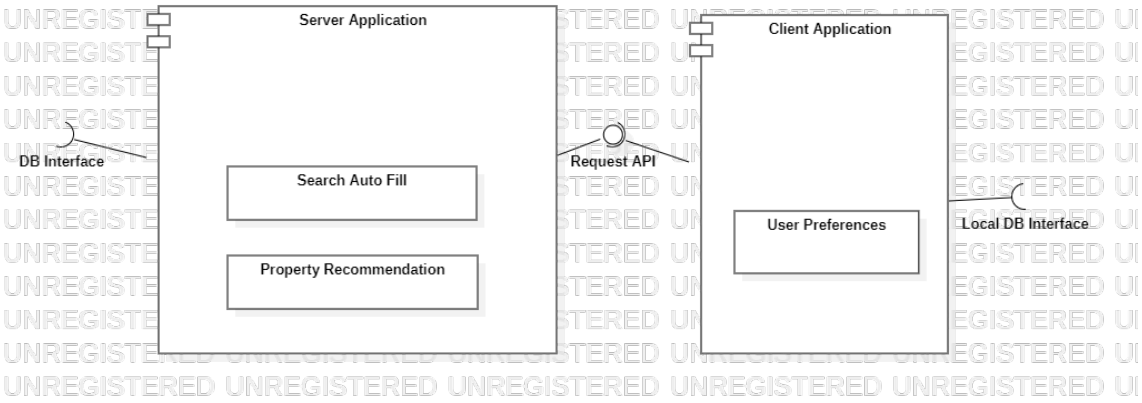

Figure 4.4: Component Diagram Sketch for Usability

Here you can see the components for maintaining user, task and system models, which are User Preferences in the Client Application, Search Auto Fill in the Server Application and Property Recommendation in the Server Application respectively.

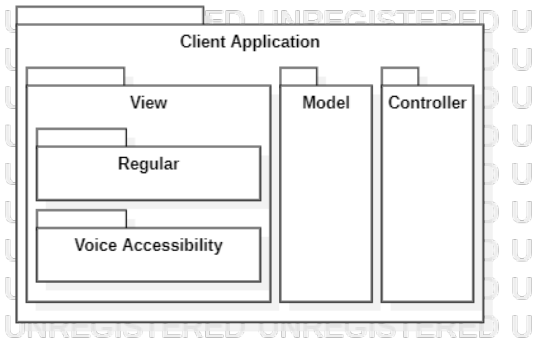

Figure 4.5: Package Diagram Sketch for Usability

In the second diagram, you can see the package layout for the client application to actualize the MVC pattern and provide multiple views that will work with the same model and controller modules.

4.3.7 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

This iteration supplied the ground work for client and server applications and specified the components responsible for actualizing the ASR scenarios and functional requirements that we picked as goals. It also specified the client structure of the modules and helped with development allocation as a side effect. We can say that the iteration was successful in achieving its goals.

4.4 Security

4.4.1 Step 1: Review Inputs

4.4.1.1 The purpose of the design round:

- To design a secure architecture for Online Booking project that protects user data, financial information, and system integrity.
- This round specifically focuses on the security quality attribute.

4.4.1.2 The primary functional requirements:

- Functional Requirement 4 - Users can login or signup securely.
- Functional Requirement 11 - Can process payments and transactions securely.
- Functional Requirement 18.1 - Users can manage their bookings and accounts.

4.4.1.3 The primary quality attribute (QA) scenarios:

- ASR Scenario (1) - Access Control
- ASR Scenario (2) & (3) - Encryption

4.4.1.4 Constraints

- Compliance with relevant security regulations
- Scalability to handle large volumes of users and transactions
- Performance and responsiveness

4.4.1.5 Concerns

- Potential security threats such as denial-of-service attacks, and account takeovers
- Integration with third-party systems
- User privacy and data protection

4.4.2 Step 2: Establish Iteration Goal by Selecting Drivers

For this iteration, the goal is to design and implement architectural structures that enhance the overall security of the online booking platform. We will try to establish the authorization and authentication of the users. We also aim to create a system that makes the database information useless if accessed, i.e. encrypted.

