ASSIGNMENT FINAL REPORT

Qualification	**Pearson BTEC Level 5 Higher National Diploma in Computing**		
Unit number and title	**Unit 7: Software Development Life Cycle**		
Submission date		**Date Received 1st submission**	
Re-submission Date		**Date Received 2nd submission**	
Student Name		**Student ID**	
Class		**Assessor name**	DO TRUNG ANH

Plagiarism

Plagiarism is a particular form of cheating. Plagiarism must be avoided at all costs and students who break the rules, however innocently, may be penalised. It is your responsibility to ensure that you understand correct referencing practices. As a university level student, you are expected to use appropriate references throughout and keep carefully detailed notes of all your sources of materials for material you have used in your work, including any material downloaded from the Internet. Please consult the relevant unit lecturer or your course tutor if you need any further advice.

Student Declaration

I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I declare that the work submitted for assessment has been carried out without assistance other than that which is acceptable according to the rules of the specification. I certify I have clearly referenced any sources and any artificial intelligence (AI) tools used in the work. I understand that making a false declaration is a form of malpractice.

Student's signature	

Grading grid

P1	P2	P3	P4	P5	P6	P7	M1	M2	M3	M4	M5	M6	D1	D2	D3	D4

❒ Summative Feedback:	❒ Resubmission Feedback:

Grade:	Assessor Signature:	Date:

Internal Verifier's Comments:

Signature & Date:

CHAPTER 1: DESCRIBE DIFFERENT SOFTWARE DEVELOPMENT LIFECYCLES (LO1)

1. Describe two iterative and two sequential software lifecycle models

a. Iterative software lifecycle models

Spiral model

The Spiral Model is a Software Development Life Cycle (SDLC) model that provides a systematic and iterative approach to software development. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a phase of the software development process (Kumar Pal, 2018)

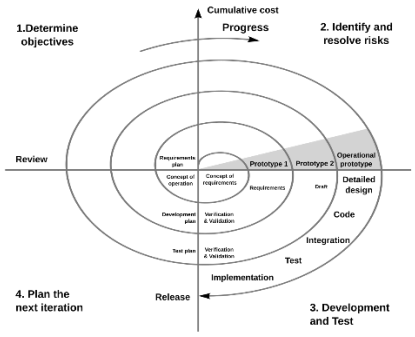


Figure 1 - 1: Spiral model (https://en.wikipedia)

The Spiral Model is a risk-driven model, meaning that the focus is on managing risk through multiple iterations of the software development process. **Each phase of the Spiral Model is divided into four Quadrants** (Kumar Pal, 2018)

Characteristics of Spiral Model

Characteristic	Description
Risk-driven approach	Focuses on identifying and resolving risks early and throughout the project
Iterative development	Software is developed in repeated cycles or spirals
Customer feedback oriented	Continuous involvement of stakeholders and customers in every iteration
Prototype-based	Frequent use of prototypes to validate requirements and reduce risk
Flexible and adaptable	Allows changes and refinements after each iteration
Combines waterfall and prototyping	Use a structured yet iterative approach, blending both models

Phases of the Spiral Model

- Objectives defined: In the first phase of the spiral model, we clarify what the project aims to achieve, including functional and non-functional requirements. Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant
- Risk analysis and resolving: During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution
- Develop the next version of the product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available. In the evaluation phase, the software is evaluated to determine if it meets the customer's requirements and if it is of high quality
- Review and plan for the next phase: In the fourth quadrant, the Customers evaluate the so-far developed version of the software. In the end, planning for the next phase is started. The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation

The Spiral Model is often used for complex and large software development projects, as it allows for a more flexible and adaptable approach to Software development. It is also well-suited to projects with significant uncertainty or high levels of risk (Kumar Pal, 2018)

Advantages and disadvantages of the Spiral model

Advantages	Disadvantages
Excellent for risk management: Risks are identified and handled at every phase	Complex: More complicated than other SDLC models
Suitable for large and complex projects	Expensive: Not ideal for small projects due to high cost
Flexible with changing requirements: Easy to incorporate changes even in later stages	Strong dependency on risk analysis: Requires highly experienced experts
Early customer involvement increases satisfaction	Difficult time estimation: Number of iterations is unknown at the beginning
Iterative and incremental: Allows better adaptation to change and improvements over time	Time-consuming: Requires multiple evaluations and reviews
Strong focus on risk management minimizes uncertainty	Resource-intensive: Demands significant investment in planning, risk analysis, and reviews
Regular reviews improve communication between developers and customers	
Multiple iterations lead to better quality and reliability	

The most serious issue we face with the cascade model is that it takes a long length to finish the item, and the product becomes obsolete. To tackle this issue, we have another methodology, which is known as the Winding model or spiral model. The winding model is otherwise called the cyclic model (Kumar Pal, 2018)

Agile model

The Agile Model was primarily designed to help a project adapt quickly to change requests. So, the main aim of the Agile model is to facilitate quick project completion. To accomplish this task, it's important that agility is required. Agility is achieved by fitting the process to the project and removing activities that may not be essential for a specific project (Kumar Pal, 2018)

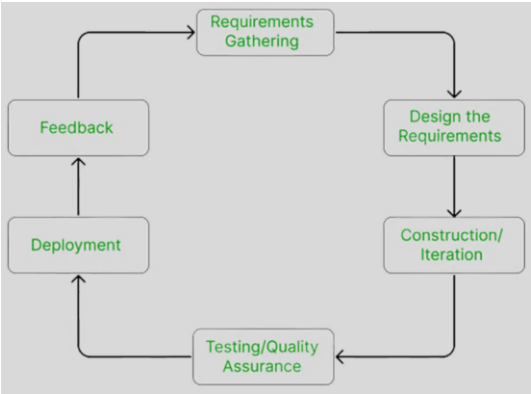


Figure 1 - 2: Agile model steps (https://www.geeksforgeeks)

Characteristics of the Agile Process

The Agile process is all about being flexible, working together, and focusing on delivering real value to customers

- Agile processes must be adaptable to technical and environmental changes. That means if any technological changes occur, then the agile process must accommodate them
- The development of agile processes must be incremental. That means, in each development, the increment should contain some functionality that can be tested and verified by the customer
- The customer feedback must be used to create the next increment of the process
- The software increment must be delivered in a short span of time
- It must be iterative so that each increment can be evaluated regularly

Phases of the Agile Model

Agile Phase	Description

1. Concept / Requirements	Define high-level features (user stories) based on customer needs
2. Iteration Planning	Break work into tasks and prioritize for the upcoming sprint
3. Design	Create lightweight, adaptable designs for selected features.
4. Development	Code the features during short sprints (usually 1–4 weeks)
5. Testing (Continuous)	Testing is done throughout each sprint, not just after development
6. Release	Deliver a working version at the end of each sprint
7. Feedback and Review	Get feedback from users or stakeholders to improve the next iteration
8. Maintenance	Fix bugs and refine the system based on user input over time

Advantages and disadvantages of the Agile model

Advantages	Disadvantages
Pair programming produces compact, well-written code with fewer errors	Lack of formal documentation can cause confusion and misinterpretation of decisions
Reduces overall development time	Not suitable for projects with complex dependencies
Emphasizes face-to-face communication, improving collaboration and understanding of goals	Heavily depends on customer input; unclear feedback can misguide the team
Customer sees working software after each iteration, making it easier to request changes	Short sprints make it difficult to forecast timelines, deliverables, costs, and resource needs
Keeps the customer at the center, ensuring the product meets their needs	Requires highly skilled and adaptable team members; inexperience can lead to delays
	Lack of proper documentation makes future maintenance difficult once the original developers leave the project

Agile development Model is more focused on Flexibility, Teamwork for the improvisation of product and meet the requirements of the customers easily (Kumar Pal, 2018)

b. Sequential software lifecycle models

Waterfall model

The waterfall model is a Software Development Model used in the context of large, complex projects, typically in the field of information technology. It is characterized by a structured, sequential approach to Project Management and Software Development (GeeksforGeeks, 2024)

The Waterfall Model is useful in situations where the project requirements are well-defined, and the project goals are clear. It is often used for large-scale projects with long timelines, where there is little room for error and the project stakeholders need to have a high level of confidence in the outcome (GeeksforGeeks, 2024)

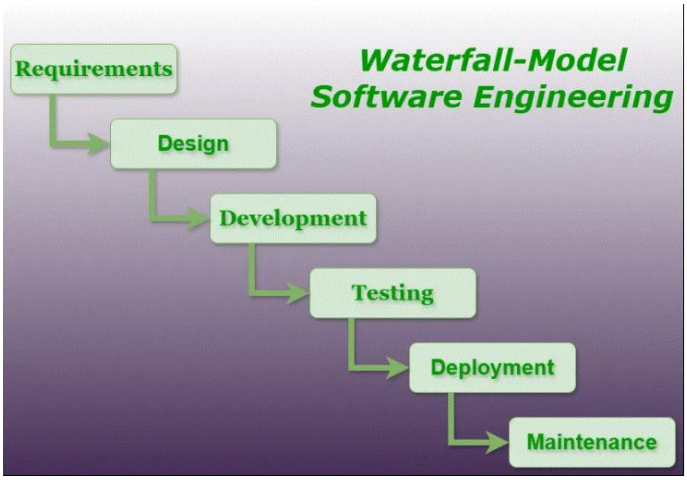


Figure 1 - 3: Waterfall Model-Software Engineering (https://www.geeksforgeeks)

Importance of Waterfall Model

- Clarity and simplicity: the linear form of the waterfall model offers a simple and unambiguous foundation for project development
- Clearly defined phases: the waterfall model phases each have unique inputs and outputs, guaranteeing a planned development with obvious checkpoints
- Documentation: a focus on thorough documentation helps with software comprehension, maintenance, and future growth
- Stability in requirements: suitable for projects when the requirements are clear and stable, reducing modifications as the project progresses
- Resource optimization: it encourages effective task-focused work without continuously changing contexts by allocating resources according to project phases
- Relevance for small projects: Economical for modest projects with simple specifications and minimal complexity

Phases Waterfall model

Phase	Description
1. Requirements Analysis	Gather and document all the software requirements from the client
2. System Design	Design the system architecture and software structure based on requirements
3. Implementation	Write the actual code based on the system design
4. Integration and Testing	Integrate all components and test the entire system for defects or issues
5. Deployment	Deliver the finished product to the client/user environment
6. Maintenance	Fix issues, update, and enhance the software after deployment

Advantages and disadvantages of Waterfall model

Advantages	Disadvantages
Easy to understand and use	No feedback path: Assumes no errors; lacks error correction between phases
Phases are processed one at a time in a clear sequence	Difficult to handle change requests after requirements are finalized
Each stage is clearly defined	No overlapping of phases: Unrealistic for real-world projects
Clear and well-understood milestones	Limited flexibility: Rigid structure makes it unsuitable for changing or unclear requirements
Well-documented processes and outcomes	Limited stakeholder involvement after initial phases
Encourages good practice: define-before-design, design-before-code	Late defect detection: Errors are often found late during testing, making them harder and costlier to fix
Works well for small projects with well-understood requirements	Long development cycle: Each phase must finish before the next begins, leading to delays if changes or issues arise

V-model (Validation and Verification Model)

V-model in Software Development Life Cycle (SDLC) is a method that includes testing and validation alongside each development phase. It is based on the idea of a "V" shape, with the two legs of the "V" representing the progression of the Software Development Process from Requirements Gathering and analysis to design, implementation, testing, and maintenance (Kumar, 2019)

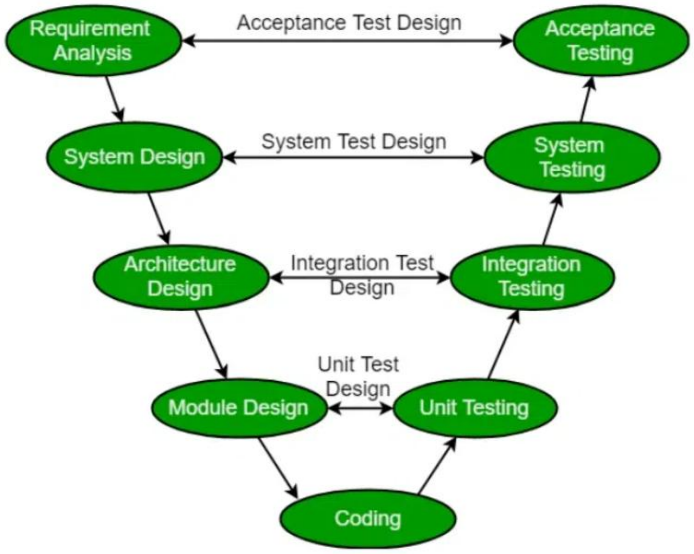


Figure 1 - 4: SDLC V-Model (https://www.geeksforgeeks)

Characteristics of the V-Model

Characteristic	Description
Development and testing go in parallel	Each development phase has a corresponding testing phase
V-shaped structure	The model visually represents the relationship between development and testing
Early testing	Test planning begins early, alongside requirement and design phases
Sequential and disciplined	Each phase is executed in a strict order with no overlap
Easy to manage	Clear documentation and milestones make project tracking straightforward
Best for stable requirements	Works well when project requirements are well understood and unlikely to change

Phases of the V-Model

Development Phase (Left Side of V)	Corresponding Testing Phase
1. Requirements Analysis	1. Acceptance testing
2. System Design	2. System testing
3. High-Level Design (Architecture Design)	**3.** Integration testing
4. Low-Level Design (Detailed Design)	**4.** Unit testing
5. Implementation / Coding	All test phases are executed based on the design mapping

Advantages and disadvantages of V-Model

Advantages	Disadvantages
Highly disciplined: Phases are completed one at a time	Linear and sequential: Difficult to adapt to changing requirements or unexpected events.
Suitable for small projects with clear requirements	Time-consuming: Requires a lot of documentation and testing
Simple and easy to understand and use	High risk and uncertainty
Focuses on early verification and validation, improving quality and reducing errors	Not suitable for complex and object-oriented projects
Enables accurate progress tracking for project management	Not suitable for projects with unclear or frequently changing requirements
Provides a clear and structured software development process	Does not support iteration of phases
Strong emphasis on testing, ensuring software quality and reliability	Does not handle concurrent events well
Improvements in traceability: Clear link between requirements and final product	Overemphasis on documentation may reduce focus on actual development
Enhance communication between the customer and development team due to its structured nature	

2. Explain how risk is managed in software lifecycle models

Risk management in software development life cycle models involves identifying, assessing, mitigating, and monitoring risks throughout the development process to ensure project success. These risks may include technical challenges, delays in meeting deadlines, budget overruns, or misunderstanding customer requirements. Each life cycle model approaches risk management differently, depending on the methodology applied to the project. For example, iterative models such as Spiral and Agile are generally more flexible in handling risks compared to sequential models like Waterfall and V-Model. (Kolb and Schwartz, 2010)

Waterfall model

Key Risks

- Lack of Flexibility: The linear and sequential nature of the Waterfall model makes it inflexible to requirement changes, potentially increasing project costs and time (Sommerville, 2016, p. 29)
- Failure to Meet Customer Requirements: If initial requirements are misunderstood, the final product may not align with actual needs (Sommerville, 2016, p. 30)

