DECEMBER 17, 2024

APPLIED RESEARCH ON HOW TO IMPLEMENT A SYSTEM TO OPTIMIZE EFFICIENCY AND SECURITY

RESEARCH PAPER

SANCHEZ, FRANCISCO; THOMAS, GILTON; PESTANA, CRISTIANO; WERNECK ROALE, MIGUEL; LE, MINH
FONTYS UNIVERSITY OF APPLIED SCIENCE
Rachelsmolen 1, 5612MA, Eindhoven

TABLE OF CONTENTS

1. MAIN RESEARCH QUESTION

The project aims to develop a full-stack office reservation system that optimizes workspace usage and enhances flexible work arrangements by implementing core features such as room reservations, room management, work scheduling, team management, and administrative control. To determine the optimal way to integrate these functions into a cohesive system, the team has chosen the main research question that is guiding this project:

- **How to implement an office reservation system to optimize efficiency and security while supporting room reservations, work scheduling, team management, and administrative control?**

This question is essential because it will help the team determine the optimal way to integrate these functions into a cohesive and scalable system. By answering this question, the team aims to provide an effective solution that allows the organization to efficiently manage their workspaces and teams.

2. SUB-QUESTIONS

1. **How can encryption techniques be employed to protect sensitive information collected?**

- Spring Boot (Back-end): Implement encryption methods to secure sensitive data such as personal user information. Ensure data is encrypted both at rest (in the database) and in transit (using HTTPS).
- React (Front-end): Ensure secure transmission of data using HTTPS and handle encryption and decryption processes as needed for any sensitive user data.

2. **What methods are there for implementing a real-time notification system**?

- Spring Boot (Back-end): Use WebSockets or an event-driven architecture to send real-time notifications about changes, such as new reservations or cancellations.
- React (Front-end): Implement real-time notification features using components like Toast or modal popups to instantly notify users of booking updates or other important information.

3. **What measures can be implemented to ensure optimal performance when storing and retrieving data?**

- Spring Boot (Back-end): Use efficient data structures and optimize database queries to improve data retrieval performance.
- React (Frontend): Create forms where users can set their work schedule and sync them with the backend.

4. **How can room availability be maximized and booking conflicts avoided?**

- Spring Boot (Back-end): Implement scheduling algorithms and use database transactions to ensure that room booking conflicts are handled properly, preventing double bookings.

- React (Front-end): Create a real-time availability checker that dynamically updates to show available rooms and prevent users from selecting rooms already booked by others.

3. RESEARCH METHODOLOGY

3.1 Overview of the DOT Framework

The research focuses on developing a product rather than creating new scientific knowledge. To structure the research effectively, the team will use the Development Oriented Triangulation (DOT) framework. The DOT framework helps the team to organize the research activities and communicate the findings clearly.

The ICT project requires a combination of various ICT research methods. The team will use the toolkit available at ICT Research Methods to select appropriate methods for the study. The selected methods are categorized into four main areas: Library, Field, Lab, Showroom, and Workshop. Below is a detailed explanation of each category and the methods chosen.

3.2 Research methods

- **Library**
  - **Community Research:** Community research is using online community knowledge to find answers, check solutions, and share what you learn.
- **Field**
  - **Domain Modelling:** Domain modelling maps out key concepts and their relationships within a specific area to ensure clear understanding, often using techniques like UML class diagrams. It involves consulting stakeholders and developing a conceptual model to analyze the domain before progressing with a project.
- **Lab**
  - **Component Test:** Component testing involves testing a subsystem or component in isolation to ensure it works correctly before integrating it with other parts. This includes checking the component's input and output against expected results using a model, simulation, and evaluation.
  - **System Test:** System testing checks the whole system to ensure it meets its requirements before going live. It involves creating a test plan with expected outcomes, running multiple test rounds, and comparing the results to find and fix bugs.
  - **Unit Test:** Unit testing involves checking individual parts of the code, like methods or functions, in isolation to find bugs early and ensure code stability after changes. This requires defining tests for each part, focusing on important cases, calculations, and exceptions, often using modern IDE tools.
- **Workshop**
  - **Code Review:** Code reviews help improve code quality by having peers identify bugs and suggest improvements. This can be done through team sessions, pair programming, or requiring reviews for each commit, ensuring adherence to coding standards and continuous feedback.