4.4.3 Step 3: Choose One or More Elements of the System to Refine

The elements to refine will be the client and server applications, since they will now do authentication and authorization checks of the user. Also the other components to increase security will be in the server application so that the security isn't compromised.

4.4.4 Step 4: Choose One or More Design Concepts That Satisfy the Selected Drivers

4.4.4.1 Authenticate actors

Authentication is essential in a client server application where there are many users. Authentication both ensures that the users can see their own page among many others, so the server recognizes the user and sends a subset of the information, and that they are the valid user because they have used their credentials to login to the system. This will be the first step to improve security.

4.4.4.2 Authorize actors

Authorization is the second step in maintaining the security because after authenticating the user, we also need to make sure that the user does not and cannot access others' pages, and others' data through queries. This also means that regular users cannot change property data because the system denies access.

4.4.4.3 Encrypt data

Encryption will also be essential because even when there are security leaks, sensitive data cannot be read through and is thus safe for storage. Data leaks happen from time to time and user information here is especially important to keep encrypted because both passwords and sensitive data such as credit card information should be protected.

4.4.4.4 Intercepting Validator

We shall also place an intercepting validator to protect the system from certain attacks such as denial of service, and because we the messaging system of users need a way to be encrypted before being sent to the other end. Thus it will act as a helper to the encryption of data in this sense.

4.4.5 Step 5: Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces

We will have two new components in the system, one will be the Authentication Helper which will reside in the client application, and the other will be in the server which will be named Authentication Service. These two components will communicate together through the usual Request API and the server will decide if the user is the real user or not. For this, the server will store tokens in the database.

To further enhance the access management of users, another component which is named Authorization service will be instantiated in the server. This component will help the request handler so that the user request is denied or allowed access to the function, based on if the user has required access to that section. This component will also use the database to get permissions and to store and read user tokens to compare.

Another component will be the Encryption Wrapper in the server, which regulates access to the database, and makes sure that the data that is written to the database is always encrypted.

Lastly, there will be a Message Interceptor to understand if the requests are made very frequently. This will make sure that the availability is kept under control for normal users, and deny service for the suspected messages that come from the outside. This component will also be used to encrypt messages that come from the outside, so that the messaging system between the users and hosts are encrypted.

4.4.6 Step 6: Sketch Views and Record Design Decisions

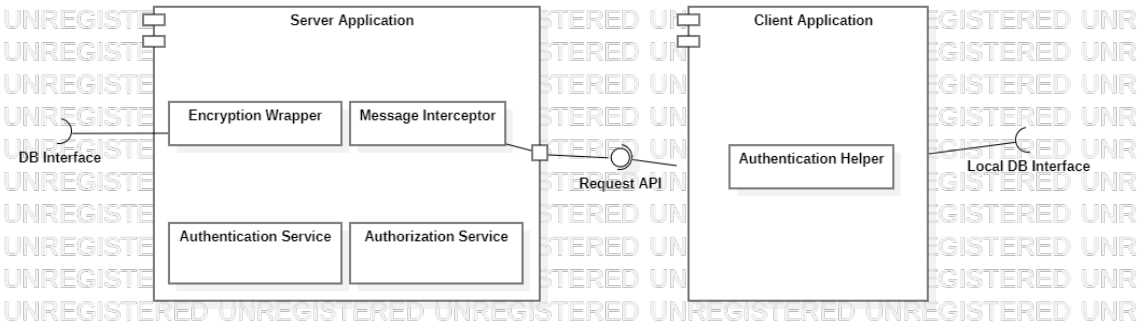

Figure 4.6: Component Diagram Sketch for Security

Here in this diagram you can see the components talked about in the previous step and their locations, both in the server and client applications.

