When including information related to users in your bachelor thesis, you may want to consider covering various aspects. Here's a comprehensive outline of topics that you can explore regarding users:

1. Introduction:
   * Briefly explain the importance of user management in software systems.
   * Define the scope and objectives of your thesis regarding user-related aspects.
2. User Management System:
   * Provide an overview of user management systems and their role in software applications.
   * Explain the benefits of a robust user management system, such as authentication, authorization, and user data management.
3. User Models and Schema:
   * Describe the concept of user models and their representation in the database schema.
   * Discuss different user attributes such as name, email, password, role, tokens, etc.
   * Explain the rationale behind the chosen attributes and their significance in the application.
4. User Registration and Authentication:
   * Discuss the process of user registration, including capturing user details and validating inputs.
   * Explain the authentication mechanisms used to verify user identities, such as password hashing and token-based authentication.
   * Highlight the importance of secure authentication to protect user accounts.
5. User Roles and Permissions:
   * Explain the concept of user roles and their significance in access control.
   * Discuss different user roles and their associated permissions, such as admin, publisher, or regular user.
   * Describe how role-based access control (RBAC) is implemented to restrict user access to specific features or resources.
6. User Profile Management:
   * Discuss how users can manage their profiles, update personal information, and change passwords.
   * Explain the implementation of profile update functionality, including form validation and data persistence.
7. User Interactions and Activities:
   * Discuss user interactions within the system, such as product purchases, reviews, or social interactions.
   * Explain how user activities are tracked and stored, including capturing timestamps and associating activities with user accounts.
8. Token Management:
   * Describe the use of tokens in the application, such as for payments or rewards.
   * Explain how tokens are assigned, deducted, or accumulated based on user actions or transactions.
   * Discuss the role of tokens in facilitating user transactions and incentivizing user engagement.
9. User Interface and User Experience (UI/UX):
   * Discuss the design considerations for user interfaces, including usability and accessibility.
   * Explain how user experience (UX) principles are applied to create intuitive and user-friendly interfaces.
10. Privacy and Data Protection:
    * Discuss the importance of user privacy and data protection in the application.
    * Explain the measures taken to secure user data, such as encryption, secure storage, and compliance with data protection regulations.
11. Testing and Validation:
    * Explain the testing strategies and methodologies used to validate user-related functionalities.
    * Discuss the techniques employed to ensure the security and reliability of user management features.
12. Conclusion:
    * Summarize the key findings and contributions of your thesis related to user management.
    * Reflect on the challenges encountered and lessons learned during the development and evaluation of user-related functionalities.
    * Discuss potential future enhancements or research directions in user management systems.

In addition to the user-related aspects, if your bachelor thesis also covers the handling of 3D models in OBJ format and their storage on AWS S3, you can consider including the following points:

1. Introduction:
   * Introduce the concept of 3D models and their relevance in software applications.
   * Explain the need for efficient storage and management of 3D models.
2. OBJ Format:
   * Describe the OBJ file format commonly used for representing 3D models.
   * Explain the structure and components of the OBJ format, such as vertices, faces, textures, and materials.
3. AWS S3:
   * Provide an overview of Amazon Simple Storage Service (S3) and its features.
   * Explain the benefits of using AWS S3 for storing and serving 3D models, such as scalability, durability, and availability.
4. Uploading 3D Models to AWS S3:
   * Describe the process of uploading 3D models in OBJ format to AWS S3.
   * Discuss the techniques and APIs used to interact with the S3 service for file upload.
5. File Management and Organization:
   * Discuss strategies for organizing and managing 3D model files stored on AWS S3.
   * Explain how metadata, such as model names, categories, or tags, can be associated with the files for efficient retrieval.
6. Access Control and Permissions:
   * Explain the mechanisms for controlling access to 3D model files stored on AWS S3.
   * Discuss the implementation of access control policies to restrict or grant access to specific users or user groups.
7. Integration with User Management:
   * Discuss how the user management system you described earlier can be integrated with the storage and retrieval of 3D models on AWS S3.
   * Explain how user authentication and authorization can be used to control access to the uploaded 3D models.
8. Retrieval and Rendering of 3D Models:
   * Explain how the stored 3D models can be retrieved from AWS S3 for rendering or display in the application.
   * Discuss the techniques and libraries used to parse and render OBJ files on the client-side.
9. Performance Considerations:
   * Discuss performance optimizations when working with 3D models stored on AWS S3, such as caching, compression, or streaming techniques.
   * Explain how the delivery of 3D models can be optimized to ensure a smooth user experience.
10. Security and Encryption:
    * Discuss the security measures taken to protect the integrity and confidentiality of 3D model files stored on AWS S3.
    * Explain the use of encryption mechanisms, access control, and other security best practices.
11. Testing and Validation:
    * Describe the testing strategies employed to validate the upload, retrieval, and rendering of 3D models.
    * Discuss the techniques used to ensure the correctness and performance of the system.
12. Conclusion:
    * Summarize the findings and contributions related to the handling and storage of 3D models in OBJ format on AWS S3.
    * Reflect on the challenges faced and lessons learned during the implementation and evaluation of the system.
    * Discuss potential future enhancements or research directions in the context of handling 3D models in cloud storage.