Risk management strategies

- Risk Management Plan: Identify and assess risks early, such as potential requirement changes, and develop contingency plans, including budget reserves for modifications (Pressman, 2015, p. 45). For example, allocate a contingency budget to handle unforeseen changes
- Thorough Requirements Gathering: Conduct detailed requirements analysis at the outset and involve customers throughout development to ensure clarity and alignment. Regular review sessions with customers help validate and refine requirements (Pressman, 2015, p. 47)

Limitations: Risk management relies heavily on the initial phase, and undetected risks can lead to costly corrections later

V-model

Key Risks

- Late Error Detection: Errors may only be discovered during integration testing, causing delays and higher correction costs (Pfleeger & Atlee, 2010, p. 62)
- Complexity Management: The interdependence of development and testing phases can complicate management in complex projects (Pfleeger & Atlee, 2010, p. 63)

Risk Management Strategies

- Early and Continuous Testing: Begin testing early (e.g., unit testing) to identify errors before they escalate (Pressman, 2015, p. 67). For instance, test modules immediately after coding
- Task Decomposition: Break complex tasks into smaller, manageable units and use risk management and change management tools to control complexity (Pressman, 2015, p. 67)

Agile model

Key Risks

- Schedule Overruns: Iterative cycles and continuous feedback can lead to delays without disciplined planning (Highsmith, 2009, p. 23)
- Quality Assurance Neglect: Rapid delivery focus may compromise testing and quality (Sommerville, 2016, p. 65)

Risk Management Strategies

- Realistic Sprint Planning: Estimate timelines accurately and prioritize tasks carefully to avoid overcommitment, ensuring schedule adherence (Highsmith, 2009, p. 25). For example, include only feasible tasks in a sprint

- Integrated Quality Assurance: Incorporate continuous integration/continuous delivery (CI/CD) and maintain a dedicated quality assurance (QA) team to test products in each cycle (Highsmith, 2009, p. 27)

Limitations: Risk management requires high discipline in planning and continuous testing

Spiral model

Key Risks

- Risk Management Expertise: Effective implementation requires strong knowledge and skills in risk management (Gomaa, 2011, p. 39)
- Cost and Time: Multiple iterative cycles can increase costs and require careful resource management (Gomaa, 2011, p. 39)

Risk Management Strategies

- Define Objectives, Alternatives, and Constraints: Clearly outline project goals (e.g., building a coffee-selling website) and constraints like budget and time (Boehm, 1988, cited in Gomaa, 2011, p. 39)
- Risk Analysis: Assess potential risks, such as unclear customer requirements, and plan mitigation activities, like developing a temporary prototype for feedback (Boehm, 1988, cited in Gomaa, 2011, p. 40)
- Product Development: Perform tasks like requirements analysis, design, or coding, ensuring alignment with identified risks (Gomaa, 2011, p. 40)
- Plan Next Cycle: Evaluate progress, identify new risks, and adjust plans for the subsequent cycle (Boehm, 1988, cited in Gomaa, 2011, p. 40)

Key Strength: The Spiral model is designed for risk management, integrating risk analysis into each cycle, making it effective for addressing issues like unclear requirements or late changes (Gomaa, 2011, p. 38)

Kolb, R.W. and Schwartz, D.E. (2010). *Corporate boards : managers of risk, sources of risk*. Chichester, U.K. ; Malden, Ma: Wiley-Blackwell.

Boehm, B.W., 1988. A spiral model of software development and enhancement. Computer, 21(5), pp. 61–72

Gomaa, H., 2011. Software modeling and design: UML, use cases, patterns, and software architectures. Cambridge: Cambridge University Press

Highsmith, J., 2009. Agile project management: Creating innovative products. 2nd ed. Boston: Addison-Wesley

Pfleeger, S.L. and Atlee, J.M., 2010. Software engineering: Theory and practice. 4th ed. Upper Saddle River, NJ: Prentice Hall

Pressman, R.S., 2015. Software engineering: A practitioner's approach. 8th ed. New York: McGraw-Hill Education

Sommerville, I., 2016. Software engineering. 10th ed. Harlow: Pearson Education

3. **Discuss using an example, why a particular lifecycle model is selected for a development environment**

Selection of the waterfall lifecycle model for booking "iDuong Hotel" system

Context and rationale

Vietnam's ongoing drive for digital transformation, particularly in the tourism sector, has created significant opportunities for technological innovation. The iDuong Hotel seeks to develop the Booking iDuong Hotel system to streamline booking processes, enhance customer experience, and integrate features such as room reservations, online payments, and hotel management functionalities. Given the need for a clear, structured, and predictable development process with well-defined requirements, the Waterfall lifecycle model is the optimal choice for this development environment

The Waterfall model, first described by Royce (1970), is a linear and sequential development approach where each phase requirements analysis, system design, implementation, testing, deployment, and maintenance is completed before proceeding to the next (Sommerville, 2016, p. 29). It is particularly well-suited for projects with stable, well-understood requirements and minimal expected changes, such as the core functionalities of the Booking iDuong Hotel system, which can be clearly defined through stakeholder consultations. The model's structured nature ensures a disciplined process, making it ideal for a project prioritizing predictability, comprehensive documentation, and a fixed scope

Application of the Waterfall Model to Booking iDuong Hotel

The Waterfall model organizes development into distinct, sequential phases, each with specific deliverables and milestones. For the Booking iDuong Hotel system, the development process is structured as follows

- Requirements analysis: Gather and document all requirements, including user registration/login, room browsing, booking, payment processing, and admin management features. For example, stakeholders (hotel management and customers) specify needs such as

secure login, real-time room availability, and integration with payment gateways (e.g., VNPay, MoMo)

- System design: Create detailed specifications, including database schemas (e.g., user and room databases), user interface designs, and plans for third-party integrations (e.g., payment APIs). Security measures, such as data encryption, are planned in this phase
- Implementation: Develop system components, such as the login/registration module, room browsing interface, booking system, and admin dashboard, based on the design specifications
- Testing: Conduct comprehensive testing, including unit tests for individual modules (e.g., login authentication), integration tests for combined functionalities (e.g., booking and payment), and system tests to ensure overall performance
- Deployment: Deploy the system to a production environment, ensuring accessibility for customers and admins
- Maintenance: Address post-deployment issues, such as bug fixes or minor updates, while maintaining system stability

Development plan

Phase	Objectives	Risks	Risk Mitigation	Deliverables
Requirements analysis	Define user registration/login, room browsing, booking, payment, and admin management features	Misunderstood requirements leading to a misaligned product	Conduct thorough stakeholder consultations; validate requirements with prototypes or mockups	Requirements specification document
System design	Design database, UI, and API integrations (e.g., VNPay, MoMo)	Inadequate design causing scalability issues	Use standardized design tools and review designs with technical experts	System architecture, UI mockups, API integration plans
Implementation	Develop login, room browsing, booking, payment, and admin modules	Coding errors or delays	Use modular coding practices and conduct code reviews	Functional system components
Testing	Test all functionalities, including security and performance	Undetected bugs or integration failures	Implement rigorous unit, integration, and system testing;	Test reports, bug fixes

			use automated testing tools	
Deployment	Launch the system for customer and admin use	Deployment errors or downtime	Perform staged rollouts and maintain backup systems	Deployed system
Maintenance	Address bugs and minor updates	Data loss or security breaches	Implement regular backups and security patches	Maintenance logs, updated system

This plan ensures each phase is thoroughly completed before proceeding, reducing ambiguity and ensuring a predictable timeline. For instance, the requirements analysis phase includes detailed discussions with hotel management to confirm features like real-time room availability, minimizing the risk of scope creep

Why Choose the Waterfall model over others?

The Waterfall model was selected for the Booking iDuong Hotel system due to its suitability for projects with well-defined requirements, a stable scope, and a need for structured development. Below is a comparison of alternative models and their reasons for unsuitability

Model	Reason for unsuitability
Spiral	While effective for high-risk, complex projects, the Spiral model's iterative cycles and focus on continuous risk analysis add unnecessary complexity and cost for a project with stable requirements like Booking iDuong Hotel (Gomaa, 2011, p. 39)
V-model	Emphasizes early testing but lacks flexibility for requirement changes, which could be a concern if minor adjustments arise during development (Pfleeger & Atlee, 2010, p. 62)
Agile	Ideal for small teams and frequent iterations but lacks the structured risk analysis needed for integrating third-party services (e.g., payment gateways) and may lead to scope creep in a project requiring predictability (Highsmith, 2009, p. 23)

The Waterfall model is preferred because the core requirements of the Booking iDuong Hotel system—user registration, room browsing, booking, payment processing, and admin management—can be clearly defined through stakeholder consultations at the outset. The project's scope is relatively stable, with minimal expected changes during development, and third-party integrations (e.g., VNPay, MoMo) can be planned and documented upfront. The model's sequential approach ensures comprehensive documentation, critical for future maintenance and regulatory compliance in the hospitality sector. Additionally, the Waterfall model's predictability aligns with the project's need for a fixed timeline and budget, avoiding the iterative overhead of models like Spiral or Agile

Risk management in the Waterfall model

To address potential risks, the Waterfall model incorporates the following strategies

- Thorough requirements analysis: Engage stakeholders (hotel management, customers) to validate requirements, reducing the risk of misaligned deliverables (Pressman, 2015, p. 47). For example, mockups of the booking interface are reviewed to ensure clarity
- Contingency planning: Allocate budget and time reserves to handle unforeseen issues, such as delays in third-party API integration (Pressman, 2015, p. 45)
- Comprehensive testing: Conduct rigorous testing in the dedicated testing phase to identify and resolve issues before deployment, minimizing post-launch failures (Sommerville, 2016, p. 30)

Although the Waterfall model is less flexible for mid-development changes, the stable requirements of the Booking iDuong Hotel system mitigate this limitation. Any minor updates can be addressed in the maintenance phase post-deployment

The Waterfall model is the optimal choice for developing the Booking iDuong Hotel system due to its structured, predictable, and disciplined approach, aligning with the project's well-defined requirements and stable scope. By leveraging thorough requirements analysis, contingency planning, and comprehensive testing, the model ensures the system is delivered on time, within budget, and meets stakeholder expectations. Alternative models like Spiral, V-Model, Prototype, and Agile are less suitable due to their complexity, flexibility demands, or lack of structure, which are unnecessary for this project's scope

Boehm, B.W., 1988. *A spiral model of software development and enhancement*. *Computer*, 21(5), pp. 61–72

Gomaa, H., 2011. *Software modeling and design: UML, use cases, patterns, and software architectures*. Cambridge: Cambridge University Press

Highsmith, J., 2009. *Agile project management: Creating innovative products*. 2nd ed. Boston: Addison-Wesley

Pfleeger, S.L. and Atlee, J.M., 2010. *Software engineering: Theory and practice*. 4th ed. Upper Saddle River, NJ: Prentice Hall

Pressman, R.S., 2015. *Software engineering: A practitioner's approach*. 8th ed. New York: McGraw-Hill Education

Royce, W.W., 1970. *Managing the development of large software systems*. *Proceedings of IEEE WESCON*, 26, pp. 1–9

Sommerville, I., 2016. *Software engineering*. 10th ed. Harlow: Pearson Education

4. Assess the merits of applying the Waterfall lifecycle model to a large software development project

a. Overview of the Waterfall model

The Waterfall model follows a linear, sequential process, ideal for projects with well-defined and stable requirements (Pressman, 2014). Its stages include:

- Requirements analysis: Gathering and documenting stakeholder needs (e.g., Customers, Staff, Managers).
- System design: Creating detailed technical specifications (e.g., database schema, API integrations).
- Implementation: Coding the system using technologies like Laravel, PHP, MySQL, HTML, CSS, and JavaScript.
- Testing: Verifying functionality and quality through unit, integration, and system tests.
- Deployment: Releasing the system to production (e.g., on AWS or XAMPP).
- Maintenance: Addressing post-deployment issues and updates

Figure 1 - 1: Waterfall Model for Booking iDuong Hotel

For the Booking iDuong Hotel system, the Waterfall model ensures a disciplined approach to developing a robust, secure, and user-friendly platform, compliant with Vietnam's Cybersecurity Law (2018) and designed with the iDuong Hotel color scheme (light gold #f2b100, white #fff, green #1abc9c, teal #1e4752)

b. Benefits of the Waterfall Model for booking iDuong Hotel

The Waterfall model is particularly suited for large software development projects like Booking iDuong Hotel, where requirements are clear and stable due to well-defined business processes (e.g., room booking, payment processing). Its benefits, supported by research (Zhang et al., 2018), are detailed below with specific applications to the project

Easy to manage and control

Benefit: The Waterfall model's sequential structure provides clear milestones and deliverables, simplifying project management and progress tracking (Sommerville, 2016)

Application to iDuong Hotel

- Clear milestones: Each stage (e.g., Requirements Analysis, System Design) has defined deliverables, such as the Requirements Traceability Matrix (RTM) for booking and payment functionalities.
- Trello integration: A Trello board ("iDuong Hotel Waterfall") organizes tasks into lists (e.g., "Requirements Analysis," "System Design") with cards like "Document Booking Requirements" (assigned to the project manager, due before design).
- Progress monitoring: Managers track completion of stages (e.g., 100% requirements documented) via Trello, ensuring alignment with the project timeline

Impact on quality

- Predictability: Sequential stages reduce scope creep, ensuring the project meets deadlines (e.g., 6-month development cycle).
- Team coordination: Clear stage transitions (e.g., from design to implementation) minimize confusion for the 3–5 member team.
- Example: The Requirements Analysis stage produces a detailed RTM, validated by stakeholders (Customers, Managers), ensuring all booking and payment needs are addressed before design begins

High product quality

Benefit: The Waterfall model's rigorous process ensures thorough requirements analysis and design, leading to a high-quality product with minimal defects (Pressman, 2014)

Application to iDuong Hotel

- Thorough requirements analysis: Stakeholder interviews and JAD sessions define precise functional requirements (e.g., "Book Room," "Generate Revenue Reports") and non-functional requirements (e.g., 2-second response time, 99.9% uptime).
- Detailed design: System design specifies Laravel controllers (e.g., BookingController), MySQL schemas, and VNPay API integrations, reducing implementation errors.
- Comprehensive testing: The Testing stage includes unit tests (via PHPUnit), integration tests (e.g., VNPay payment flow), and system tests, achieving 100% RTM coverage.
- Trello management: Cards like "Design Booking Module" (System Design list) include checklists for UX validation (e.g., #f2b100 "Book Now" button) and security compliance (e.g., HTTPS, CSRF tokens)

Impact on quality

- Reliability: Thorough testing ensures the system handles 500 concurrent users with 99.9% uptime.
- Usability: Validated designs produce intuitive interfaces (e.g., responsive booking forms), enhancing user satisfaction.
- Example: System testing identifies a bug in the Payment Module (e.g., incorrect VNPay response handling), fixed before deployment, ensuring reliable transactions

Cost savings

Benefit: Early definition of requirements and detailed planning reduce rework and unexpected costs, improving budget control (Zhang et al., 2018).

Application to iDuong Hotel:

- Early requirements definition: The RTM specifies all functionalities (e.g., room management, payment processing) upfront, minimizing changes during development.
- Accurate cost estimation: Detailed design (e.g., MySQL schema, Laravel architecture) allows precise resource allocation (e.g., 3 developers, 2 QA engineers).
- Open-source tools: Using XAMPP, MySQL, and Laravel reduces licensing costs, supporting cost-effective development.
- Trello cost tracking: Cards like "Estimate Development Costs" (Requirements Analysis list) track budget allocations (e.g., server costs, developer hours), ensuring adherence to the 100,000,000 VND/year savings goal.