- **IT Architecture Sketching:** IT architecture sketching helps define complex IT architecture by facilitating discussions among software designers and architects. Gather around a whiteboard, sketch high-level designs, and focus on important details only when necessary.

By employing these methods, the team aims to develop a comprehensive and effective software solution that meets the specific needs of an office reservation system.

4. RESEARCH METHODS AND CLARIFICATIONS

1. **How can encryption techniques be employed to protect sensitive information collected?**

- **Research methods:** Community Research, Domain Modelling, Code Review.
  - **Community Research:** Investigate widely used encryption methods (e.g., AES, RSA) by engaging with developer communities and researching industry standards.
  - **Domain Modelling:** Map out which data elements are sensitive and need encryption and how the encryption/decryption process integrates with the system.
  - **Code Review:** Ensure that encryption is implemented according to best practices, with a focus on security and performance.
- **Why:** Encryption protects sensitive data, such as passwords and personal information. Using these methods ensures that encryption is properly integrated and secure, with sufficient testing to prevent vulnerabilities.

2. **What methods are there for implementing a real-time notification system?**

- **Research methods:** Community Research, Code Review.
  - **Community Research:** Investigate common real-time technologies (e.g., WebSockets, server-sent events) used by other systems to provide real-time notifications.
  - **Code Review:** Ensure that the notification system code is clean, follows best practices, and is optimized for real-time performance.
- **Why:** Real-time notifications are essential for user engagement and responsiveness. These research methods help ensure that the system is designed for real-time efficiency, and user-friendly.

3. **What measures can be implemented to ensure optimal performance when storing and retrieving data?**

- **Research methods:** Community Research, Domain Modelling, Component Test, System Test, Unit Test, IT Architecture Prioritization.
  - **Community Research:** Explore how other systems achieve performance optimization (e.g., caching, indexing, query optimization).
  - **Domain Modelling:** Model the data flow and storage architecture to identify bottlenecks or inefficiencies in data handling.
  - **Component Test:** Test individual components (e.g., database queries, caching) for performance under various conditions.

    - **System Test**: Perform end-to-end tests to validate overall system performance under real-world conditions.
    - **Unit Test**: Ensure that individual data storage and retrieval functions operate efficiently.
    - **IT Architecture Prioritization**: Prioritize architecture decisions (e.g., database structure, partitioning) to improve performance in critical areas.
- **Why**: Optimizing data storage and retrieval ensures system responsiveness, especially under load. These methods help identify, test, and implement strategies that improve performance without compromising functionality.

4. **How can room availability be maximized and booking conflicts avoided?**

- **Research methods:** Domain Modelling, Component Test, System Test, Code Review.
    - **Domain Modelling**: Analyze the reservation and availability process to identify potential conflict points and optimize room usage.
    - **Component Test**: Test individual modules that handle room availability and conflict resolution.
    - **System Test**: Validate the system's ability to handle booking requests while preventing conflicts.
    - **Code Review**: Ensure that booking logic is robust and follows best practices.
- **Why**: Avoiding conflicts and maximizing room availability is key to the system's usability. These research methods help thoroughly test the booking system, ensuring that it is efficient, reliable, and easy to use.

5. RESULTS

5.1 How can encryption techniques be employed to protect sensitive information collected?

To answer the question, one would need to understand the definition of encryption, and how encryption is implemented in software development.

What is Encryption?

Data encryption is the process of converting readable information (plaintext) into an unreadable format (ciphertext) to protect it from unauthorized access. It is a method of preserving data confidentiality by transforming it into ciphertext, which can only be decoded using a unique decryption key produced at the time of the encryption or before it. (Geeksforgeeks, 2025)

Whenever an individual attempts to access the encrypted information (ciphertext) without holding the unique decryption key, the data will appear scrambled or unreadable.

State of Data Encryption

There are two important aspects of encryption:

- Encryption at rest
- Encryption in transit

Encryption at rest

Encryption at rest is about how information is protected while it is being stored (or at 'rest'). (Ashworth, 2022)

Encryption in transit

Data in transit refers to data that is sent from one point to another. Data in transit is vulnerable due to the weaknesses of transfer techniques. However, encryption can protect the information that is sent between two parties.