Adapt and expand on these topics based on your specific requirements and the scope of your bachelor thesis.

If your bachelor thesis includes the management of orders through a queue system for mailer functionality, you can consider including the following points:

1. Introduction:
   * Introduce the concept of order management and the importance of efficient communication with users via email.
   * Explain the need for a queue system to handle mailer functionality effectively.
2. Order Management:
   * Describe the process of managing orders within the system, including order creation, updates, and fulfillment.
   * Explain the significance of order information, such as order ID, customer details, product details, and order status.
3. Mailer Functionality:
   * Discuss the purpose of the mailer functionality in the application, which includes sending order-related emails to customers.
   * Explain the types of emails sent, such as order confirmation, shipping updates, or order cancellation notifications.
4. Queue System:
   * Introduce the concept of a queue system and its benefits in managing asynchronous tasks, such as sending emails.
   * Discuss different queue technologies or libraries that can be used, such as RabbitMQ, AWS SQS, or Bull.
5. Order Queue:
   * Describe the implementation of an order queue specifically for managing mailer tasks.
   * Explain how order-related emails are enqueued in the queue for processing.
6. Mailer Service:
   * Discuss the architecture and components of the mailer service responsible for processing tasks from the order queue.
   * Explain how the mailer service retrieves tasks from the queue and sends the corresponding emails.
7. Email Templates:
   * Discuss the design and implementation of email templates for different types of order-related emails.
   * Explain how dynamic data, such as order details or customer information, is merged with the email templates.
8. Error Handling and Retry Mechanisms:
   * Explain how errors or failures in sending emails are handled within the mailer service.
   * Discuss retry mechanisms, such as exponential backoff or dead-letter queues, to handle failed email delivery.
9. Monitoring and Logging:
   * Discuss the importance of monitoring and logging in the mailer functionality.
   * Explain how system administrators or developers can track the status of mailer tasks and troubleshoot any issues.
10. Performance Optimization:
    * Discuss performance optimizations for the mailer functionality, such as throttling email sending, batch processing, or prioritization of critical emails.
11. Testing and Validation:
    * Describe the testing strategies used to validate the order queue and mailer functionality.
    * Discuss techniques such as unit testing, integration testing, and load testing for ensuring the reliability and scalability of the system.
12. Conclusion:
    * Summarize the key findings and contributions related to the implementation and evaluation of the queue system for mailer functionality.
    * Reflect on the challenges encountered and lessons learned during the development and testing of the system.
    * Discuss potential future enhancements or research directions in the context of order management and mailer systems.

Remember to adapt and expand on these topics based on your specific requirements and the scope of your bachelor thesis.

Backend Tehnologies And Infrastructure

Great! Based on the additional information you provided, you can include the following details in your bachelor thesis:

1. Backend Framework and Technologies:
   * Describe the use of Express.js as the backend framework for your application.
   * Explain how Express.js simplifies the development of RESTful APIs and handles routing and middleware.
2. Database Management:
   * Discuss the use of MongoDB as the database management system.
   * Explain the advantages of using a NoSQL database like MongoDB for storing and retrieving data.
3. Hashing Passwords:
   * Explain the use of bcrypt for password hashing in your application.
   * Discuss the importance of hashing passwords to enhance security and protect user data.
4. Development Environment:
   * Discuss the use of nodemon for automatically restarting the development environment.
   * Explain how nodemon improves the development workflow by monitoring file changes and restarting the server.
5. Queue System and RabbitMQ:
   * Describe the integration of RabbitMQ with your application for implementing the queue system.
   * Explain how amqplib is used to communicate with RabbitMQ and enqueue tasks.
6. Email Sending and Nodemailer:
   * Discuss the use of Nodemailer for sending emails in your application.
   * Explain how Nodemailer simplifies the process of composing and sending emails.
7. Containerization with Docker:
   * Explain the concept of containerization and its benefits in application deployment and scalability.
   * Discuss the use of Docker to encapsulate your application's components into containers for easier deployment.
8. Deployment with Docker Swarm:
   * Describe the use of Docker Swarm for deploying your application.
   * Explain how Docker Swarm orchestrates and manages container clusters for high availability and scalability.
9. Conclusion:
   * Summarize the key technologies and frameworks used in the backend development of your application.
   * Discuss the advantages and challenges faced during the implementation and deployment process.
   * Reflect on the impact of using these technologies on the development workflow, performance, and scalability.