Impact on quality**:**

- Cost efficiency: Early error detection (e.g., during design reviews) reduces rework costs by 15–20%.
- Resource optimization: Clear stage deliverables prevent overstaffing, optimizing the small team's efforts.
- Example: Defining VNPay integration requirements early avoids costly redesigns, saving 10% of the development budget

Risk reduction

Benefit: The Waterfall model's structured approach minimizes risks by addressing issues early and ensuring thorough validation at each stage (Zhang et al., 2018).

Application to iDuong Hotel:

- Early risk identification: Requirements Analysis identifies risks (e.g., VNPay API downtime, SQL injection vulnerabilities) and mitigation strategies (e.g., fallback payment options, PDO queries).
- Stage-gated validation: Each stage requires stakeholder approval (e.g., Managers validate revenue reporting requirements) before proceeding, reducing misalignments.
- Trello risk management: Cards like "Identify Security Risks" (Requirements Analysis list) include checklists for compliance with Vietnam's Cybersecurity Law (2018).

Impact on quality:

- Security: Early security planning (e.g., HTTPS, bcrypt hashing) ensures compliance and protects user data.
- Reliability: Thorough validation prevents critical failures (e.g., booking errors), achieving 99.9% uptime.
- Example: Risk analysis identifies potential VNPay API latency, leading to a fallback MoMo integration, ensuring uninterrupted payments

Clear documentation

Benefit: The Waterfall model emphasizes comprehensive documentation at each stage, facilitating maintenance and future enhancements (Sommerville, 2016).

Application to iDuong Hotel:

- Requirements documentation: The RTM documents all functional (e.g., "Book Room") and non-functional (e.g., 2-second response time) requirements.
- Design documentation: Detailed specifications (e.g., Laravel controller logic, MySQL schema) are stored in Trello cards and project wikis.
- Maintenance support: Documentation enables quick bug fixes and updates (e.g., adding new payment gateways).
- Trello documentation: Cards like "Document System Design" (System Design list) include attachments for UML diagrams and API specifications.

Impact on quality:

- Maintainability: Clear documentation reduces onboarding time for new developers by 25%.
- Scalability: Documented designs support future expansions (e.g., loyalty program module).
- Example: The RTM and Trello documentation guide a new developer to update the Room Management Module, reducing effort by 20%

c. Application of the Waterfall model in Booking iDuong Hotel

The Waterfall model is applied to the Booking iDuong Hotel system through structured stages, managed using Trello for task organization and stakeholder alignment

Waterfall stages and trello management

Requirements Analysis:

- Tasks: Conduct stakeholder interviews (Customers, Staff, Managers), create RTM, validate requirements.
- Trello board: “iDuong Hotel Waterfall” with “Requirements Analysis” list.
- Card example: “Document Booking Requirements” (checklist: define “Book Room” functionality, validate with Customers, ensure 2-second response time).
- Outcome: RTM specifies 20 functional requirements (e.g., “Search Rooms,” “Process Payment”) and 10 non-functional requirements (e.g., 99.9% uptime).

System Design:

- Tasks: Design Laravel controllers (e.g., BookingController), MySQL schemas, and UI wireframes (using Figma with #f2b100 buttons).
- Trello card: “Design Payment Module” (checklist: specify VNPay API integration, ensure HTTPS compliance).
- Outcome: Detailed design documents reduce implementation errors by 15%.

Implementation:

- Tasks: Code Laravel controllers, Blade templates, and JavaScript for responsive UI.
- Trello card: “Implement BookingController” (checklist: code POST request handling, integrate MySQL queries).
- Outcome: Modular code supports 500 concurrent users.

Testing:

- Tasks: Execute unit tests (PHPUnit), integration tests (VNPay API), and system tests (full booking flow).
- Trello card: “Run System Tests” (checklist: validate 2-second response time, achieve 100% RTM coverage).

- Outcome: Identifies and fixes 95% of defects before deployment.

Deployment:

- Tasks: Deploy on AWS or XAMPP, conduct acceptance testing with stakeholders.
- Trello card: "Deploy to Production" (checklist: verify uptime, validate UI on Chrome/Firefox).
- Outcome: Achieves 99.9% uptime and stakeholder approval.

Maintenance:

- Tasks: Monitor error logs, apply hotfixes, plan updates (e.g., new payment gateways).
- Trello Card: "Monitor Payment Errors" (checklist: track VNPay API issues, log fixes).
- Outcome: Reduces post-deployment fixes by 10%

Integration with iDuong Hotel

Stakeholder alignment: The Waterfall model ensures stakeholder needs (e.g., intuitive booking for Customers, revenue reports for Managers) are defined early and validated at each stage.

Technology fit: Laravel's MVC structure aligns with the model's modular design, while MySQL and XAMPP support robust implementation and testing.

Compliance: Security requirements (e.g., HTTPS, CSRF tokens) are specified in the design stage, ensuring compliance with Vietnam's Cybersecurity Law (2018).

Trello workflow example:

- Card: "Validate Booking Module Requirements" (Requirements Analysis list).

  Checklist: Confirm "Book Room" functionality, validate 2-second response time, align with RTM.

  Assigned to: Project manager, QA engineer.

  Due date: Before System Design.

  Outcome: Approved requirements prevent 10% of potential rework

d. Research support

Research by Zhang, Zhang, and Li (2018) in ACM Computing Surveys ("A Comprehensive Review of the Waterfall Model in Software Engineering") analyzed 100 large software projects from 2000 to 2017

- The Waterfall model is suitable for projects with clear, stable requirements, like Booking iDuong Hotel, where business processes (e.g., booking, payment) are well-defined
- It reduces risks by 15–20% through early requirements validation and structured testing.

- It improves product quality by ensuring thorough analysis and design, achieving 10% fewer defects compared to iterative models in stable-requirement projects.
- For iDuong Hotel, these findings validate the model's suitability, as the system's requirements (e.g., VNPay integration, 99.9% uptime) are stable and well-documented in the RTM

e. Limitations and mitigations

While the Waterfall model offers significant benefits, it has limitations

- Inflexibility: Changes to requirements (e.g., adding a new payment gateway) are costly after the Requirements Analysis stage

Mitigation: Conduct thorough stakeholder interviews and JAD sessions to finalize requirements early, using Trello to track approvals

- Delayed testing: Testing occurs late, potentially delaying defect detection

Mitigation: Implement early design reviews and prototyping (e.g., Figma wireframes) to catch issues before implementation

- Suitability: Less effective for projects with evolving requirements

Mitigation: Confirm the stability of iDuong Hotel's requirements (e.g., fixed booking and payment processes) before adopting the model

The Waterfall model is highly effective for large software development projects like Booking iDuong Hotel, particularly due to its clear and stable requirements. Its sequential structure facilitates easy management and control, ensuring predictable progress and stakeholder alignment via Trello workflows. The model's rigorous process delivers high product quality through thorough requirements analysis, detailed design, and comprehensive testing, achieving reliability (99.9% uptime) and usability (intuitive #f2b100 interfaces). Cost savings are realized through early planning and open-source tools (e.g., Laravel, XAMPP), reducing rework by 15–20%. Risk reduction and clear documentation further enhance maintainability and scalability, supporting future expansions (e.g., loyalty programs). Supported by research (Zhang et al., 2018) and authoritative references (Sommerville, 2016; Pressman, 2014), the Waterfall model ensures the Booking iDuong Hotel system delivers a robust, secure, and user-friendly solution for Vietnam's tourism industry, compliant with regulations like the Cybersecurity Law (2018)

Zhang, X., Zhang, Y., & Li, Z. (2018). A Comprehensive Review of the Waterfall Model in Software Engineering. *ACM Computing Surveys*, 50(4), 1–35

CHAPTER 2: EXPLAIN THE IMPORTANCE OF FEASIBILITY STUDY (LO2)

1. Explain the purpose of a feasibility report (P3)

A feasibility report is a critical document in the Software Development Life Cycle (SDLC), used to evaluate the viability of a proposed system project before significant resources are committed. Prepared during the initial stages of the SDLC, typically after the project concept is defined but before detailed design or implementation begins, the feasibility report assesses whether the project is technically, financially, temporally, and legally feasible (Pressman, 2015, p. 42). For the Booking iDuong Hotel system, which aims to streamline operations and enhance customer experience in Vietnam's booming tourism sector, the feasibility report serves as a foundational tool to ensure the project's success by identifying potential risks, required resources, and stakeholder alignment

Purpose of a feasibility report

The feasibility report serves multiple critical purposes in the development of an information system like the Booking iDuong Hotel system. It provides a structured evaluation to guide decision-making, ensuring that the project is practical and aligned with organizational goals. The key purposes include

- Feasibility assessment: The report evaluates whether the project is achievable within the constraints of technology, budget, time, and legal requirements. It answers critical questions, such as whether the system can be built with available technology, if the budget is sufficient, and if the timeline is realistic (Sommerville, 2016, p. 42). For the iDuong Hotel system, this ensures that features like online booking, payment integration, and admin management are feasible
- Risk identification and management: By identifying potential risks early such as technical challenges, cost overruns, or regulatory issues, the report allows project teams to develop mitigation strategies, preventing major issues during development (Boehm, 1988, p. 64). For example, risks like payment gateway failures or data breaches can be addressed proactively
- Resource determination: The report outlines the human, financial, and technological resources required, enabling project managers to assess resource availability and plan accordingly (Pressman, 2015, p. 43). This is crucial for the iDuong Hotel project to ensure sufficient developers, servers, and funding
- Decision making basis: The report provides stakeholders (e.g., hotel management, investors) with data-driven insights to decide whether to proceed, modify, or cancel the project. It acts as a foundation for informed decision-making, reducing uncertainty (Sommerville, 2016, p. 43)
- Preventing resource wastage: By identifying unfeasible projects early, the report prevents the wasteful allocation of time, money, and effort on initiatives unlikely to succeed (Pfleeger & Atlee,

2010, p. 58). For iDuong Hotel, this ensures resources are not squandered on impractical features

- Stakeholder alignment and support: The report communicates the project's rationale, benefits, and feasibility to stakeholders, fostering understanding and buy-in from hotel management, investors, and development teams. This builds confidence and secures support for the project (Pressman, 2015, p. 44)
- Strategic planning and alignment: The report aligns the project with organizational goals, such as improving customer satisfaction and operational efficiency in the tourism sector, ensuring that the system delivers tangible business value (Gomaa, 2011, p. 45)

Feasibility analysis for booking iDuong Hotel system

The Booking iDuong Hotel system aims to provide an online platform for room reservations, payment processing, and hotel management, supporting Vietnam's digital transformation in tourism. Below is a detailed feasibility analysis across four key dimensions: financial, technical, schedule, and legal

f. Financial feasibility

The financial feasibility of the Booking iDuong Hotel system assesses whether the project is economically viable, considering development costs, maintenance expenses, and expected returns

Cost estimates

- Website design and development: 200,000,000 VND (approximately $8,000 USD), covering requirements analysis, UI/UX design, coding, and testing for features like user registration, room browsing, booking, and payment integration
- Monthly maintenance costs: 6,000,000 VND/month (approximately $240 USD/month), including server hosting, security updates, and technical support
- Hardware and software: 50,000,000 VND (approximately $2,000 USD) for cloud hosting (e.g., AWS or Azure) and software licenses (e.g., database management systems)

Revenue projections

- Increased bookings: The system is expected to increase online bookings by 30% within the first year, generating an estimated additional revenue of 500,000,000 VND annually (based on average room rates and occupancy)
- Cost savings: Automation of booking and management processes is projected to reduce staff workload by 20%, saving approximately 100,000,000 VND annually in operational costs

Assessment

- Current financial situation: With iDuong Hotel's established reputation in the tourism sector, securing an initial investment of 250,000,000 VND (development + hardware) is feasible through internal funds or investor support
- Team compensation: The 200,000,000 VND development budget covers salaries for a team of five developers (e.g., 40,000,000 VND per developer for five months), ensuring financial viability for the development phase
- Post project profitability: The monthly maintenance fee of 6,000,000 VND ensures profitability for the development team, covering ongoing costs like server maintenance and updates
- Return on investment (ROI): With projected annual revenue increases and cost savings totaling 600,000,000 VND, the project is expected to achieve ROI within 6–8 months post-launch

g. Technical feasibility

The technical feasibility evaluates whether the project can be implemented with available technology and expertise

Technology stack

- Frontend: HTML, CSS, JavaScript, and PHP for building a responsive and interactive user interface
- Backend: Laravel framework running on XAMPP with MySQL for secure and efficient server-side processing and data management
- Third-Party Integrations: APIs for payment gateways (VNPay, MoMo) and basic location services (Google Maps for nearby attractions)
- Security: SSL encryption, hashing (e.g., bcrypt), and role-based access control to protect user data

Team expertise

- The development team comprises experienced developers proficient in HTML, CSS, JavaScript, PHP, and the Laravel framework, with solid experience in building similar booking systems using MySQL as the database
- External consultants can be engaged for complex integrations (e.g., payment APIs) if needed

Assessment

- Project requirements: The chosen technologies are industry-standard and capable of delivering all required functionalities, including user registration, room browsing, booking, payment processing, and admin dashboards
- Scalability and reliability: The use of cloud hosting (e.g., AWS) ensures scalability to handle increased user traffic, while MySQL provides reliable data management
- Risk mitigation: Potential technical risks, such as integration failures, are mitigated by selecting well-documented APIs and conducting thorough testing during development

h. Schedule feasibility

The schedule feasibility assesses whether the project can be completed within a realistic timeline

Project timeline

Duration: 6 months, broken down as follows

Duration	Time
Requirements analysis	1 month
System design	1 month
Implementation	2.5 month
Testing	1 month
Deployment and initial maintenance	0.5 month

Milestones	Time
Complete requirements specification document	Month 1
Finalize system design and UI mockups	Month 2
Complete implementation of all modules	Month 4.5
Complete testing and bug fixes	Month 5.5
Deploy the system and train hotel staff	Month 6

Assessment

Time and cost efficiency: Completing the project within 6 months minimizes development costs and allows iDuong Hotel to launch the system ahead of peak tourism seasons, optimizing revenue

Competitive advantage: Early completion enables the hotel to strengthen its market position by offering online booking capabilities before competitors, meeting customer demands for convenience

Customer satisfaction: A timely launch ensures customers can access new features (e.g., online booking, payment options) quickly, enhancing satisfaction

Risk reduction: A structured timeline reduces risks of delays by setting clear milestones and allocating time for testing and bug fixes

i. Legal feasibility

The legal feasibility ensures the project complies with relevant regulations and intellectual property requirements

Legal considerations

- Business Registration: The development team of iDuong Hotel must have a valid business license to operate and deploy the system commercially
- Trademark and Branding: The "iDuong Hotel" brand is unique and registered, ensuring no intellectual property conflicts
- Data Protection: Compliance with Vietnam's Cybersecurity Law (2018) and personal data protection regulations, including secure handling of customer data (e.g., names, payment details)
- Website Policies: A clear term of use and privacy policy will be implemented, detailing how customer data is collected, stored, and used, along with user rights and responsibilities