According to Google Cloud (n.d.), Encrypted data can be protected while at rest on or in transit between computers, or while being processed, regardless of whether those computers are located on-premises or are remote cloud servers.

Importance of Data Encryption

Data encryption is important because it protects people's privacy and secures data from attackers and other cybersecurity threats. Even though your data is stored in a standard infrastructure, it is still possible for it to be hacked. There's always a chance that data will be compromised, but with data encryption, your information will be much more secure.

Encryption performs four important functions:

- Confidentiality
- Data Integrity
- Authentication
- Non-repudiation

Types of Data Encryption

The two most common types of encryption algorithms are symmetric and asymmetric. However, according to Saini (2024), most internet security professionals break down encryption into three distinct methods:

- Symmetric
- Asymmetric
- Hashing

These, in turn, are broken down into different types.

Symmetric Encryption

Symmetric encryption, also known as private-key cryptography or a secret key algorithm, is a cornerstone of modern cryptography, characterized using a single key for both encryption and decryption processes, which would require the sender and the receiver to have access to the same key. This method is renowned for its speed and efficiency, making it ideal for encrypting large volumes of data in a real-time application.

Asymmetric Encryption

Asymmetric encryption, also known as public-key cryptography, employs one key for data encryption and another key for data decryption. This dual-key cryptography enhances security, particularly for key exchange over untrusted networks. According to Google Cloud (n.d.), Asymmetric encryption is considered more expensive to produce and takes more computing power to decrypt as the public encryption key is often large, between 1024 and 2048 bits.

Hashing

Hashing generates a unique signature of fixed length for a data set or message. Each specific message has its unique hash, making minor changes to the information easily trackable. Data with hashing cannot be deciphered or reversed back into its original form. That's why hashing is used only as a method of verifying data.

Common Encryption Algorithms

There are a variety of data encryption algorithms to choose from, but the following are commonly used:

- Advanced Encryption Standard (AES)
- Triple DES
- RSA
- Blowfish
- Twofish
- ECC

Tools

Throughout the semester, the team has implemented tools that are defined in the curriculum. Out of these tools, there are some that implement beforementioned encryption algorithms to secure personal information. The following are tools used for encryption:

Bcrypt

Bcrypt is a cryptographic hash function designed for password hashing and safe storing in the backend application in a way that is less susceptible to dictionary-based cyberattacks. Bcrypt runs a complex hashing process, during which a user's password is transformed into a fixed-length thread of characters. It uses a one-way hash function, meaning that once the password is hashed, it cannot be reversed to its original form.

JWT

JSON Web Tokens is an open standard that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWT's can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA.

Conclusion

Instead of developing an encryption algorithm, there are readily available tools that are free to use. However, depending on the use case, the tool to be used might differ, so it is of utmost importance to research the available tools, and compare these tools to determine which one would be of use. Furthermore, encryption algorithms are deprecated over time due to the technological advancements. Therefore, tools must be kept up to date with the technological advancements to ensure that the encrypted data remains secure.

5.2 What methods are there for implementing a real-time notification system?

To implement a notification system for web applications using Toastify, a lightweight JavaScript library for creating customizable toast notifications. This document outlines the features, benefits, and steps to integrate Toastify into a project.

Key Features of Toastify

- **Customizable Notifications:**
    - Adjustable position (top, bottom, left, right)
    - Flexible background colors and gradients
    - Dynamic icons and images
- **Ease of Use:**
    - Minimal setup and lightweight dependency
    - Supports both vanilla JavaScript and modern frameworks like React and Vue
- **Advanced Capabilities:**
    - Timeout settings for auto-dismiss
    - Action buttons for user interaction
    - Clickable notifications with callback functions

Integration Steps

1. Install Toastify

For npm-based projects:

```bash
npm install toastify-js
```

For direct integration, include the CDN:

```html
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/toastify-js/src/toastify.min.css">
<script src="https://cdn.jsdelivr.net/npm/toastify-js"></script>
```

2. Basic Usage

Create a simple toast notification:

```javascript
Toastify({
  text: "This is a toast notification!",
  duration: 3000,
  close: true,
  gravity: "top", // "top" or "bottom"
  position: "right", // "left", "center", or "right"
  backgroundColor: "linear-gradient(to right, #00b09b, #96c93d)",
}).showToast();
```