For the frontend part of your bachelor thesis, where Vue.js is used along with Pinia for state management and reactivity, axios for API requests, vue-router for routing, and a library for rendering .obj files, you can include the following details:

1. Introduction:
   * Provide an overview of the frontend technologies used in your application.
   * Explain the importance of Vue.js and its ecosystem in modern web development.
2. Vue.js Framework:
   * Discuss the key features and benefits of Vue.js as a frontend framework.
   * Explain how Vue.js facilitates the development of reactive and component-based user interfaces.
3. State Management with Pinia:
   * Introduce Pinia as a state management solution for Vue.js applications.
   * Discuss the advantages of using Pinia for managing application-level state and reactivity.
4. API Requests with Axios:
   * Explain the role of Axios as an HTTP client library for making API requests.
   * Discuss how Axios simplifies the process of handling asynchronous data fetching and updating.
5. Routing with vue-router:
   * Describe the use of vue-router for handling client-side routing in your application.
   * Explain how vue-router enables navigation between different views and components.
6. Rendering .obj Files:
   * Introduce the library or tool used for rendering .obj files in your application.
   * Discuss the features and capabilities of the library for rendering 3D models.
7. Component Development:
   * Explain the component-based development approach in Vue.js.
   * Discuss the organization and structure of Vue.js components in your application.
8. Data Flow and State Management:
   * Explain how data flows between components and the central store managed by Pinia.
   * Discuss the reactive nature of Pinia and how it ensures consistency and reactivity in the application.
9. Integration with Backend APIs:
   * Describe how Axios is used to send HTTP requests to the backend APIs.
   * Explain the process of retrieving data from the server and updating the application state.
10. Routing and Navigation:
    * Discuss the use of vue-router for defining routes and navigation within your application.
    * Explain how routing is integrated with component rendering and state management.
11. 3D Model Rendering:
    * Describe how the library for rendering .obj files is integrated into your Vue.js components.
    * Explain the techniques and features used to display and interact with 3D models in the browser.
12. Conclusion:
    * Summarize the role of Vue.js, Pinia, Axios, vue-router, and the 3D rendering library in your frontend development.
    * Reflect on the advantages and challenges encountered during the implementation process.
    * Discuss potential future enhancements or research directions in the context of frontend development with Vue.js.

When creating the table of contents for your bachelor thesis, it's important to choose titles that accurately reflect the structure and content of your work. Here are some suggested table of contents titles that you can consider:

1. Introduction
   * Background and Context
   * Problem Statement
   * Objectives
   * Scope and Limitations
   * Thesis Structure
2. Literature Review
   * Overview of Relevant Literature
   * Theoretical Framework
   * Previous Studies and Research
   * Critical Analysis and Discussion
3. Methodology
   * Research Design
   * Data Collection Methods
   * Data Analysis Techniques
   * Ethical Considerations
4. System Design and Architecture
   * System Requirements
   * High-Level Architecture
   * Component Design
   * Database Design
5. Implementation
   * Technologies and Tools Used
   * Backend Development
   * Frontend Development
   * Integration and Testing
6. Results and Findings
   * Presentation of Results
   * Data Analysis and Interpretation
   * Evaluation of System Performance
   * Comparison with Objectives
7. Discussion
   * Interpretation of Findings
   * Discussion of Results in Relation to Literature
   * Insights and Implications
   * Limitations and Future Work
8. Conclusion
   * Summary of Achievements
   * Contribution to the Field
   * Practical and Theoretical Implications
   * Recommendations for Future Work
9. References
   * List of Cited Works
10. Appendices
    * Supporting Materials (e.g., Code Snippets, Diagrams)
    * Survey Questionnaires
    * Additional Data and Analysis

For your specific software application, here are some metrics and analyses that you can consider:

1. Performance Analysis:
   * 3D Model Loading Time: Measure the time taken to load and render 3D models from the .obj files.
   * Rendering Frame Rate: Evaluate the frame rate at which the 3D models are rendered in the application.
   * AWS S3 Transfer Speed: Monitor the speed of uploading and downloading 3D models to/from AWS S3.
2. Quality Analysis:
   * Compatibility Testing: Test the compatibility of the application with different web browsers and devices.
   * Error Handling: Monitor and analyze error logs and exception reports specific to 3D model rendering and AWS S3 operations.
   * 3D Model Validation: Validate the integrity and correctness of the .obj files before rendering.
3. User Experience Analysis:
   * User Interaction Metrics: Monitor user interactions within the 3D model viewer, such as zooming, rotating, and panning.
   * Performance Feedback: Gather user feedback on the performance and responsiveness of the 3D model rendering.
   * AWS S3 Upload/Download Success Rate: Track the success rate of uploading and downloading 3D models from AWS S3.
4. Security Analysis:
   * AWS S3 Access Control: Ensure proper access controls and permissions are implemented for the AWS S3 bucket containing the 3D models.
   * Data Encryption: Evaluate the encryption mechanisms employed during file transfers and storage on AWS S3.
5. Business Impact Analysis:
   * Revenue Generation: Analyze the impact of the software application on revenue generation, considering sales or subscriptions related to 3D models.
   * User Retention Rate: Measure the rate at which users continue using the application and accessing 3D models.
6. Customer Satisfaction Analysis:
   * User Feedback on 3D Model Rendering: Gather feedback from users regarding the quality, performance, and visual experience of the rendered 3D models.
   * User Satisfaction with AWS S3 Integration: Assess user satisfaction with the integration and usage of AWS S3 for storing and retrieving 3D models.