4.4.7 Step 7: Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose

This iteration of the security accomplished its goals of having the system for authenticating, authorizing and keeping the data and messages secure in the server. Thus we can say that meeting the goals was successful and we also improved the availability aspect thanks to this iteration.

5 Architecture Design

5.1 Module View

5.1.1 Stakeholders' uses of this view

Customers: Customers indirectly benefit from a well-structured Module View. It leads to a system that is logically organized, resulting in efficient and user-friendly applications.
Hotel Providers (Hosts): Understanding the Module View can help hotel providers grasp how their service information is encapsulated within the system modules. This knowledge aids in effective listing management.
Transportation Partners: Transportation partners, by comprehending the Module View, can understand how their services are encapsulated and interact with other modules in the system.
Employees: Employees use the Module View to comprehend the system's modular structure. This understanding aids in feature implementation, issue debugging, performance optimization, and system maintenance.
Investors: Investors use the Module View to assess the system's modularity and logical organization. This helps them gauge the system's maintainability and extensibility, informing their investment decisions.

5.1.2 UML Diagrams for Module View

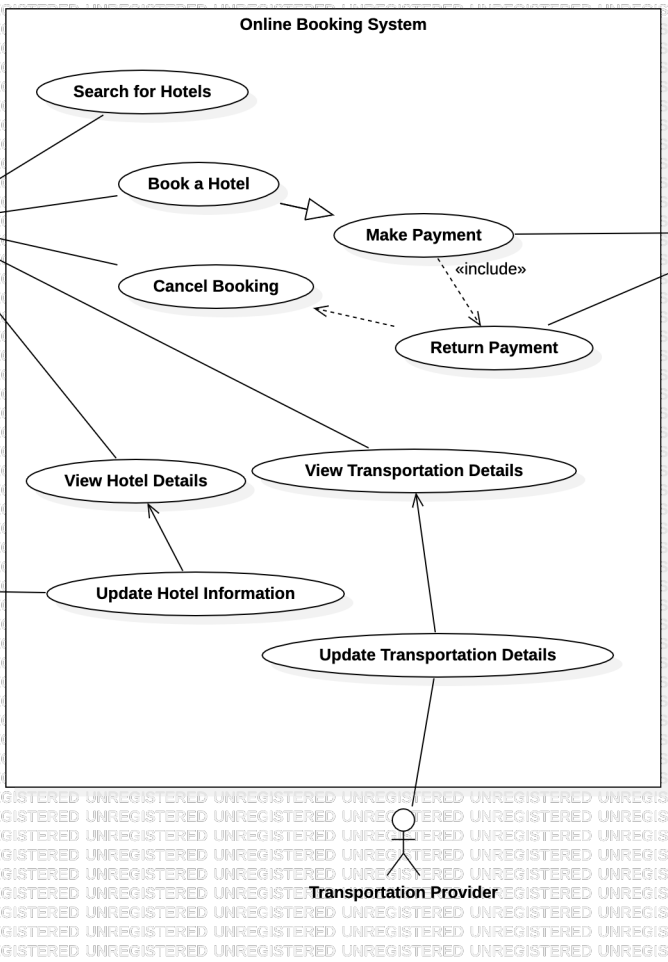

Figure 5.1: Use Case Diagram For Online Booking System

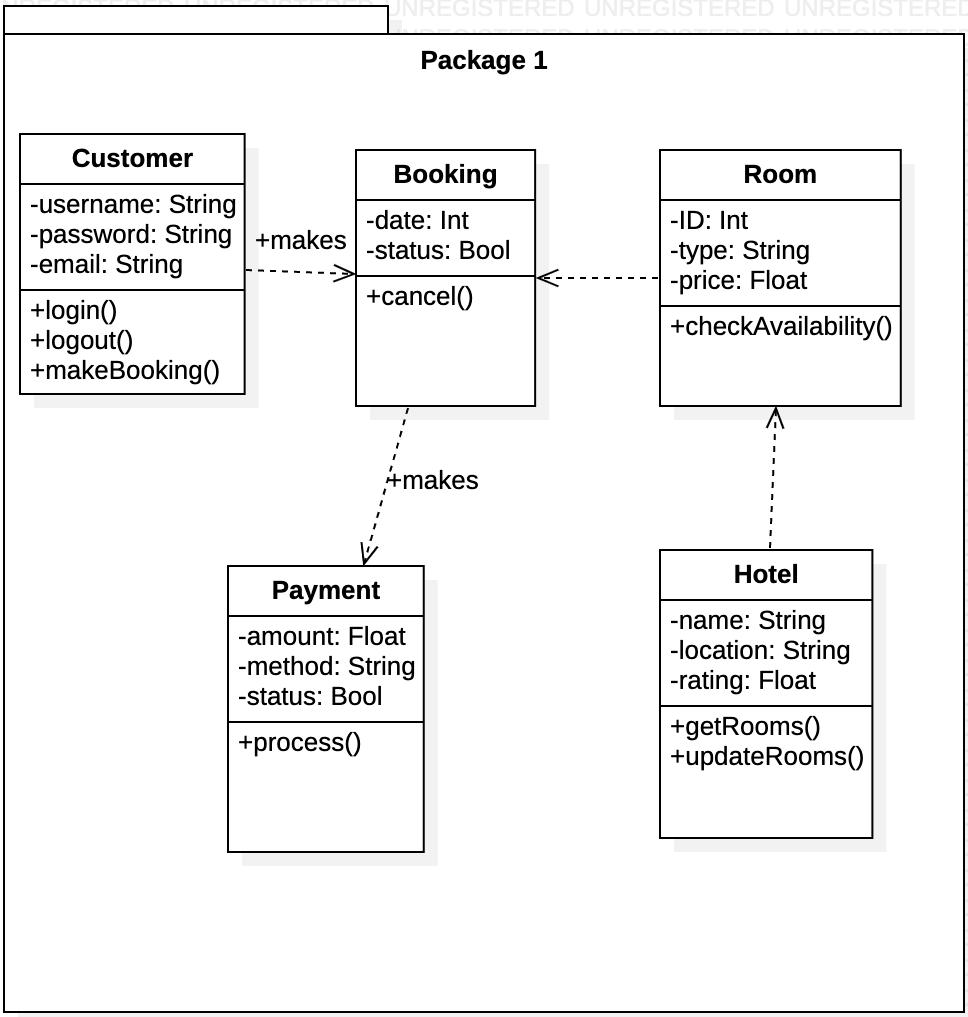

Figure 5.2: Class Diagram For Online Booking System

5.2 C&C View

5.2.1 Stakeholders' uses of this view

Customers: While customers may not directly interact with the view, it indirectly affects their experience. A well-structured system can lead to a more efficient and user-friendly application, enhancing the customer's experience.
Hotel Providers (Hosts): Hotel providers can benefit from understanding the view as it can help them understand how information about their services is accessed and updated in the system. This can help them manage their listings more effectively.
Transportation Partners: Similar to hotel providers, transportation partners can also benefit from understanding the view. It can help them understand how their services are integrated into the system and how information flow is managed.
Employees: Employees, especially those involved in system development, maintenance, and operations, can use the view to understand the system's structure and interactions. This can help them in implementing features, debugging issues, optimizing performance, and maintaining the system over time.
Investors: Investors may use the view to gain a high-level understanding of the technical robustness and scalability of the system. This can help them make informed decisions about their investments.