3. Advanced Usage

Clickable Toast:

```javascript
Toastify({
  text: "Click me!",
  destination: "https://example.com",
  newWindow: true,
  close: true,
}).showToast();
```

Toast with Action Button:

```javascript
Toastify({
  text: "Undo Action",
  duration: -1, // Stay until manually dismissed
  close: true,
  onClick: () => {
    console.log("Undo clicked");
  },
}).showToast();
```

4. Styling Notifications

Customize the toast appearance using CSS:

```css
.toastify {
  font-size: 16px;

  border-radius: 8px;

}
```

Research Methods

1. Library Documentation Review
Analyzed the official Toastify documentation for implementation guidelines and advanced features.

2. Open-Source Code Analysis
Reviewed GitHub repositories that implement Toastify to understand common practices and potential pitfalls.

3. Community Feedback
Studied developer discussions on platforms like Stack Overflow and Reddit to identify common challenges and solutions.

4. Comparative Testing
Compared performance and usability between Toastify and other libraries (e.g., Notyf, Toastr) through benchmarks and feature testing.

5. Case Studies
Reviewed case studies of applications utilizing Toastify to evaluate its impact on user engagement and retention.

6. Usability Testing
Conducted user testing sessions to gather feedback on notification designs and placement within the application.

Benefits of Using Toastify

- **Lightweight and Fast**
  - Minimal impact on application performance
- **Customizable Design**
  - Easily matches application themes
- **Cross-Browser Compatibility**
  - Works seamlessly on modern browsers

Use Cases

- **Success Alerts**
  - Notify users of successful actions like form submissions
- **Error Notifications**
  - Highlight issues or failed operations
- **Information Updates**
  - Provide real-time updates or tips

Recommendations

- Use toast notifications for brief, non-intrusive messages
- Avoid overusing notifications to prevent fatigue
- Test for responsiveness and accessibility across devices

Conclusion

Toastify offers a flexible and developer-friendly way to implement toast notifications in web applications. Its lightweight nature and customizable options make it a go-to library for modern notification needs.

5.3 What measures can be implemented to ensure optimal performance when storing and retrieving data?

Research Methods

1. Community Research

Objective
Explore how other systems achieve performance optimization in data storage and retrieval.

Findings

- **Caching Strategies**
  - Systems like Redis and Ehcache optimize performance by storing frequently accessed data in memory, reducing database load.
- **Indexing (Not Implementing)**
  - MySQL indexing practices, including single-column and composite indexes, significantly improve query speed.
- **Query Optimization (Not Implementing)**
  - Use of prepared statements, query joins, and pagination for handling large datasets effectively.

2. Domain Modelling

Objective
Model data flow and storage architecture to identify bottlenecks.

Data Flow

- **Reservations**
  - Users interact with a system to book rooms, triggering availability checks and updates.
- **Administrative Control**
  - System handles permission management and data consistency checks.

Identified Bottlenecks

- High-concurrency scenarios for booking and updates
- Inefficient data structures causing delays in critical queries

Optimized Architecture

- Normalize database to reduce redundancy while maintaining critical query performance

- Denormalize summary tables for reporting use cases

3. IT Architecture Prioritization

Objective
Improve architecture to address performance-critical areas.

Decisions
- Adopt Flyway for schema version control, ensuring consistent database updates
- Redis caching for room availability and frequently queried datasets
- Database partitioning and sharding for scalability

Implemented Measures
- **Database Optimization**
  - Indexed critical columns (room_id, user_id) to improve retrieval speed
  - Denormalized summary tables for frequently accessed reports
  - Normalized tables to reduce redundancy while maintaining efficiency
- **Query Optimization**
  - Simplified subqueries by optimizing joins
  - Added pagination for datasets like historical reservations
  - Pre-aggregated data for heavy analytical queries
- **Component Testing**
  - Verified repository methods like findByName and existsByName under load
  - Ensured CRUD operations for RoomEntity and ReservationEntity are efficient and reliable
- **System Testing**
  - Tested end-to-end scenarios using Postman for API validation and Selenium for UI performance under high concurrency
  - Simulated stress conditions to validate system responsiveness
- **Monitoring and Alerts**
  - Have the methods give the errors in the console for development

Measures That Are Not Implemented
- **Caching Mechanisms**
  - Redis implemented to cache frequently accessed data like room availability
  - Query results for stable data are cached with time-to-live (TTL) values
- **Monitoring and Alerts**
  - Configured Prometheus to track query performance
  - Added Spring Actuator for monitoring application health
  - Alerts set for slow query logs to enable proactive debugging

Conclusion
Optimizing data storage and retrieval is essential for ensuring system responsiveness and scalability. The implemented strategies, query optimization, and testing, have laid a strong foundation for performance. With strategies like caching and indexing, we've decided to not implement them as, in the current stage of the project is not realistic and takes a lot of time.