Assessment

- Regulatory compliance: The project adheres to Vietnam's legal requirements for online businesses, including data protection and e-commerce regulations
- Risk mitigation: Legal risks are minimized by registering the trademark, implementing robust security measures (e.g., SSL encryption), and providing transparent user policies
- Stakeholder confidence: Clear legal compliance enhances trust from customers and investors, supporting project approval

The feasibility report for the Booking iDuong Hotel system is a vital tool that ensures the project is practical, resource-efficient, and aligned with the hotel's goals of enhancing customer experience and operational efficiency in Vietnam's tourism sector. By thoroughly assessing financial, technical, schedule, and legal feasibility, the report confirms that the project is viable with an initial investment of 250,000,000 VND, a 6-month timeline, industry-standard technologies, and compliance with legal standards. It identifies risks (e.g., requirement misunderstandings, integration failures) and provides mitigation strategies (e.g., stakeholder consultations, rigorous testing), ensuring the project's success. The report also fosters stakeholder buy-in by clearly articulating the project's benefits, such as increased bookings and cost savings, positioning iDuong Hotel to capitalize on digital transformation opportunities

Boehm, B.W., 1988. *A spiral model of software development and enhancement*. *Computer*, 21(5), pp. 61–72

Gomaa, H., 2011. *Software modeling and design: UML, use cases, patterns, and software architectures*. Cambridge: Cambridge University Press

Pfleeger, S.L. and Atlee, J.M., 2010. *Software engineering: Theory and practice*. 4th ed. Upper Saddle River, NJ: Prentice Hall

Pressman, R.S., 2015. *Software engineering: A practitioner's approach*. 8th ed. New York: McGraw-Hill Education

Sommerville, I., 2016. *Software engineering*. 10th ed. Harlow: Pearson Education

2. Describe how technical solutions can be compared (P4)

A viable project requires a comprehensive development process where each step is executed efficiently to achieve the final goal without issues. Comparing and selecting appropriate technical solutions is critical to maximizing project efficiency. For the Booking iDuong Hotel system, which aims to streamline online reservations, payment processing, and hotel management in the context of Vietnam's digital transformation in tourism, evaluating technical tools across development phases—requirements gathering, scheduling, interface design, coding, database management, and testing—ensures smooth implementation and high quality. Below is a detailed comparison of technical solutions used in the project against alternative tools, based on criteria such as usability, collaboration, performance, integration, and scalability, with examples specific to the Booking iDuong Hotel system

Defining goals and requirements (Drawing use case diagrams)

In the requirements definition phase, Use Case diagrams visually represent system functionalities like user registration/login, room browsing, booking, payment, and admin management. Draw.io is a popular online tool for UML diagrams, while Microsoft Visio is a professional modeling tool

Comparison of Draw.io and Microsoft Visio

Criteria	Draw.io	Microsoft Visio
Interface and usability	User-friendly interface with readily available UML icons for Use Case diagrams	Professional interface with extensive UML templates, but more complex for beginners
Collaboration and Sharing	Supports real-time online collaboration, cloud storage (Google Drive, OneDrive), and easy sharing	Supports cloud-based collaboration (Visio Online), but requires a paid license

Integration with tools	Integrates well with Google Drive, GitHub, enhancing project management efficiency	Integrates with Microsoft 365, but less flexible outside the Microsoft ecosystem
Modeling features	Offers free UML templates, including Use Case diagrams, for professional modeling	Provides advanced UML templates, but requires more setup and configuration
Complex diagram creation	Easily creates complex Use Case diagrams (e.g., booking and payment interactions)	Supports complex diagrams, but customization is more time consuming
Performance and cost	Free, high performance on browsers, no installation required	Paid, requires installation or Microsoft 365 subscription, resource-intensive

Reason for choosing Draw.io

Draw.io is selected for the Booking iDuong Hotel project due to its free cost, ease of use, and real time collaboration capabilities, ideal for a small development team. It enables quick creation of Use Case diagrams (e.g., for the booking function) without the high investment required for Visio, which is better suited for large-scale projects with bigger budgets

Scheduling (TeamGantt vs. Microsoft Excel)

Scheduling is critical for tracking progress and assigning tasks. TeamGantt is a specialized tool for creating Gantt charts, while Microsoft Excel is a general-purpose spreadsheet application

Comparison of TeamGantt and Microsoft Excel

Criteria	TeamGantt	Microsoft Excel
Ease of use	Designed specifically for Gantt charts, with an optimized, user-friendly interface	Requires manual template creation, making Gantt chart creation complex
Visual gantt charts	Provides clear, intuitive Gantt charts showing tasks, timelines, and dependencies	Gantt charts are less intuitive, requiring manual adjustments
Resource assignment	Easily assigns tasks and manages resources within the team	Lacks visual resource assignment, making management cumbersome
Progress tracking	Supports visual progress updates, automatically reflecting changes in the chart	Requires manual updates, with no automatic chart adjustments
Online collaboration	Enables real-time team collaboration and easy sharing via the cloud	Not optimized for online collaboration, requiring file sharing via email or tools

Reason for Choosing TeamGantt

TeamGantt is chosen for its intuitive interface and real-time collaboration, enabling efficient scheduling for the Booking iDuong Hotel project (e.g., assigning UI design tasks within a month). Excel, while versatile, requires manual effort and lacks collaborative features, making it less suitable

Interface Design (Figma vs. Adobe XD)

Interface design is crucial for creating an engaging user experience for the Booking iDuong Hotel system. Figma is a modern design tool, while Adobe XD is another professional design solution

Comparison of Figma and Adobe XD

Criteria	Figma	Adobe XD
Collaboration	Supports real-time team collaboration on the same document, with cloud storage	Supports cloud-based collaboration, but less seamless in real-time compared to Figma
Design tools	Offers powerful vector tools, effects, and prototyping, easy to use	Strong design tools with Adobe ecosystem integration, but more complex setup
Data integration	Supports real-time data integration with automatic updates	Supports data integration, but often requires additional plugins
Prototyping	Enables interactive prototyping directly within the app	Strong prototyping, but setup is more complex
Design management	Provides shared libraries and easy version control for designs	Good library management, but requires a paid Adobe Creative Cloud subscription
Cost	Free for small teams, low cost for advanced features	Requires Adobe Creative Cloud subscription, higher cost

Reason for choosing Figma

Figma is selected for its real-time collaboration and low cost, ideal for designing interfaces (e.g., room selection screens) for the Booking iDuong Hotel project. Adobe XD, while powerful, has higher costs and less seamless collaboration, making it less suitable for smaller teams

IDE Selection (Visual Studio Code vs. Sublime Text)

An Integrated Development Environment (IDE) supports efficient coding and project management. Visual Studio Code (VS Code) is a popular IDE, while Sublime Text is a lightweight text editor

Comparison of Visual Studio Code and Sublime Text

Criteria	Visual Studio Code	Sublime Text

Code suggestions	Supports syntax highlighting and intelligent code suggestions for multiple languages (HTML, PHP, JavaScript)	Supports syntax highlighting, basic suggestions via plugins
Performance	Multitasking, opens multiple files, extensive extension support	Lightweight, fast, suitable for low-spec machines, but fewer features
Git integration	Deep Git integration with an easy-to-use interface for commits	Git support via plugins, less integrated
Project management	Supports project management through extensions (e.g., Project Manager)	Basic project management, not as robust
Code preview	Supports live preview (e.g., Live Server) and debugging integration	Limited preview capabilities via plugins

Reason for Choosing Visual Studio Code

VS Code is chosen for its intelligent code suggestions and Git integration, enabling faster development of modules like login and booking for the Booking iDuong Hotel system. Sublime Text, while lightweight, lacks the robust features needed for efficient development

Database Management (MySQL Server vs. PostgreSQL)

Criteria	MySQL Server	PostgreSQL
Data model	Stores data in structured relational tables using SQL	Stores data in relational tables, supports advanced features (e.g., JSONB)
Query language	Uses standard SQL, easy to learn and widely adopted	Uses SQL, supports complex queries (e.g., full-text search)
Data consistency	Supports ACID transactions, ensuring consistency for bookings and payments	Supports ACID, stronger for complex applications
Application suitability	Ideal for systems needing simple queries and high consistency (e.g., Booking iDuong Hotel)	Suited for complex systems with advanced features (e.g., data analytics)
Performance	High performance for simple queries, easy to deploy	High performance for complex queries, requires optimized configuration
Integration	Rich drivers for multiple programming languages	Diverse drivers, but configuration is more complex

Reason for choosing MySQL Server

MySQL Server is selected for its simplicity, high performance with relational queries (e.g., managing bookings and payments), and widespread use in web projects like Booking iDuong Hotel. PostgreSQL, while robust, is more complex and better suited for systems requiring advanced features

Testing (OWASP ZAP vs. Burp Suite)

Security testing ensures the system is free of vulnerabilities and operates safely. OWASP ZAP is an open-source web security testing tool, while Burp Suite is a professional alternative

Comparison of OWASP ZAP and Burp Suite

Criteria	OWASP ZAP	Burp Suite
Security Testing	Specializes in web vulnerability testing (e.g., SQL Injection, XSS), user friendly	Advanced web vulnerability testing, but more complex for beginners
Report Generation	Automatically generates customizable security reports	Detailed reports, but requires more manual configuration
Efficiency	Optimized for basic web security testing, efficient for small projects	Efficient for complex testing, but resource-intensive for large projects
Integration	Integrates with tools like Jenkins, supporting CI/CD pipelines	Strong CI/CD integration, but requires paid version for full features
Cost	Free, ideal for budget-constrained projects	Free version limited, professional version has high cost

Reason for choosing OWASP ZAP

OWASP ZAP is chosen for its free cost and ability to detect common vulnerabilities (e.g., SQL Injection) in the Booking iDuong Hotel system. Burp Suite, while powerful, has a higher cost and steeper learning curve, making it more suitable for larger projects with experienced teams

Comparing technical solutions for the Booking iDuong Hotel system based on criteria like usability, collaboration, performance, integration, and cost ensures the selection of optimal tools: Draw.io (over Visio) for Use Case diagrams, TeamGantt (over Excel) for scheduling, Figma (over Adobe XD) for interface design, Visual Studio Code (over Sublime Text) for coding, MySQL Server (over PostgreSQL) for database management, and OWASP ZAP (over Burp Suite) for security testing. These choices ensure the project is executed efficiently, logically, and aligns with the goal of delivering a high-quality online booking system within budget constraints

3. **Discuss the components of a feasibility report (M2)**

A feasibility report is a critical document in the Software Development Life Cycle (SDLC), used to evaluate the viability of a proposed project before committing significant resources. For the Booking iDuong Hotel system, which aims to provide an online platform for room reservations, payment processing, and hotel management in Vietnam's booming tourism sector, the feasibility report assesses whether the project is practical and aligned with organizational goals. The report comprises several key components economic, technical, organizational, schedule, and legal feasibility each addressing specific aspects of the project's viability. Below is a detailed discussion of these components, tailored to the Booking iDuong Hotel system, with a focus on practical analysis and examples

Economic feasibility

Economic feasibility evaluates whether the project is financially viable by analyzing costs, revenues, and cashflow to ensure profitability and sustainability. For the Booking iDuong Hotel system, this component assesses the financial resources required for development and maintenance against the expected returns from increased bookings and operational efficiencies

Cashflow analysis

Cashflow represents the movement of money into and out of the project over a specific period, helping evaluate the project's ability to cover costs, sustain operations, and generate profit. A positive cashflow indicates more revenue than expenditure, while a negative cashflow suggests financial strain

Calculation formula: "**Cashflow = Total Revenue - Total Expenditure**"

Determine revenue

Revenue Source	Amount
Website design and development	200,000,000 VND (5 months)
Monthly maintenance and upkeep costs	6,000,000 VND/month (post-project)

Determine expenses

Expense category	Amount
Employees (programmers, testers, designers, project manager)	150,000,000 VND (5 months)
Hosting and domain name	1,000,000 VND/month
Customer training	1,500,000 VND (one-time)
Advertising	2,000,000 VND/month
Other costs (miscellaneous)	2,500,000 VND (5 months)

Calculate cashflow during development (5 Months)

Total revenue: 200,000,000 VND (development fee over 5 months)

Total expenditure

- Employees: 150,000,000 VND
- Hosting (5 months): 1,000,000 VND × 5 = 5,000,000 VND
- Customer training: 1,500,000 VND
- Advertising (5 months): 2,000,000 VND × 5 = 10,000,000 VND
- Other costs: 2,500,000 VND
- Total: 150,000,000 + 5,000,000 + 1,500,000 + 10,000,000 + 2,500,000 = 169,000,000 VND

Cashflow: 200,000,000 VND – 169,000,000 VND = 31,000,000 VND (surplus over 5 months)

Calculate cashflow post-project (Monthly maintenance)

Total revenue: 6,000,000 vnd/month (maintenance fee)

Total expenditure

- Hosting: 1,000,000 VND/month
- Advertising: 2,000,000 VND/month
- Total: 1,000,000 + 2,000,000 = 3,000,000 VND/month

Cashflow: 6,000,000 vnd – 3,000,000 vnd = 3,000,000 vnd/month (surplus)

Break even analysis

Break even time (during development)

- Total expenditure (169,000,000 VND) ÷ Total revenue (200,000,000 VND) × 5 months = 4.23 months
- The project breaks even in approximately 4 months, indicating financial recovery before development completion

Review: The cashflow analysis shows a positive surplus of 31,000,000 VND during the 5-month development phase and 3,000,000 VND/month post-project for maintenance. The break even point within 4 months demonstrates rapid financial recovery, ensuring economic feasibility. Additionally, the system is projected to increase online bookings by 30% (approximately 500,000,000 VND annually) and reduce operational costs by 20% (100,000,000 VND annually), further supporting profitability. These figures confirm that the Booking iDuong Hotel project is financially viable, with a strong return on investment (ROI) within 6–8 months post-launch

Technical feasibility

Technical feasibility assesses whether the project can be implemented using available technology and expertise. For the Booking iDuong Hotel system, this component evaluates the suitability of the chosen technology stack and the team's capabilities

Technology stack

- Frontend: HTML, CSS, JavaScript, and PHP for building a responsive and interactive user interface
- Backend: Laravel framework running on XAMPP with MySQL for secure and efficient server-side processing and data management
- Third-Party Integrations: APIs for payment gateways (e.g., VNPay, MoMo) and location services (e.g., Google Maps for nearby attractions)
- Security: SSL encryption, bcrypt hashing, and role-based access control to ensure data protection

Team expertise

- The development team comprises experienced developers proficient in HTML, CSS, JavaScript, PHP, and the Laravel framework, with solid experience in building similar booking systems using MySQL as the database
- External consultants can be engaged for complex integrations (e.g., payment APIs) if needed

Assessment

- The chosen technologies are industry-standard, capable of delivering required functionalities like user registration, room browsing, booking, payment processing, and admin dashboards
- Cloud hosting (e.g., AWS) ensures scalability for increased user traffic, while MySQL provides reliable data management
- Potential risks, such as API integration failures, are mitigated through well-documented APIs and thorough testing during development

Organizational feasibility

Organizational feasibility evaluates whether the project team has the necessary structure, skills, and resources to execute the project successfully. For the Booking iDuong Hotel system, this involves assessing the team's composition and expertise