5.2.2 UML Components Diagram for C&C View

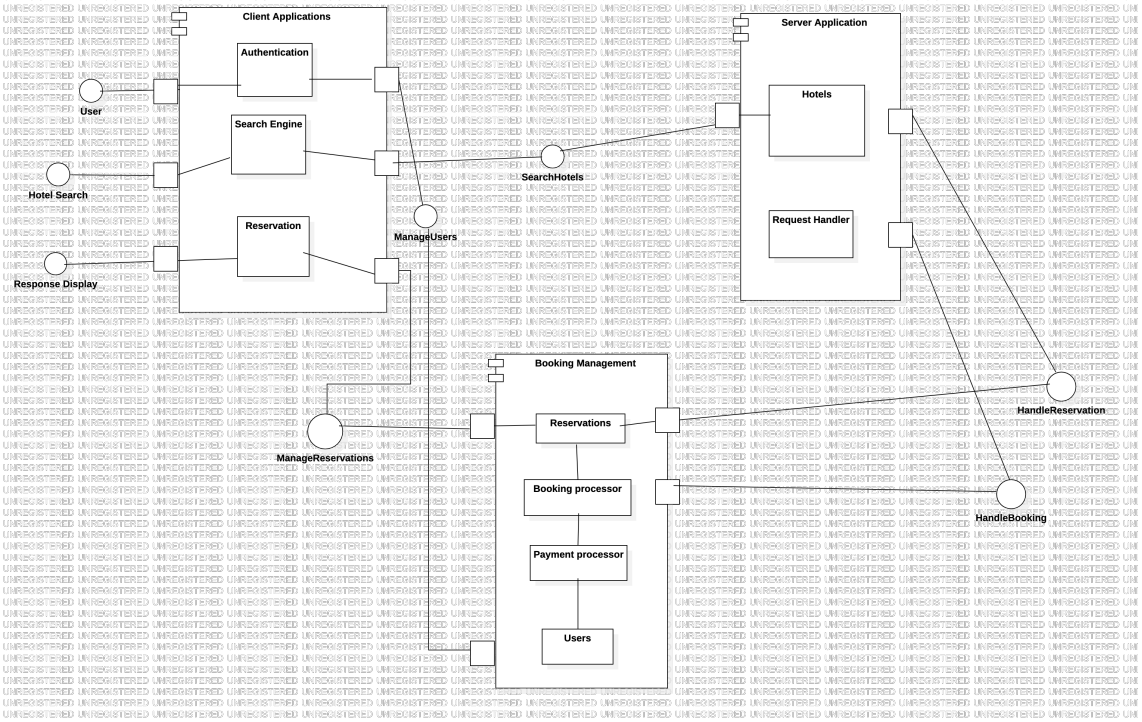

Figure 5.3: Component Diagram for CC View

5.3 Allocation View

5.3.1 Stakeholders' uses of this view

Customers: Customers interact with the system through the user interface, which could be a web application or a mobile app. The allocation view helps identify how the system is deployed to provide a seamless and efficient user experience.
Hotel Providers (Hosts): Hotel providers need to manage their listings and availability. The allocation view can help them understand how their information is stored and accessed in the database, and how it's presented to the customers.
Transportation Partners: These partners provide transportation services that are integrated with the booking system. The allocation view can help them understand how their services are accessed and used within the system.
Employees: Employees, especially those in technical roles, would use the allocation view to understand the deployment of the system. This includes understanding how different components are distributed across servers, how load balancing is achieved, and how data is managed.
Investors: Investors shall use the allocation view to assess the technical clarifications and scalability of the system. It can provide insights into the system's ability to handle growth and adapt to changes in demand.