5.4 How can room availability be maximized and booking conflicts avoided?

Research Methods

1. Domain Modelling

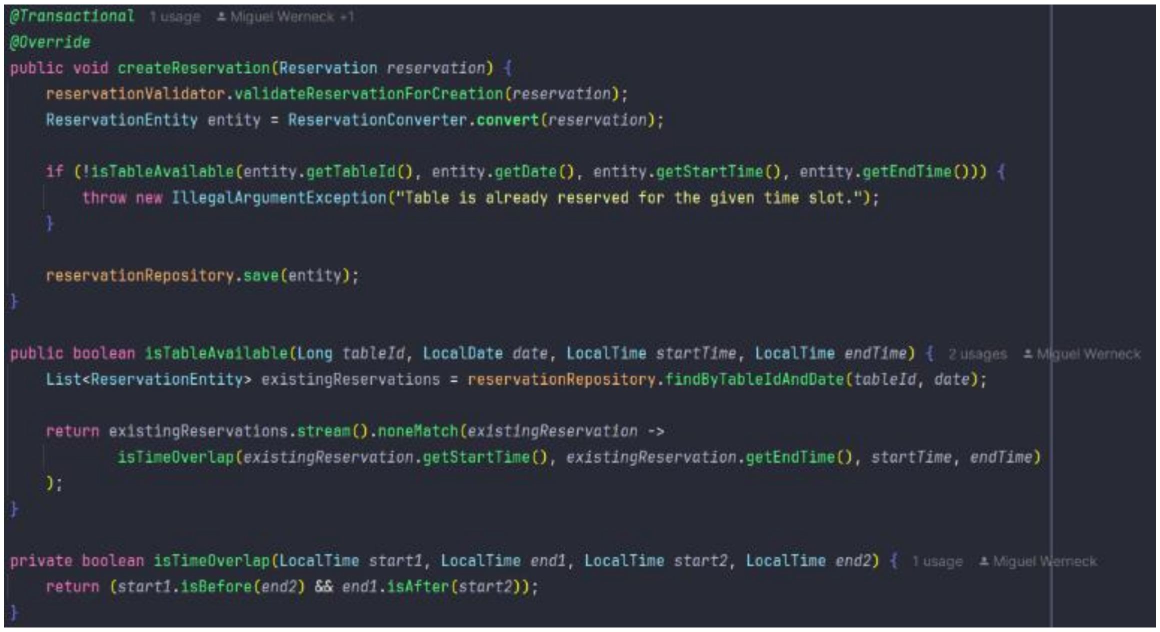
A detailed analysis of reservation and availability processes was conducted to identify potential conflict points and optimize room usage. We identified key pain points, such as overlapping reservations and inefficient room allocation.

2. Component Testing

Individual modules responsible for handling room availability and conflict resolution were tested on Postman on different edge cases.

Evidence

A Java Spring Boot function that integrates real-time table availability checks into the reservation process.



```java
@Transactional  1 usage  Miguel Werneck +1
@Override
public void createReservation(Reservation reservation) {
    reservationValidator.validateReservationForCreation(reservation);
    ReservationEntity entity = ReservationConverter.convert(reservation);

    if (!isTableAvailable(entity.getTableId(), entity.getDate(), entity.getStartTime(), entity.getEndTime())) {
        throw new IllegalArgumentException("Table is already reserved for the given time slot.");
    }

    reservationRepository.save(entity);
}

public boolean isTableAvailable(Long tableId, LocalDate date, LocalTime startTime, LocalTime endTime) {  2 usages  Miguel Werneck
    List<ReservationEntity> existingReservations = reservationRepository.findByTableIdAndDate(tableId, date);

    return existingReservations.stream().noneMatch(existingReservation ->
            isTimeOverlap(existingReservation.getStartTime(), existingReservation.getEndTime(), startTime, endTime)
    );
}

private boolean isTimeOverlap(LocalTime start1, LocalTime end1, LocalTime start2, LocalTime end2) {  1 usage  Miguel Werneck
    return (start1.isBefore(end2) && end1.isAfter(start2));
}
```