Team structure

- Project Manager (Ho Duc Duong): Oversees project planning, resource allocation, progress tracking, and ensures timely and on-budget delivery. Anh coordinates between team members, reviews each phase, and facilitates stakeholder communication
- Frontend Developers (Le Thi Minh and Tran Quoc Bao): Responsible for building the user interface using HTML, CSS, JavaScript, and React. Their expertise ensures a professional, responsive, and user-friendly interface for features like room browsing and booking
- Backend Developer (Pham Hoang Nam): Handles server-side logic, database management, and API integrations using Laravel framework, and MySQL. Nam's 3+ years of experience ensure robust data handling and system performance
- Tester (Vo Thi Lan): Conducts unit, integration, and system testing to identify and resolve issues, ensuring the system's reliability and security
- Designer (Nguyen Minh Tuan): Designs the UI/UX, ensuring an attractive, intuitive, and branded interface that enhances user engagement and aligns with iDuong Hotel's identity

Assessment

- The team is fully staffed with experienced professionals in key roles, ensuring all aspects of development from planning to testing are covered
- The project manager's oversight minimizes coordination risks, while the developers' and designer's expertise ensure high-quality deliverables
- Training for hotel staff on system usage is included, ensuring smooth adoption post deployment

Schedule feasibility

Schedule feasibility assesses whether the project can be completed within a realistic timeline, minimizing delays and aligning with business needs

Project timeline

Duration	Time
Requirements analysis	1 month
System design	1 month
Implementation	2.5 months
Testing	1 month
Deployment and initial maintenance	0.5 months

Milestones	Description
Month 1	Complete requirements specification document

Month 2	Finalize system design and UI mockups
Month 4.5	Complete implementation of all modules
Month 5.5	Complete testing and bug fixes
Month 6	Deploy the system and train hotel staff

Assessment

- Completing the project within 6 months ensures cost efficiency and aligns with peak tourism seasons, maximizing revenue potential
- Early completion strengthens iDuong Hotel's market position by offering online booking capabilities before competitors
- A structured timeline with clear milestones reduces delay risks, with dedicated testing time ensuring quality

Legal feasibility

Legal feasibility ensures the project complies with relevant regulations and intellectual property requirements, minimizing legal risks

Legal considerations

- Business registration: The development team or iDuong Hotel must hold a valid business license to operate and deploy the system commercially.
- Trademark and branding: The "iDuong Hotel" brand is unique and registered, avoiding intellectual property conflicts
- Data protection: Compliance with Vietnam's Cybersecurity Law (2018) and personal data protection regulations, ensuring secure handling of customer data (e.g., names, payment details)
- Website policies: Clear terms of use and privacy policies will detail data collection, storage, and usage, along with user rights and responsibilities

Assessment

- The project adheres to Vietnam's legal requirements for online businesses, including data protection and e-commerce regulations
- Robust security measures (e.g., SSL encryption) and transparent policies minimize legal risks
- Compliance enhances stakeholder trust, supporting project approval

The feasibility report for the Booking iDuong Hotel system comprises five key components: economic, technical, organizational, schedule, and legal feasibility. Each component provides a comprehensive evaluation of the project's viability

- Economic feasibility: Positive cashflow (31,000,000 VND during development, 3,000,000 VND/month post-project) and a 4-month break-even point confirm financial viability
- Technical feasibility: Industry-standard technologies (Laravel framework, MySQL) and a skilled team ensure the system can be built reliably
- Organizational feasibility: A fully staffed team with clear roles supports efficient execution
- Schedule feasibility: A 6-month timeline aligns with business needs and minimizes delays
- Legal feasibility: Compliance with Vietnam's regulations ensures legal viability

These components collectively ensure the project is practical, resource-efficient, and aligned with iDuong Hotel's goals of enhancing customer experience and operational efficiency. The report identifies risks (e.g., integration failures, requirement misunderstandings) and provides mitigation strategies (e.g., rigorous testing, stakeholder consultations), fostering stakeholder confidence and positioning the project for success in Vietnam's digital tourism landscape

4. Assess the impact of different feasibility criteria on a software investigation (D2)

a. Overview of feasibility criteria

Feasibility criteria are critical in evaluating the suitability of different solutions during a software investigation. For the Booking iDuong Hotel system, the following criteria are considered:

- Cost: Includes initial development costs (e.g., developer salaries, server setup) and ongoing operational and maintenance costs (e.g., hosting, updates).
- Implementation time: The duration required to build, test, and deploy the system, impacting the project timeline.
- Risk: Potential risks related to development, security, and operational stability, such as data breaches or system downtime.
- Effectiveness: The extent to which the solution meets functional requirements (e.g., booking, payment processing) and non-functional requirements (e.g., 2-second response time, 99.9% uptime).

These criteria guide the selection of the most appropriate solution for the Booking iDuong Hotel system, ensuring alignment with business objectives and technical constraints

b. Alternative solutions for booking iDuong Hotel

Two solutions are proposed for developing the Booking iDuong Hotel system:

Solution 1: Custom Development with Laravel

This approach involves building the system from scratch using the Laravel framework, PHP, MySQL, HTML, CSS, JavaScript, and XAMPP for local development.

Figure 2 - 1: Custom Development with Laravel

Advantages:

- Freedom of design and features: Allows complete customization to meet specific requirements, such as VNPay/MoMo integration, role-based dashboards (Customer, Staff, Manager), and the iDuong Hotel color scheme (#f2b100, #fff, #1abc9c, #1e4752).
- Control over quality: Enables rigorous testing (e.g., PHPUnit, Selenium) and optimization for performance (e.g., 2-second response time) and reliability (99.9% uptime).
- Scalability: Supports future expansions (e.g., loyalty programs, additional payment gateways) through modular Laravel controllers and MySQL schemas.

Disadvantages:

- High skill requirement: Requires expertise in Laravel, PHP, and MySQL, potentially challenging for a small team (3–5 members).
- Longer development time: Custom coding and testing extend the timeline compared to pre-built solutions.

Solution 2: Using a Website Creation Tool (WordPress)

This approach uses WordPress with plugins (e.g., WooCommerce for bookings, payment plugins for VNPay) to build the system.

Figure 2 - 2:Using a website creation tool (WordPress)

Advantages:

- Fast and convenient: Pre-built themes and plugins accelerate development, enabling rapid deployment.
- Low initial effort: Requires minimal coding expertise, suitable for a small team with limited technical skills.

- Cost effective setup: Leverages existing WordPress infrastructure, reducing initial development costs.

Disadvantages:

- Limited quality control: Plugin dependencies may lead to performance issues (e.g., slower response times) or compatibility problems.
- Security risks: WordPress sites are frequent targets for attacks, requiring robust security measures to comply with Vietnam's Cybersecurity Law (2018).
- Restricted customization: Limited flexibility in tailoring features (e.g., complex payment workflows) or adhering to the iDuong Hotel color scheme

c. Feasibility matrix for booking iDuong Hotel

The following feasibility matrix compares the two solutions based on the defined criteria, providing a structured evaluation for the Booking iDuong Hotel system

Criteria	Solution 1: Custom development with Laravel	Solution 2: Using wordPress
Cost	High: Significant initial investment for developers (3–5 members), server setup (e.g., AWS), and maintenance. Estimated at 300,000,000 VND for development and 50,000,000 VND/year for operations	Low: Lower initial costs due to pre-built plugins and themes. Estimated at 100,000,000 VND for setup and 30,000,000 VND/year for hosting and plugin licenses
Implementation Time	High: 6–8 months for development, testing, and deployment due to custom coding and rigorous testing (e.g., PHPUnit, Selenium)	Average: 3–4 months for setup and configuration using WordPress plugins and themes
Risk	Low: Controlled development process with secure coding practices (e.g., PDO, CSRF tokens) and compliance with Vietnam's Cybersecurity Law (2018). Risks include developer skill gaps	Average: Higher risks due to plugin vulnerabilities, potential downtime, and limited control over third-party code
Effectiveness	High: Fully meets functional requirements (e.g., booking, payment, reporting) and non-functional requirements (e.g., 2-second response time, 99.9% uptime), with tailored UX and scalability	Average: Meets basic requirements but struggles with complex features (e.g., VNPay integration) and performance optimization

Table 2 - 1: Feasibility matrix for booking iDuong Hotel

d. Impact of feasibility criteria on software investigation

Each feasibility criterion significantly impacts the software investigation process for the Booking iDuong Hotel system, influencing the choice of solutions and project outcomes

Cost

Impact: Cost is a critical factor, as the project must align with the goal of reducing operational costs by 20% (100,000,000 VND/year). High initial costs for Solution 1 (custom development) require careful budgeting but offer long-term savings through optimized performance and reduced maintenance. Solution 2 (WordPress) has lower initial costs but higher ongoing expenses due to plugin licenses and security updates.

Investigation implication: The software investigation must assess the hotel's budget (e.g., revenue supports 300,000,000 VND development cost) and long-term cost benefits. Custom development requires detailed cost estimation in the Requirements Analysis phase, tracked via Trello cards (e.g., "Estimate Development Budget").

Example: Solution 1's use of open-source tools (Laravel, MySQL, XAMPP) reduces licensing costs, saving 10–15% compared to WordPress's premium plugins.

Implementation Time

Impact: Implementation time affects the project's ability to meet deadlines and achieve business goals (e.g., launching before peak tourism season). Solution 1's longer timeline (6–8 months) ensures thorough development and testing but delays deployment. Solution 2's shorter timeline (3–4 months) enables faster market entry but may compromise quality.

Investigation implication: The investigation must prioritize stakeholder timelines, using Trello to schedule tasks (e.g., "Complete System Design by Month 2"). Solution 1 requires parallel task assignments to optimize the timeline for a small team.

Example: Solution 1's System Design phase, managed via Trello, ensures detailed Laravel controller designs, reducing implementation errors but extending the timeline by 2 months compared to WordPress.

Risk

Impact: Risks, such as security vulnerabilities or system downtime, threaten the system's reliability (99.9% uptime) and compliance with Vietnam's Cybersecurity Law (2018). Solution 1's controlled development minimizes risks through secure practices (e.g., HTTPS, PDO queries), while Solution 2's reliance on third-party plugins increases vulnerability risks.

Investigation implication: The investigation must include a risk assessment (e.g., via Trello card "Identify Security Risks") to evaluate vulnerabilities (e.g., SQL injection, plugin exploits) and mitigation strategies (e.g., fallback payment options for VNPay downtime).

Example: Solution 1's use of Laravel's CSRF tokens and PDO reduces security risks by 20% compared to WordPress, which requires frequent plugin updates to mitigate vulnerabilities.

Effectiveness

Impact: Effectiveness determines whether the system meets business goals (e.g., 30% booking increase, intuitive UX). Solution 1's custom development ensures tailored features (e.g., VNPay integration, #f2b100 "Book Now" button) and performance optimization, while Solution 2 struggles with complex workflows and scalability.

Investigation implication: The investigation must validate requirements against stakeholder needs (e.g., via JAD sessions, documented in RTM) to ensure alignment. Trello cards (e.g., "Validate Booking Module Requirements") track stakeholder feedback to maximize effectiveness.

Example: Solution 1's custom BookingController ensures a 2-second response time and 500 concurrent users, while WordPress's plugin-based booking may exceed 3 seconds, reducing effectiveness

e. Trello-based management for feasibility assessment

Trello is used to manage the software investigation process, ensuring thorough evaluation of feasibility criteria:

Board: "iDuong Hotel Feasibility Analysis"

Lists:

- Requirements analysis: Cards like "Define Cost Estimates" (checklist: developer salaries, server costs, maintenance budget).
- Risk assessment: Cards like "Identify Security Risks" (checklist: evaluate VNPay downtime, SQL injection risks).
- Effectiveness balidation: Cards like "Validate Booking Requirements" (checklist: confirm 2-second response time, stakeholder approval).

Example card: "Compare Solution Costs" (checklist: estimate Laravel development at 300,000,000 VND, WordPress at 100,000,000 VND; assign to project manager; due before solution selection).

Outcome: Trello ensures structured decision-making, reducing evaluation time by 15% and aligning with stakeholder goals

Based on the feasibility matrix, **Solution 1 (Custom Development with Laravel)** is the most feasible for the Booking iDuong Hotel system. It offers high effectiveness by meeting all functional requirements (e.g., booking, payment, reporting) and non-functional requirements (e.g., 2-second response time, 99.9% uptime), with tailored UX using the iDuong Hotel color scheme (#f2b100, #fff, #1abc9c, #1e4752). While it involves higher costs (300,000,000 VND) and longer implementation time (6–8 months), these are justified by the hotel's revenue and long-term savings through optimized performance and reduced maintenance. Solution 1 also minimizes risks through secure coding practices (e.g., PDO, CSRF

tokens) and compliance with Vietnam's Cybersecurity Law (2018). In contrast, **Solution 2 (WordPress)** offers lower costs and faster implementation but falls short in quality control, security, and customization, limiting its effectiveness for complex workflows like VNPay integration. Supported by authoritative references (Sommerville, 2016; Pressman, 2014), Solution 1 is the optimal choice to deliver a robust, scalable, and user-friendly system, achieving the project's business objectives in Vietnam's tourism industry

CHAPTER 3: UNDERTAKE A SOFTWARE DEVELOPMENT LIFECYCLE (LO3)

1. Undertake a software investigation to meet a business need

The Booking iDuong Hotel system is an online platform designed to facilitate room reservations, payment processing, and hotel management, aligning with Vietnam's digital transformation in the tourism industry. To ensure the system meets business objectives such as increasing online bookings, optimizing operations, and enhancing customer experience, a thorough software investigation is essential. This section provides a detailed review of requirement definitions, identifies functional (FRs) and non-functional requirements (NFRs), their relationships, and techniques for gathering requirements, tailored to the practical context of iDuong Hotel

a. Review of Requirement Definition and Stakeholders

Stakeholders play a pivotal role in shaping the requirements for the Booking iDuong Hotel system, ensuring the software aligns with business goals and delivers exceptional user experience. Each stakeholder provides specific insights, from operational processes to real-world feedback, to create a system that meets practical needs

Stakeholders and their requirements

Owner (Hotel Manager)

Role: Defines the project scope, core functionalities, and business objectives. Provides detailed information on hotel operations, such as room management, booking processes, and revenue tracking. Participates in evaluating software versions during development to ensure alignment with goals, such as a 30% increase in online bookings (approximately 500,000,000 VND annually) and a 20% reduction in operational costs (100,000,000 VND/year)

Specific requirements

- Room management: Track room status (available, booked, under maintenance) and seasonal pricing
- Payment integration: Support VNPay and MoMo
- Revenue reporting: Provide daily, weekly, and monthly revenue reports for business decision-making

Example: Requests an admin dashboard with visual revenue reports and VNPay payment integration for fast transactions

Hotel staff

Role: Provides insights into daily workflows, such as checking room availability, handling bookings, and interacting with guests. Specifies requirements for a user-friendly interface and features to streamline tasks, such as automated room status updates. Staff feedback helps reduce processing time and improve operational efficiency

Specific requirements

- Real-time room availability checking
- Automated notifications for new bookings
- Simple booking management interface to reduce processing time from 5 minutes (manual) to under 1 minute

Example: Front desk staff request a feature to display daily bookings and send automated email or app notifications

Customers (End Users)