5.3.2 Deployment Diagram for Allocation View

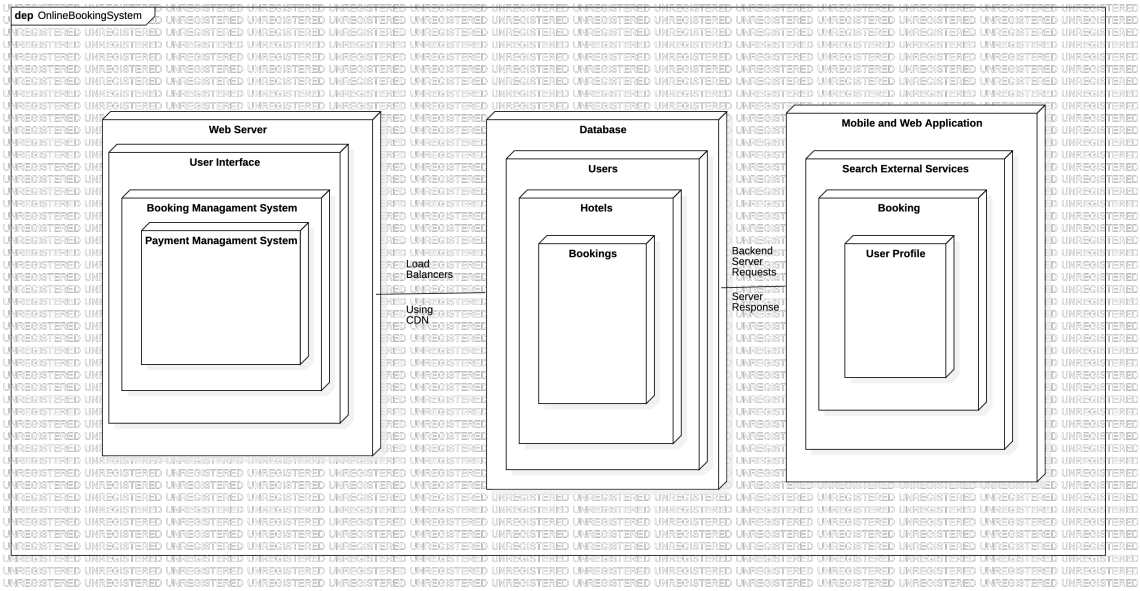

Figure 5.4: Deployment Diagram for Online Booking System Allocation View

5.4 Behavior

5.4.1 Stakeholders' uses of this view

Customers: Customers interact with the system's behavior through the user interface. While they may not directly use the behavior view, it indirectly affects their experience. A well-designed behavior view can lead to a more responsive and user-friendly application.
Hotel Providers (Hosts): Hotel providers can benefit from understanding the behavior view as it can help them understand how the system reacts to their actions, such as updating room availability or changing pricing.
Transportation Partners: These partners provide transportation services that are integrated with the booking system. The behavior view can help them understand how the system responds to updates or changes in their services.
Employees: Employees, especially those in technical roles, would use the behavior view to understand the dynamic behavior of the system. This includes understanding how different components interact, how data flows through the system, and how the system reacts to different events.
Investors: Investors might use the behavior view to assess the robustness and flexibility of the system. It can provide insights into the system's ability to handle different scenarios and adapt to changes in demand or behavior.