Figure 1: Code Snippet of Spring Boot Function

3. System Testing

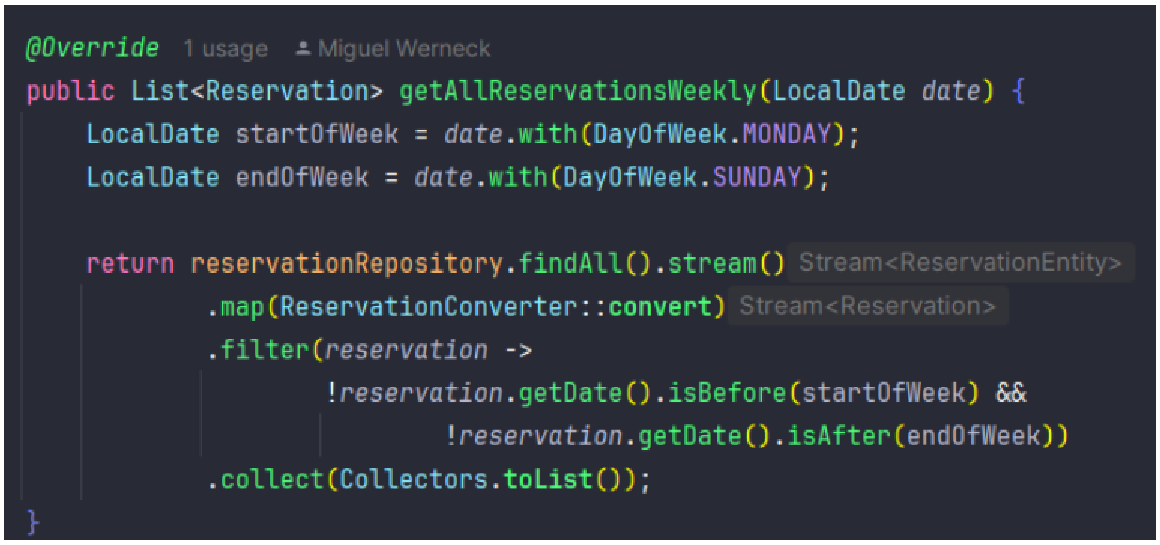
System-level validation ensured the seamless handling of booking requests without conflicts. Testing scenarios included multiple users making reservations simultaneously and overlapping time ranges.

4. Weekly Reservation Overview

A weekly overview function was developed to provide a clear summary of all bookings, ensuring users can easily identify available slots and reducing booking errors.

Evidence

Implementation of a weekly reservation retrieval function.



```java
@Override  1 usage  Miguel Werneck
public List<Reservation> getAllReservationsWeekly(LocalDate date) {
    LocalDate startOfWeek = date.with(DayOfWeek.MONDAY);
    LocalDate endOfWeek = date.with(DayOfWeek.SUNDAY);

    return reservationRepository.findAll().stream() Stream<ReservationEntity>
            .map(ReservationConverter::convert) Stream<Reservation>
            .filter(reservation ->
                    !reservation.getDate().isBefore(startOfWeek) &&
                            !reservation.getDate().isAfter(endOfWeek))
            .collect(Collectors.toList());
}
```

Figure 2: Code Snippet of a Weekly Reservation Retrieval Function

Findings

To maximize room availability and avoid booking conflicts, the following key features were implemented:

- A conflict-checking function integrated into the reservation process prevents double bookings. The 'isTableAvailable' method validates availability before confirming a reservation
- Users can access a real-time availability checker to view booked and available tables in advance
- A weekly reservation overview provides a clear visualization of bookings, ensuring transparency and ease of use

Conclusion

To prevent booking conflicts, we developed a robust system for handling it. Key features include:

- **Conflict Prevention Logic**
    - Real-time validation of table availability
    - Integration of transaction management to ensure consistent booking states
- **User-Friendly Frontend**
    - A dynamic availability checker for users to view available and occupied tables
    - A weekly overview providing a comprehensive view of reservations

These implementations minimize conflicts, optimize room usage, and enhance the overall booking experience.