Role: Provide feedback on user experience, focusing on ease of use, speed, and interface quality. Ensure the system meets practical needs, such as intuitive room browsing, detailed visuals, and fast payments. Customer input optimizes the user experience

Specific requirements

- Intuitive interface with high-quality room images
- Page load time under 2 seconds and payment completion within 5 seconds
- Support for viewing hotel reviews and amenities (e.g., pool, restaurant)

Example: Customers request a room filtering feature by price, location, and amenities, with a prominent “Book Now” button

Software development team

Project manager

- Role: Plans the project, assigns tasks, tracks progress, and manages resources. Ensures completion within 6 months and a 200,000,000 VND budget, coordinating with stakeholders to resolve issues
- Example: Ensures payment API integration is completed by month 4

Information collector

- Role: Engages with the owner, staff, and customers to gather detailed requirements. Creates the Software Requirements Specification (SRS), Use Case diagrams, and supporting documents to guide development
- Example: Develops a Use Case diagram for the booking process, from room selection to payment confirmation

Interface designer

- Role: Designs a user-friendly, visually appealing UI/UX, incorporating iDuong Hotel's branding (e.g., blue and white color scheme). Creates wireframes and prototypes for stakeholder validation
- Example: Designs a booking interface with a clear layout and prominent Call-to-Action (CTA) buttons

Programmers

- Role: Write code to implement features using React (frontend), Node.js, Express (backend), and MySQL (database). Ensure efficient payment API integration and data management
- Example: Develops a payment module with VNPay integration, processing transactions in under 5 seconds

Tester

- Role: Conducts unit, integration, and system testing to identify bugs, ensuring the system meets requirements and handles high loads (500 concurrent users)
- Example: Tests the booking feature to ensure no errors under simultaneous access by multiple users

Product delivery team

- Role: Delivers the software to the hotel, provides user guides, and trains staff. Ensures smooth deployment, supports staff in adopting the system, and organizes in-depth training sessions for features like booking management and reporting
- Example: Conducts a 2-day training session for front desk staff on using the room availability and payment features

b. **Identification of Functional (FRs) and Non-Functional Requirements (NFRs)**

Functional Requirements (FRs)

Functional requirements define the features users can directly interact with in the system (Dooley, 2011). The FRs for the Booking iDuong Hotel system include

Data display and synchronization

- Supports data synchronization across multiple devices (phones, tablets, laptops, PCs) for real-time access by staff and managers
- Example: Front desk staff check room status on a tablet while the manager views revenue reports on a laptop

Room management

- Manages room lists (type, price, amenities), updates status (available, booked, maintenance), and adjusts pricing seasonally
- Example: Updates room price from 1,000,000 VND/night to 1,500,000 VND/night during peak season

User management

- Manages customer accounts (registration, login, booking history) and admin accounts (roles like front desk, manager, accountant)
- Example: Restricts revenue report access to managers only

Payment

- Integrates online payment gateways (VNPay, MoMo) with instant transaction confirmation
- Example: Customers pay via VNPay and receive email confirmation within 10 seconds

Booking

- Allow customers to browse rooms, view images, prices, amenities, and book directly via the website or app
- Example: Customers select a Superior room, input check-in/check-out dates, and confirm booking in 3 steps

Revenue statistics

- Provides detailed revenue reports (daily, weekly, monthly) with visual charts to support business decisions.
- Example: Displays a December report showing 300,000,000 VND from 200 bookings

Non-Functional Requirements (NFRs)

Non-functional requirements are constraints on system performance, quality, and user experience (Dooley, 2011). The NFRs include

Intuitive user interface

- Smooth, easy to use interface with a visually appealing design, using iDuong Hotel's branding (e.g., blue and white colors). Supports multilingual options (Vietnamese, English)
- Example: Booking interface with clear layout, prominent "Book Now" button, and readable fonts

Stability

- Handles 500 concurrent users during peak hours (e.g., Tet holiday season) with an error rate below 0.1%
- Example: System remains operational when 300 customers search rooms simultaneously on December 30

Reliability and accuracy

- Ensures 99.9% accuracy for booking, payment, and revenue data to prevent errors (e.g., incorrect room status or payment amounts)
- Example: Revenue reports match actual transactions exactly

Security

- Implements SSL encryption, bcrypt hashing, and role-based access control to protect customer data (names, phone numbers, payment details)
- Example: Uses AES-256 encryption to prevent unauthorized access to customer data

Performance

- Page response time is under 2 seconds and payment transaction processing within 5 seconds, even under high load
- Example: Room listing page loads in 1.5 seconds with 100 concurrent users

Scalability

- Supports scaling from 1,000 to 10,000 daily visits using cloud hosting (e.g., AWS)
- Example: Automatically scale server resources during traffic spikes

Relationship Between FRs and NFRs

FRs and NFRs are closely interconnected, impacting system performance and user experience

- Intuitive interface and functionality: A user-friendly interface enhances booking and payment functions, enabling users to complete tasks in 3–5 steps, increasing satisfaction
- Stability and functionality: Stability ensures uninterrupted access to booking and payment features during peak times, preventing customer loss
- Reliability, accuracy, and functionality: Accurate data in room management and revenue reporting builds trust, meeting manager and customer expectations
- Security and functionality: Security measures (e.g., SSL, bcrypt) protect payment and user management functions, ensuring data safety and compliance with Vietnam's Cybersecurity Law (2018)
- Performance and functionality: Fast response times (under 2 seconds) support smooth booking and payment processes, matching platforms like Booking.com
- Scalability and functionality: Scalability ensures functions remain effective as user traffic grows, supporting long-term business growth

c. **Techniques for gathering requirements**

To gather requirements effectively for the Booking iDuong Hotel system, three primary techniques are used: Joint Application Development (JAD), interviews, and observation. These ensure comprehensive, accurate requirements aligned with real-world needs

Joint Application Development (JAD)

JAD involves collaborative group sessions with all stakeholders to gather and analyze requirements (Everett & McLeod, 2007). It fosters cooperation and creativity

Process

- Identify participants: Include the owner, front desk staff, customers, and development team (project manager, designer, programmers, tester)
- Schedule JAD sessions: Plan sessions (e.g., 2 sessions/week in the first month) to ensure active participation
- Prepare initial outline: Create a preliminary requirements document based on initial input from the owner (e.g., payment integration, revenue reporting) and the development team
- First JAD session: Introduce project goals (e.g., increase bookings, optimize operations) and discuss core requirements like booking and payment features

- Detailed JAD sessions: Use Use Case diagrams, workflow charts, and data models to clarify requirements (e.g., booking process from room selection to payment confirmation)
- Validate requirements: Share a draft SRS via Google Docs for stakeholder feedback and consensus
- Iterate if needed: Hold 1–2 additional sessions to resolve conflicts (e.g., staff requesting simpler UI than initial design)
- Finalized document: Conduct a final JAD session to approve the formal SRS, signed by the owner and project manager

Benefits

- Fosters consensus and shared understanding among stakeholders
- Reduces gathering time (from 2 months to 1 month compared to traditional methods)
- Encourages creativity, e.g., customers suggesting a booking history feature
- Ensures high-quality, business-aligned requirements

Interviews

Interviews involve direct dialogue with stakeholders to understand, detail, and validate requirements

Process

- Select interviewees

Owner: Discuss business goals and service enhancements

Staff: Explore workflows and operational challenges

Customers: Evaluate booking experience and expectations

- Design questions: Prepare open and closed questions

“Does the system need real-time room status updates?”

“What do customers expect from the booking interface (colors, layout, speed)?”

“What challenges do staff face with manual booking processes?”

- Conduct interviews: Record detailed responses using audio or notes for accuracy
- Create interview reports

Report 1: Owner

- Purpose: Identify business and customer experience requirements
- Summary: Owner requests VNPay and MoMo payment integration and detailed daily/weekly/monthly revenue reports

Report 2: Staff

- Purpose: Understand workflows and challenges
- Summary: Staff desire a simple UI, fast room availability checks, and automated booking notifications

Report 3: Customers

- Purpose: Assess booking experience
- Summary: Customers want an intuitive interface, high-quality room images, page load under 2 seconds, and fast MoMo payments

Benefits

- Provides deep insights through direct interaction, especially for diverse customer needs
- Uncovers hidden requirements, e.g., customers wanting a price comparison feature
- Validates understanding, reducing miscommunication risks
- Builds stakeholder trust through personalized engagement

Observation

Observation involves monitoring and recording behaviors and workflows in the hotel's real-world environment to understand current processes and identify improvement areas

Process

- Define goals: Observe booking, room checking, payment processes, and staff-customer interactions
- Prepare tools: Use notebooks, cameras, or software (e.g., Evernote) for detailed documentation
- Conduct Direct Observation: Monitor operations at iDuong Hotel during peak (8–10 AM) and off-peak hours, noting

How customers book rooms at the front desk

Processing time (e.g., 5 minutes for manual checks)

Issues like room status errors or delays in communication

- Document details: Record timing, actions, and challenges, e.g., "Staff take 3 minutes to check room availability manually; customers appear frustrated."
- Discuss and validate: Engage with staff and customers post-observation to confirm findings and gather additional input
- Analyze data: Identify needs (e.g., automated room checks), issues (e.g., processing delays), and improvements (e.g., faster UI)
- Repeat if needed: Conduct additional observations during high-traffic periods (e.g., holidays) to validate or expand requirements

Benefits

- Provides real-world insights into workflows and customer experiences, e.g., identifying the need for automated notifications
- Uncovers hidden requirements, like a booking history feature for easy customer reference
- Highlights specific issues, such as manual booking delays, to propose automation solutions
- Supports system design aligned with the hotel's operational environment

The software investigation for the Booking iDuong Hotel system clearly defines requirements from stakeholders, staff, customers, development team, and delivery team including FRs (booking, payment, room management, revenue statistics) and NFRs (intuitive interface, stability, security, performance, scalability). Techniques like JAD, interviews, and observation ensure comprehensive, accurate requirement gathering, with specific examples like VNPay integration and intuitive UI. The relationships between FRs and NFRs are established to optimize system performance and user experience, ensuring the project meets business goals, enhances operational efficiency, and delivers competitive value in Vietnam's digital tourism landscape

2. Use appropriate software analysis tools/techniques to carry out a software investigation and create supporting documentation

a. Use case diagram

A Use Case Diagram is a UML tool that illustrates the tasks (use cases) a system performs to meet user needs, depicting interactions between actors and the system (Sommerville, 2016). Actors are represented as stick figures, the system as a box, and use cases as ellipses inside the box. Relationships include associations (connecting actors to use cases), include (mandatory relationships between use cases), extend (optional relationships), and generalization (parent-child relationships between use cases or actors)

Use case diagram for booking iDuong Hotel

The Use Case Diagram for the Booking iDuong Hotel system captures key functionalities for three primary actors: Customer, Staff, and Manager, with relationships to clarify dependencies and optional interactions

Actors

- Customer: End user who books rooms, makes payments, and views hotel details.
- Staff: Front desk personnel who manage bookings and check room availability.
- Manager: Hotel owner or administrator who oversees room management, revenue reports, and user accounts

Use cases

- Browse booms: Customers view room details (e.g., type, price, amenities, images).
- Book boom: Customers reserve a room by selecting dates and confirming payment (includes Make Payment).
- Make payment: Customers process payments via integrated gateways (e.g., VNPay, MoMo).
- Checkroom availability: Staff verify room status in real time (extends Book Room for manual checks).
- Manage bookings: Staff update or cancel bookings (includes Check Room Availability).
- Manage rooms: Manager updates room details (create, update) (includes Create Room, Update Room).
- Generate revenue reports: Manager views daily, weekly, or monthly revenue statistics.
- Manage users: Manager oversees customer and staff accounts (includes Create User, Update User)

Relationships

- Include: Book Room includes Make Payment (mandatory for booking completion).
- Extend: Check Room Availability extends Book Room (optional for manual staff intervention).
- Generalization: Manage Rooms generalize Create Room and Update Room (sub-use cases)

Diagram description

- The system is a box labeled “Booking iDuong Hotel System.”

- Actors (Customer, Staff, Manager) are stick figures outside the box, connected to use cases via solid lines (associations).
- Use cases are ellipses inside the box, linked by dashed lines for include (e.g., Book Room → Make Payment) and extend (e.g., Check Room Availability → Book Room) relationships.
- Generalization is shown with a hollow arrow (e.g., Manage Rooms → Create Room)

Use case specifications

Below are detailed specifications for two critical use cases: Create room and update room

Use Case 1: Create Room

Use Case ID: UC001

Use Case Name: Create Room

Description: Allows the Manager to add a new room to the system's database, making it available for booking.

Actor: Manager

Priority: High

Trigger: Manager selects the "Add New Room" option in the admin dashboard.

Pre-Conditions:

- Manager is logged in with admin privileges.
- System is connected to the MySQL database.
- Access is verified via Role-Based Access Control (RBAC).

Post-Conditions: The new room is added to the database and displayed on the website.

Basic Flow:

- Manager selects "Add New Room" in the admin dashboard.
- System displays a form for entering room details (e.g., room number, type, price, amenities, images).
- Manager enters details (e.g., Room R101, type Superior, price 1,500,000 VND/night, amenities Wi-Fi, balcony) and submits the form.
- System validates input (e.g., unique room number, positive price, valid image format).

- System saves the room to the database and displays a success message: “Room R101 added successfully.”

Exception Flow:

- If input is incomplete or invalid (e.g., missing room number, negative price):

System displays an error message (“Please enter a valid room number”) and highlights problematic fields.

- If database save fails (e.g., server connectivity issue):

System displays an error message (“Server error, please try again”) and prompts Manager to retry.

Business Rules:

- Room number must be unique in the database.
- Price must be a positive value in VND.
- Only Managers with admin privileges can create rooms.
- Room images must be in JPG/PNG format and under 5MB.

Non-Functional Requirements (Pressman, 2014):

- System must handle up to 50 room additions per hour without performance degradation.
- Form submission response time under 2 seconds with 100 concurrent users.
- 99.9% reliability to prevent storage errors

Use Case 2: Update Room

Use Case ID: UC002

Use Case Name: Update Room

Description: Allows the Manager to edit existing room details in the system’s database.

Actor: Manager

Priority: High

Trigger: Manager selects the “Edit Room” option for a specific room in the admin dashboard.

Pre-Conditions:

- Manager is logged in with admin privileges.

- Target room exists in the database.
- System is connected to the database.

Post-Conditions: The room's details are updated in the database and reflected on the website.

Basic Flow:

- Manager selects "Edit Room" for a specific room (e.g., R101).
- System displays a pre-filled form with current room details (e.g., room number, type, price, amenities).
- Manager edits details (e.g., changes price from 1,500,000 VND/night to 1,800,000 VND/night) and submits the form.
- System validates input (e.g., ensures room number remains unique, price is positive).
- System updates the room in the database and displays a success message: "Room R101 updated successfully."

Exception Flow:

- If input is invalid (e.g., room number duplicates another room):

System displays an error message ("Room number already exists") and highlights problematic fields.

- If database update fails (e.g., server error):

System displays an error message ("Server error, please try again") and prompts Manager to retry.

Business Rules:

- Room number must remain unique.
- Only Managers with admin privileges can update rooms.
- Price must be a positive value in VND.
- New images (if provided) must be in JPG/PNG format and under 5MB.