5.4.2 UML Diagrams for Behavior

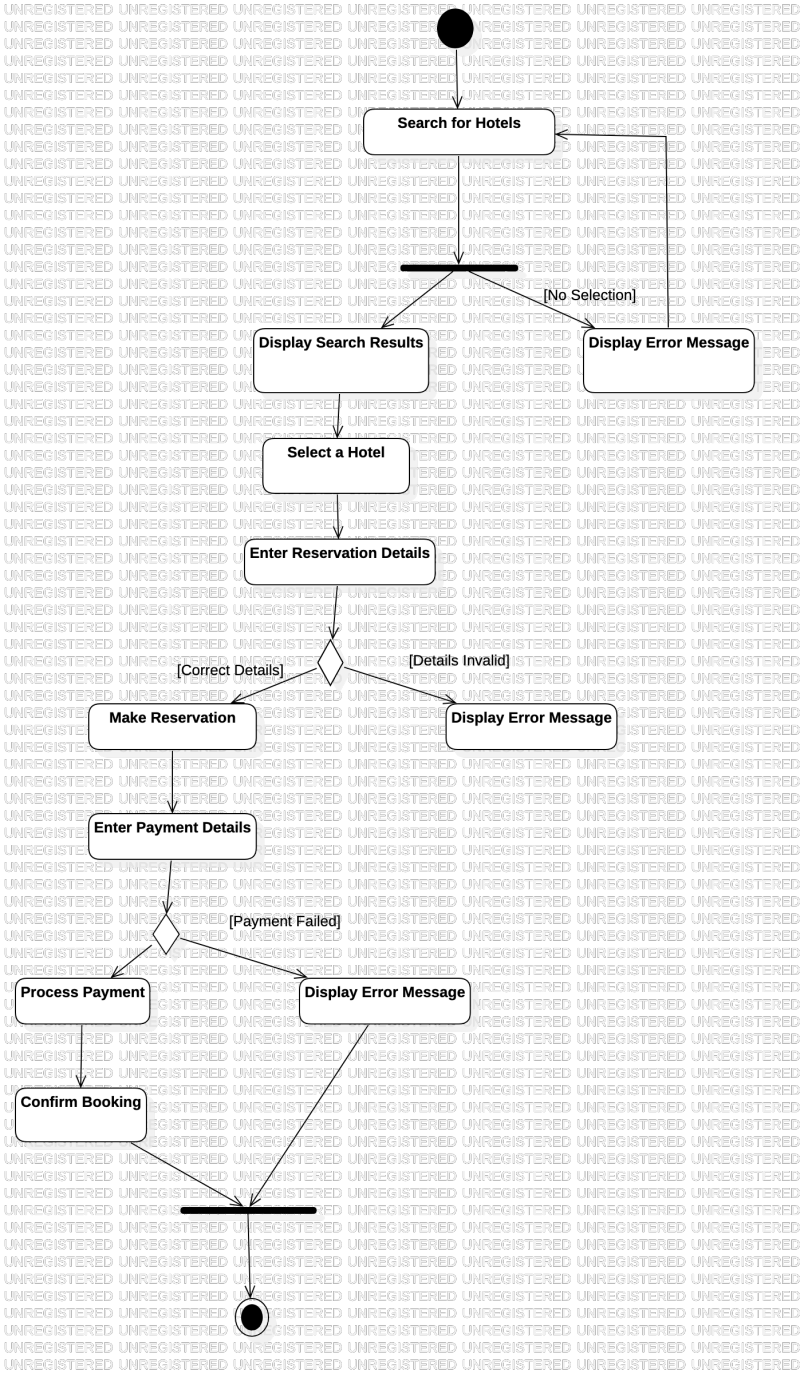

Figure 5.5: Activity Diagram to Represent the Most Architecturally Critical Function

5.5 Rationale

Design Decisions and Location	**Rationale and Assumptions (Include Discarded Alternatives)**
Resist to Attacks in Security	Instead of employing tactics for reacting and recovering, all the effort is made for resisting the attacks. This could mean that the further iterations may require to improve other aspects of the security quality attribute of the system.
Using MVC instead of Observer in Usability	Instead of using the observer pattern, MVC is prefered because we don't need to watch for model changes in multiple places at the same time. In other words, when an alternative view is preferred by the user, the old view will get detached and the new one will take its place. As stated in the book, the observer pattern is overkill in this sense. MVC satisfies our need to decouple the view and logic.
Effort in Usability to maintain alternate views	A great deal of effort has been contributed in accomplishing the possibility of alternate views in the usability section. However, based on market research, this may not be feasible because of budget limits, or at the very least it could not be a prioritization. However, we found it nice for the system to support this kind of accessibility features.
Clusters in Deployability and Availability	We decided to use already ready solutions such as Kubernetes in order not to solve the same problems in the software. Using dockerized environments, and containers and orchestration solutions, we made sure that the budget costs are kept safe, but it also means that the expertise required will be very high. However, we found that this is a reasonable tradeoff since the same problems will be encountered anyways.

Table 5.1: Example Table for Decisions Design Decisions