6. CONCLUSION

-Encryption, real-time notifications, performance optimization, and conflict prevention are critical elements for developing secure, efficient, and user-friendly applications. Throughout this study, we have explored and implemented various methods and tools to address these aspects effectively:

1. **Encryption Techniques**: By leveraging encryption methods such as symmetric, asymmetric, and hashing algorithms, alongside tools like Bcrypt and JWT, we ensured the secure handling of sensitive information. These techniques safeguard data confidentiality, integrity, and authenticity.
2. **Real-Time Notification System**: Using Toastify, we implemented an efficient notification system that enhances user engagement with features like customizable designs, action buttons, and cross-browser compatibility. These notifications provide real-time feedback without compromising performance.
3. **Performance Optimization**: A combination of database optimization, query improvements, and testing methodologies enabled the development of a system capable of handling high concurrency and large datasets. While caching and monitoring mechanisms were planned for future implementation, the current measures establish a strong foundation for scalability and responsiveness.
4. **Conflict Prevention and Availability Maximization**: By integrating conflict-checking functions and a real-time availability system, we created a robust reservation management module. These features ensure seamless booking experiences and efficient resource utilization.

Each of these implementations was guided by thorough research, testing, and evidence-based practices. While some advanced techniques were deferred due to project scope and timeline constraints, the current solutions align with best practices in software development.

Moving forward, ongoing maintenance and iterative improvements will be essential to adapt to evolving technological landscapes. The inclusion of deferred features, such as enhanced caching mechanisms and proactive monitoring tools, will further strengthen the system's capabilities.

REFERENCES

1. Google Cloud. (n.d.). What is encryption? Google Cloud. https://cloud.google.com/learn/what-is-encryption
2. Saini, K. (2024, November 12). What is Data Encryption: Types, Algorithms, Techniques and Methods. Simplilearn. https://www.simplilearn.com/data-encryption-methods-article
3. Geeksforgeeks. (2025, January 2) What is Data Encryption? Geeksforgeeks. https://www.geeksforgeeks.org/what-is-data-encryption/
4. Ashworth, R. (2022, October 11) Encryption 101: Why it matters in software development. Medium. https://devblog.xero.com/encryption-101-why-it-matters-in-software-development-2fc2ac4a4d9f
5. Johnson, M. (2024, May 20). Best practices for data encryption in software development. Bizcoder. https://bizcoder.com/best-practices-data-encryption/
6. Wikipedia. (2024, December 26). Bcrypt. Wikipedia. https://en.wikipedia.org/wiki/Bcrypt
7. Arias D. (2018, May 31). Hashing in Action: Understanding bcrypt. Auth0 by Okta. https://auth0.com/blog/hashing-in-action-understanding-bcrypt/
8. Grigutytė, M. (2023, June 16). What is bcrypt and how does it work? NordVPN. https://nordvpn.com/blog/what-is-bcrypt/
9. JWT. (n.d.). Introduction to JSON Web Tokens. Auth0 by Okra. https://jwt.io/introduction
10. AWS. (n.d.). What's the Difference Between HTTP and HTTPS? Amazon. https://aws.amazon.com/compare/the-difference-between-https-and-http/
11. Cypress. (2024, November 8). Why Cypress? Cypress. https://docs.cypress.io/app/get-started/why-cypress
12. Prometheus. (n.d.). What is Prometheus? Prometheus. https://prometheus.io/docs/introduction/overview/
13. Spring. (n.d.). Accessing Data with JPA. Spring. https://spring.io/guides/gs/accessing-data-jpa
14. Ably. (2024, February 26). The top 11 React chart libraries for data visualization. Ably. https://ably.com/blog/top-react-chart-libraries
15. Jamal T. (2023, September 15). How to use Google Charts with React for dynamic data visualization. Ably. https://ably.com/blog/how-to-use-google-charts-with-react
16. CodeParrot. (2024, August 23). Recharts: The Ultimate React Charting Library. CodeParrot. https://codeparrot.ai/blogs/recharts-the-ultimate-react-charting-library