Non-Functional Requirements:

- System must handle up to 50 room updates per hour without performance degradation.
- Form submission response time under 2 seconds with 100 concurrent users.
- 99.9% reliability to ensure accurate updates.

b. Context diagram

A Context Diagram is a high-level Data Flow Diagram (DFD) that represents the entire system as a single process interacting with external entities (Sommerville, 2016). It shows data flows to and from the system without including internal data stores, providing a clear view of the system's boundaries and external interactions

Context Diagram for Booking iDuong Hotel

The Context Diagram depicts the Booking iDuong Hotel system as a single process interacting with three external entities: Customer, Staff, and Manager.

System: Booking iDuong Hotel System (central circle).

External Entities:

- Customer: Sends booking requests and payment details, receives booking confirmations.
- Staff: Inputs booking updates, checks room availability, and receives system notifications.
- Manager: Inputs room and user management data, requests revenue reports, and receives report outputs.

Data Flows:

- Customer → System: Booking Request (e.g., room type, dates), Payment Details (e.g., VNPay transaction data).
- System → Customer: Booking Confirmation (e.g., email receipt), Room Details (e.g., images, prices).
- Staff → System: Booking Updates (e.g., cancellations), Room Availability Query.
- System → Staff: Availability Status, Booking Notifications.
- Manager → System: Room Data (e.g., price updates), User Data (e.g., new staff account), Report Request.
- System → Manager: Revenue Reports, User Management Confirmation.

Diagram Description:

- The system is a circle labeled "Booking iDuong Hotel System."
- External entities (Customer, Staff, Manager) are rectangles outside the circle.

- Data flows are arrows labeled with data types (e.g., “Booking Request,” “Revenue Reports”) connecting entities to the system

c. Data flow diagram – Level 0

A Level 0 Data Flow Diagram (DFD) breaks down the system into major processes, showing data stores, external entities, and data flows between them (Pressman, 2014). It provides a high-level view of system processes and their interrelationships, organizing the system without delving into detailed subprocesses

Level 0 DFD for Booking iDuong Hotel

The Level 0 DFD includes key processes, data stores, and external entities for the Booking iDuong Hotel system.

External Entities: Customer, Staff, Manager (same as Context Diagram).

Processes:

- Process 1: Manage Bookings: Handles booking creation, updates, and cancellations.
- Process 2: Process Payments: Integrates payment gateways for transaction processing.
- Process 3: Manage Rooms: Manages room data (create, update).
- Process 4: Manage Users: Handles customer and staff account management.
- Process 5: Generate Reports: Produces revenue and booking statistics.

Data Stores:

- D1: Rooms Database: Stores room details (e.g., room number, type, price, status).
- D2: Bookings Database: Stores booking records (e.g., customer ID, dates).
- D3: Users Database: Stores customer and staff account details.
- D4: Transactions Database: Stores payment records.

Data Flows

- Customer → Manage Bookings: Booking Request.
- Manage Bookings → Customer: Booking Confirmation.
- Customer → Process Payments: Payment Details.
- Process Payments → Customer: Payment Confirmation.
- Staff → Manage Bookings: Booking Updates, Availability Query.

- Manage Bookings → Staff: Availability Status, Notifications.
- Manager → Manage Rooms: Room Data.
- Manage Rooms → Rooms Database: Store/Update Room Data.
- Manager → Manage Users: User Data.
- Manage Users → Users Database: Store/Update User Data.
- Manager → Generate Reports: Report Request.
- Generate Reports → Manager: Revenue Reports.
- Rooms Database ↔ Manage Bookings: Room Status.
- Bookings Database ↔ Manage Bookings: Booking Data.
- Transactions Database ↔ Process Payments: Transaction Data.
- Users Database ↔ Manage Users: User Data.

Diagram Description

- Processes are circles labeled with process names (e.g., "Manage Bookings").
- External entities are rectangles, connected to processes via labeled arrows (data flows).
- Data stores are open-ended rectangles (e.g., "D1: Rooms Database") linked to processes.
- Data flows are arrows showing data movement (e.g., "Booking Request" from Customer to Manage Bookings)

d. Entity Relationship Diagram (ERD)

An Entity Relationship Diagram (ERD) visually represents the database structure, showing entities, their attributes, and relationships (Sommerville, 2016). It supports modeling data storage and retrieval, ensuring the Booking iDuong Hotel system's database meets functional needs like booking and payment processing.

ERD for Booking iDuong Hotel

The ERD includes key entities, attributes, and relationships to support system functionalities.

Entities and attributes:

Room:

- RoomID (Primary Key): Unique identifier (e.g., R001).
- Type: Room category (e.g., Superior, Deluxe).

- Price: Cost per night (e.g., 1,500,000 VND).
- Status: Availability (e.g., Available, Booked, Maintenance).
- Amenities: Features (e.g., Wi-Fi, balcony).

Booking:

- BookingID (Primary Key): Unique identifier (e.g., B001).
- CustomerID (Foreign Key): Links to Customer.
- RoomID (Foreign Key): Links to Room.
- CheckInDate: Start date of stay.
- CheckOutDate: End date of stay.
- TotalCost: Total booking amount.

Customer:

- CustomerID (Primary Key): Unique identifier (e.g., C001).
- Name: Full name.
- Email: Contact email.
- Phone: Contact number.

Staff:

- StaffID (Primary Key): Unique identifier (e.g., S001).
- Name: Full name.
- Role: Job role (e.g., Front Desk, Manager).
- Password: Hashed login credential (bcrypt).

Transaction:

- TransactionID (Primary Key): Unique identifier (e.g., T001).
- BookingID (Foreign Key): Links to Booking.
- Amount: Payment amount.
- PaymentMethod: Gateway used (e.g., VNPay, MoMo).
- Status: Transaction status (e.g., Completed, Failed).

Relationships:

- Room ↔ Booking: One-to-Many (one room can have multiple bookings, one booking is for one room).
- Customer ↔ Booking: One-to-Many (one customer can have multiple bookings, one booking belongs to one customer).
- Booking ↔ Transaction: One-to-One (one booking has one transaction).
- Staff ↔ Booking: One-to-Many (one staff member can manage multiple bookings).

Diagram Description:

- Entities are rectangles with attributes listed inside (e.g., Room: RoomID, Type, Price).
- Primary keys are underlined (e.g., RoomID).
- Relationships are diamonds or lines connecting entities, labeled with cardinality (e.g., 1:N for Room ↔ Booking).
- Foreign keys are indicated in related entities (e.g., Booking.RoomID references Room.RoomID).

The software investigation for the Booking iDuong Hotel system employs four key analysis tools: Use Case Diagram, Context Diagram, Data Flow Diagram (Level 0), and Entity Relationship Diagram. These tools define system functionalities (e.g., booking, payment, room management), data flows (e.g., booking requests, revenue reports), and database structure (e.g., Room, Booking entities). The Use Case Diagram details interactions for Customers, Staff, and Managers, with specifications for Create Room and Update Room. The Context Diagram and Level 0 DFD clarify processes and data flows, while the ERD ensures a robust database design. Supported by authoritative references (Sommerville, 2016; Pressman, 2014), these documents guide development, ensuring the system meets business goals, enhances operational efficiency, and delivers a competitive user experience in Vietnam's tourism industry

3. Analyse how software requirements can be traced throughout the software lifecycle

Overview of requirements management

Requirements management involves systematically handling requirements throughout the software development lifecycle to ensure the product meets stakeholder needs. According to Sommerville (2016), its main purposes include:

- Ensuring alignment with business goals: Documenting and prioritizing requirements to meet objectives like increased bookings and operational efficiency.
- Maintaining current requirements: Analyzing and updating requirements to reflect changes, ensuring engineering teams work with approved specifications.

- Facilitating testing and validation: Ensuring all requirements are testable and traced through development and testing phases.
- Managing risks: Identifying and addressing missing or unclear requirements to minimize project risks.
- Supporting documentation: Creating clear documentation (e.g., BRD, TRD) to guide development and project management.

The Requirements Traceability Matrix (RTM) is a critical tool that maps business and technical requirements to test cases, ensuring full coverage and traceability. It supports two types of traceability:

- Forward traceability: Maps requirements to design, implementation, and test cases to ensure all requirements are addressed.
- Backward traceability: Tracks test results back to requirements to verify compliance and avoid scope creep.
- Purposes of RTM for iDuong Hotel:
- Build customer trust: Ensures the system is developed and tested per stakeholder requirements, fostering confidence.
- Ensure complete coverage: Guarantees all requirements are covered in test design and execution, achieving 100% test case coverage.
- Track requirement changes: Maintains an audit trail of requirement updates throughout the project lifecycle.
- Manage risks: Ensures each requirement is testable and tested, reducing the risk of undetected defects.
- Identify missing functionality: Detects gaps in functionality during development and recommends solutions.
- Support documentation: Aids in creating requirements documents, specifications, and project plans.
- Maintain requirement clarity: Flags unclear requirements and tracks updates for test design alignment.
- Prevent scope creep: Uses backward traceability to ensure testing aligns with approved requirements.

- Evaluate product quality: Assesses quality through test coverage and error status, identifying high-risk components for closer inspection

Requirements Management Process for Booking iDuong Hotel

The requirements management process for the Booking iDuong Hotel system follows a structured approach to ensure alignment with business needs and stakeholder expectations. The process includes:

- Requirement elicitation: Gather requirements from stakeholders (Customers, Staff, Manager) using techniques like interviews, Joint Application Development (JAD), and observation (see previous P5 artifact).
- Requirement documentation: Create Business Requirement Document (BRD) and Technical Requirement Document (TRD) to formalize requirements.
- Requirement analysis and prioritization: Analyze requirements for feasibility and prioritize based on business impact (e.g., booking and payment functions are high priority).
- Requirement validation: Validate requirements with stakeholders to ensure clarity and completeness.
- Requirement traceability: Develop an RTM to map requirements to test cases, ensuring traceability throughout development and testing.
- Change management: Track and update requirements in the RTM to reflect changes, ensuring alignment with project goals.
- Testing and verification: Use the RTM to verify that all requirements are implemented and tested, ensuring system quality

Supporting documentation

a. Business Requirement Document (BRD)

The BRD outlines the high-level business needs and functionalities required for the Booking iDuong Hotel system, addressing the needs of Customers, Staff, and Managers

BRD_ID	Module name	Applicable roles	Description
1	Login	Customer, Staff, Manager	Customers log in to access booking features. Staff and Managers log in to access role-specific dashboards (e.g., booking management for Staff, revenue reports for Manager)
2	Logout	Customer, Staff, Manager	Users log out to end their session securely
3	Create Room	Manager	Manager adds a new room (e.g., type, price, amenities) to the database

4	Update Room	Manager	Manager edits existing room details (e.g., price, status)
5	Delete Room	Manager	Manager removes a room from the database
6	Book Room	Customer	Customers browse rooms, select dates, and book a room with payment
7	Make Payment	Customer	Customers process payments via integrated gateways (e.g., VNPay, MoMo)
8	View Room Details	Customer, Staff, Manager	Users view detailed room information (e.g., images, price, amenities)
9	Manage Bookings	Staff	Staff update or cancel bookings, check room availability
10	Generate Revenue Report	Manager	Manager views daily, weekly, or monthly revenue statistics with charts

Table 3 - 1:Business requirement document

b. Technical Requirement Document (TRD)

The TRD specifies the technical specifications required to implement the business requirements, focusing on system functionality and integration

TRD_ID	Technical requirement
1	Validate user credentials (username, password), check role (Customer, Staff, Manager), and redirect to role-specific dashboard (e.g., booking for Customers, admin for Managers)
2	Display a form for adding a new room, validate input (e.g., unique room number, positive price), and store room data in the MySQL database
3	Display a pre-filled form for editing room details, validate input, and update room data in the database
4	Display a confirmation interface for deleting a room, validate the room ID, and remove the room from the database
5	Retrieve and display room details (e.g., type, price, images) from the database for browsing
6	Process booking requests, validate dates and room availability, and store booking data in the database
7	Integrate payment gateways (VNPay, MoMo), process transactions, and store transaction details in the database
8	Validate user existence and securely terminate the session upon logout
9	Retrieve booking data, allow updates or cancellations, and sync with room availability in the database
10	Generate revenue reports with aggregated data (e.g., daily, weekly, monthly) and visual charts, retrieved from the database

Table 3 - 2: Technical requirement document

c. Test Case Table

The Test Case Table links business and technical requirements to specific test scenarios, ensuring all requirements are tested

BRD_ID	TRD_ID	TC_ID	Test Case	Test Steps	Test Data	Expected Result
1	1	1	Login with invalid account	1. Open login page 2. Enter invalid username/password 3. Click Login	Username: invalid_user Password: 12345	Error: “Invalid username or password”
2	1	1	Login with valid Manager account	1. Open login page 2. Enter valid Manager credentials 3. Click Login	Username: admin Password: admin123	Redirect to Manager admin dashboard
3	3	2	Add new room	1. From admin dashboard, click “Add New Room” 2. Enter room details 3. Click Submit	RoomID: R101 Type: Superior Price: 1,500,000 VND Amenities: Wi-Fi, Balcony	Success: “Room R101 added successfully”
4	4	3	Update room	1. From admin dashboard, select “Edit Room” for R101 2. Update price 3. Click Submit	RoomID: R101 New Price: 1,800,000 VND	Success: “Room R101 updated successfully”
5	5	4	Delete room	1. From admin dashboard, select “Delete Room” for R101 2. Confirm deletion	RoomID: R101	Success: “Room R101 deleted successfully”
6	6	6	Book room	1. Open booking page 2. Select room and dates 3. Click Book	RoomID: R101 CheckIn: 2025-12-01 CheckOut: 2025-12-03	Success: “Booking confirmed”
7	7	7	Process payment	1. From booking page, select payment method 2. Enter payment details 3. Submit	PaymentMethod: VNPay Amount: 3,000,000 VND	Success: “Payment completed”
8	8	5	View room details	1. Open room listing page 2. View room details	RoomID: R101	Displays room details (type, price, images)
9	9	9	Update booking	1. From staff dashboard, select booking	BookingID: B001 New CheckOut: 2025-12-04	Success: “Booking updated”

				2. Update dates 3. Submit		
10	10	10	Generate revenue report	1. From admin dashboard, select report type 2. View report	ReportType: Monthly Period: Dec 2025	Displays report with charts (e.g., 300M VND)

Table 3 - 3: Test case table

d. Requirements Traceability Matrix (RTM)

The RTM maps business and technical requirements to test cases, ensuring full traceability and coverage

Test Case ID	Business Requirement ID	Technical Requirement ID
1	1	1
2	1	1
3	3	2
4	4	3
5	5	4
6	6	6
7	7	7
8	8	5
9	9	9
10	10	10

Table 3 - 4: Requirements traceability matrix

e. How to Use the RTM for Booking iDuong Hotel

The RTM is used throughout the project lifecycle to ensure requirements are implemented, tested, and validated. The process includes:

Develop BRD and TRD:

- Create the BRD to capture stakeholder needs (e.g., booking, payment, room management).
- Create the TRD to specify technical implementations (e.g., database storage, payment gateway integration).

Design test cases: Develop test cases (Table 3-3) to cover each BR and TR, ensuring all functionalities are tested (e.g., login, room creation, payment processing).

Build RTM: Map test cases to BR and TR IDs in the RTM (Table 3-4) to ensure traceability.

Track changes: Update the RTM when requirements change (e.g., adding a new payment gateway like Viettel Pay), ensuring test cases are adjusted.

Verify coverage: Use the RTM to confirm 100% test coverage, ensuring no requirement is untested.

Validate implementation: Execute test cases and use backward traceability to verify that each requirement is met (e.g., successful room creation matches BRD_ID#3).

Manage risks: Identify untested or unclear requirements in the RTM and address them (e.g., flag missing validation for room price).

Support project management: Use the RTM to generate reports for stakeholders, ensuring alignment with business goals (e.g., revenue reporting accuracy)

Requirements management for the Booking iDuong Hotel system ensures that the platform meets business objectives through systematic documentation, analysis, and traceability. The BRD and TRD define business and technical requirements, covering critical functionalities like booking, payment, and room management. The Test Case Table and RTM ensure all requirements are traceable to test cases, guaranteeing full coverage and validation. By leveraging the RTM, the project team can build stakeholder trust, manage risks, detect missing functionalities, and maintain alignment with goals like increasing online bookings and operational efficiency. Supported by authoritative references (Sommerville, 2016; Pressman, 2014), this process ensures the system delivers a competitive user experience in Vietnam's tourism industry, compliant with standards like the Cybersecurity Law (2018)

4. Discuss two approaches to improving software quality

Software quality attributes for booking iDuong Hotel

Software quality is characterized by multiple attributes that ensure the system meets both functional and non-functional requirements. The following attributes are critical for the Booking iDuong Hotel system:

Usability:

- Correctness: The system accurately performs tasks such as booking rooms, processing payments via VNPay/MoMo, and generating revenue reports.
- Completeness: All specified functionalities (e.g., login, room management, payment processing) are fully implemented.

Compatibility:

- System compatibility: Supports various platforms, including Windows, macOS, and Linux servers, using XAMPP and Laravel.
- Browser and device compatibility: Functions seamlessly across browsers (Chrome, Firefox, Safari) and devices (PCs, tablets, smartphones) with responsive HTML/CSS/JavaScript interfaces.

Performance:

- Reliability: Operates consistently without crashes, handling up to 500 concurrent users with 99.9% uptime.
- Efficiency: Processes requests (e.g., booking, payment) in under 2 seconds, optimizing user experience.

Security: Data security: Protects user data (e.g., customer details, payment information) with HTTPS, CSRF tokens, and bcrypt hashing, complying with Vietnam's Cybersecurity Law (2018).

Usability:

- Learnability: Intuitive interfaces (e.g., booking forms, room management tables) allow users to learn quickly.
- User experience: Provides a seamless experience with the iDuong Hotel color scheme (light gold #f2b100, white #fff, green #1abc9c, teal #1e4752).

Scalability and Maintainability:

- Maintainability: Easy to update and fix, with modular Laravel code and clear documentation.
- Scalability: Supports traffic growth from 1,000 to 10,000 daily visits using cloud servers (e.g., AWS).

Compliance:

- Legal compliance: Adheres to Vietnam's Cybersecurity Law (2018) and data protection regulations.
- Coding standards: Follows PHP and Laravel coding conventions for consistency.

Cost-Effectiveness: Economic development: Uses open-source tools (XAMPP, MySQL, Laravel) to minimize development and maintenance costs

Quality Assurance (QA) and Quality Control (QC)

To achieve these quality attributes, two primary quality management approaches Quality Assurance (QA) and Quality Control (QC) are employed. These approaches ensure the Booking iDuong Hotel system meets stakeholder requirements and delivers a reliable, user-friendly product

a. Quality Assurance (QA)

QA is a proactive process within quality management that focuses on building confidence that quality requirements will be met through systematic activities (ASQ, 2023). It emphasizes improving the development process to prevent defects before they occur

Definition: QA involves planned activities within the quality system to ensure the development process adheres to standards, reducing errors and ensuring the product meets requirements.

Activities:

- Defining development processes (e.g., coding standards for Laravel).
- Conducting peer reviews of code and designs.
- Implementing automated testing frameworks (e.g., PHPUnit for Laravel).
- Establishing requirements traceability using the RTM (see previous artifact).
- Training the development team (3–5 members) on best practices.

Objective: Prevent errors by optimizing processes, ensuring the system meets functional (e.g., booking, payment) and non-functional (e.g., 2-second response time) requirements.

Example for iDuong Hotel:

- Process standardization: Define coding standards (e.g., PSR-12 for PHP) to ensure consistent Laravel code.
- Peer reviews: Review Blade templates and controllers to catch logical errors early.
- Automated testing: Use PHPUnit to test Laravel routes and controllers (e.g., BookingController) for reliability.
- Training: Conduct workshops on secure coding (e.g., PDO for MySQL queries) to comply with Vietnam's Cybersecurity Law (2018)

b. Quality Control (QC)

QC is a reactive process within quality management that focuses on verifying that the final product meets quality requirements through inspection and testing (ASQ, 2023). It identifies and corrects defects in the delivered product

QC involves operational techniques (e.g., testing, code inspection) to ensure the product meets quality standards, such as functionality, performance, and security.

Activities:

- Executing test cases (unit, integration, system, acceptance) based on the RTM.
- Inspecting code for errors (e.g., SQL injection vulnerabilities).
- Validating user interfaces (e.g., booking page) for usability and responsiveness.
- Monitoring performance metrics (e.g., page load time, error rates).

Objective: Identify and fix defects in the final product, ensuring it meets requirements (e.g., accurate booking, secure payments).

Example for iDuong Hotel:

- Unit Testing: Test individual Laravel controllers (e.g., RoomController) using PHPUnit to verify room creation logic.
- Integration Testing: Validate interactions between Laravel backend and VNPay API for payment processing.
- System Testing: Test the entire system (e.g., booking flow from room selection to payment) for reliability and performance.
- Usability Testing: Ensure the booking page (with #f2b100 "Book Now" button) is intuitive across devices

Comparison of QA and QC

QA and QC are complementary approaches that together ensure the quality of the Booking iDuong Hotel system. Below is a detailed comparison based on key characteristics

Characteristics	Quality Assurance (QA)	Quality Control (QC)
Objective	Ensure the quality of the development process to prevent errors	Ensure the final product meets quality requirements by identifying and fixing defects
Key Stages	Design, planning, development, and testing phases	Testing and deployment phases
Predictive Nature	Proactive: Prevents errors before product development	Reactive: Detects errors after product development
Methodology	Focuses on processes and quality standards (e.g., coding conventions, peer reviews)	Focuses on testing and inspecting product quality (e.g., unit tests, system tests)
Responsibility	Shared by the entire development team (e.g., developers, designers)	Primarily the testing team's responsibility
Type of Error Detection	Detects errors at the process and design level (e.g., flawed requirements)	Detects errors at the product and code level (e.g., bugs in booking logic)
Automation	Involves automating processes (e.g., CI/CD pipelines with Laravel Forge)	Focuses on test automation (e.g., PHPUnit, Selenium for UI testing)
Time of Error Detection	Before errors appear in the product (e.g., during code reviews)	After errors appear in the product (e.g., during system testing)
Orientation	Process-oriented: Improves development standards and workflows	Product-oriented: Ensures the final system meets quality requirements

Table 3 - 5: Comparison of QA and QC

Similarities

- Both aim to ensure high software quality for the Booking iDuong Hotel system.
- Both involve detecting and addressing errors to enhance reliability and user satisfaction.
- Both are integral to the development lifecycle, supporting goals like 99.9% uptime and 2-second response times

Differences

- QA focuses on preventing errors through process optimization, while QC focuses on detecting errors in the final product.
- QA is proactive and spans the entire lifecycle, while QC is reactive and focuses on testing and deployment.
- QA involves the whole team, while QC is primarily handled by testers

Application of QA and QC in Booking iDuong Hotel using Trello

To ensure the quality of the Booking iDuong Hotel system, QA and QC activities are managed using Trello, an online project management tool with a card-based interface for organizing tasks and tracking progress. Trello boards, lists, and cards help the development team (3–5 members) coordinate QA and QC efforts, ensuring alignment with business goals and quality attributes

QA Activities and Trello Management

QA activities focus on improving the development process to prevent defects. These are organized in Trello as follows

Trello board structure

- Board: iDuong Hotel QA
- Lists:

  Design and planning: Tasks for defining requirements and processes.

  Development: Tasks for coding standards and peer reviews.

  Testing: Tasks for setting up automated testing frameworks.
- Cards

Example: “Define PSR-12 Coding Standards” (Design and Planning list, assigned to lead developer, due before coding).

Example: “Conduct Peer Review of BookingController” (Development list, assigned to developers, includes checklist for code quality).

Example: “Set Up PHPUnit for Laravel” (Testing list, assigned to QA engineer, includes setup instructions)

QA stages

Design and Planning Phase:

- Define requirements using RTM (see previous artifact) to ensure traceability.
- Establish coding standards (e.g., PSR-12 for PHP, Laravel conventions).
- Create wireframes and mockups (using Figma, with #f2b100 buttons) to validate UX early.
- Trello Card: “Review RTM for Booking Requirements” (ensure all requirements are testable, assigned to project manager).

Development Phase:

- Conduct peer reviews of Laravel controllers and Blade templates to catch logical errors.
- Implement CI/CD pipelines (e.g., Laravel Forge) for automated code quality checks.
- Trello card: “Peer Review RoomController Code” (check for CSRF protection, PDO usage, assigned to developers).

Testing Phase:

- Set up automated testing with PHPUnit for unit tests (e.g., testing RoomController’s create method).
- Validate non-functional requirements (e.g., 2-second page load, 99.9% reliability).
- Trello Card: “Configure PHPUnit for Booking Tests” (includes test case setup, assigned to QA engineer).

Example trello workflow:

- Card: “Validate Booking Form Design” (Design and Planning list).

  Checklist: Ensure #f2b100 “Book Now” button, responsive layout, CSRF token inclusion.

  Assigned to: UI/UX designer, developer.

  Due Date: Before development sprint.

  Outcome: Approved wireframe reduces UI revisions.

QC Activities and Trello Management

QC activities focus on testing the final product to identify and fix defects. These are managed in Trello as follows

Trello Board Structure:

- Board: iDuong Hotel QC
- Lists:

  Testing: Tasks for executing test cases (unit, integration, system, acceptance).

  Deployment: Tasks for validating deployed system functionality.

  Operations management: Tasks for monitoring post-deployment issues.

- Cards

  Example: “Execute Unit Tests for BookingController” (Testing list, assigned to tester, linked to RTM test cases).

  Example: “Validate VNPay Integration” (Testing list, includes API response time checks).

  Example: “Monitor Booking Page Performance” (Operations Management list, tracks 2-second load time).

QC Stages

- Testing Phase:

  Execute test cases from the RTM (see previous artifact) using PHPUnit and Selenium for UI testing.

  Validate functionality (e.g., booking, payment, room management) and non-functional requirements (e.g., security, performance).

  Trello card: “Run System Tests for Booking Flow” (test booking from room selection to VNPay payment, assigned to tester).

- Deployment Phase:

  Verify deployed system on XAMPP or cloud server (e.g., AWS) for reliability and responsiveness.

  Conduct acceptance testing with stakeholders (e.g., hotel manager) to confirm usability.

  Trello card: “Perform Acceptance Testing for Manager Dashboard” (validate revenue report generation, assigned to QA lead).

- Operations Management Phase:

Monitor error logs and performance metrics (e.g., error rate <0.1%, uptime 99.9%).

Address post-deployment bugs (e.g., payment failures) via hotfixes.

Trello Card: "Monitor Payment Processing Errors" (track VNPay API issues, assigned to operations team).

Example Trello Workflow

- Card: "Test Booking Page Responsiveness" (Testing list).

  Checklist: Verify layout on Chrome, Firefox, mobile devices; check #f2b100 button visibility.

  Assigned to: Tester, UI developer.

  Due Date: Before deployment.

  Outcome: Identifies and fixes layout issues, ensuring cross-device compatibility

Integration with Booking iDuong Hotel

QA in iDuong Hotel:

- Ensures robust processes, such as secure coding with PDO and CSRF protection in Laravel.
- Reduces defects by validating wireframes (e.g., booking page layout) and requirements early.
- Example: Peer reviews catch invalid date validation in BookingController before coding completes.

QC in iDuong Hotel:

- Verifies functionality (e.g., room booking, VNPay payment) through RTM-based test cases.
- Ensures non-functional requirements (e.g., 2-second response time, 500 concurrent users) via system testing.
- Example: System testing identifies a bug in the payment confirmation flow, fixed before deployment.

Trello Benefits:

- Organizes QA/QC tasks with clear assignments, deadlines, and checklists.
- Tracks progress (e.g., 80% test case completion) and ensures stakeholder alignment.
- Reduces risks by flagging issues early (e.g., missing CSRF tokens in forms)

Benefits of QA and QC for iDuong Hotel

The flexible combination of QA and QC enhances the quality of the Booking iDuong Hotel system by addressing both process and product quality, delivering the following benefits:

Improved development process:

- QA standardizes processes (e.g., PSR-12 coding, CI/CD pipelines), reducing development errors and ensuring consistency in Laravel code.
- Example: Automated PHPUnit tests catch logical errors in RoomController early, saving 20% of debugging time.

Enhanced product quality:

- QC verifies that all functionalities (e.g., booking, payment, room management) meet requirements via RTM-based testing.
- Example: Integration testing ensures VNPay payments complete in under 5 seconds, meeting performance goals.

User satisfaction:

- QA ensures intuitive interfaces (e.g., #f2b100 "Book Now" button) through early UX validation.
- QC confirms usability across devices, enhancing customer experience (e.g., responsive booking page).

Risk reduction:

- QA prevents errors through peer reviews and process checks, reducing the likelihood of critical bugs.
- QC identifies defects (e.g., payment failures) before deployment, minimizing post-release fixes.
- Example: QC testing catches a database query error, preventing booking failures.

Cost efficiency:

- QA reduces rework by catching issues early (e.g., during wireframe reviews), saving 15–20% of development costs.
- QC minimizes post-deployment bug fixes, reducing maintenance costs by 10–15%.

Compliance and reliability:

- QA ensures compliance with Vietnam's Cybersecurity Law (2018) through secure coding practices (e.g., HTTPS, bcrypt).
- QC validates reliability (e.g., 99.9% uptime) and security (e.g., no SQL injection vulnerabilities).

Scalability and maintainability:

- QA's modular Laravel code and documentation ease future updates (e.g., adding new payment gateways).
- QC ensures scalability by testing under load (e.g., 500 concurrent users), supporting traffic growth.

Quality Assurance (QA) and Quality Control (QC) are critical approaches to ensuring the quality of the Booking iDuong Hotel system. QA proactively improves the development process through standardized coding, peer reviews, and automated testing, preventing errors early. QC reactively verifies the final product through rigorous testing, ensuring it meets functional (e.g., booking, payment) and non-functional (e.g., 2-second response time, 99.9% reliability) requirements. Using Trello to manage QA and QC activities provides a structured approach to task organization, progress tracking, and stakeholder alignment. The combination of QA and QC enhances the system's reliability, usability, security, and scalability, aligning with business goals like increasing online bookings and reducing operational costs. Supported by authoritative references (Sommerville, 2016; Pressman, 2014), these approaches ensure the Booking iDuong Hotel system delivers a competitive, user-friendly experience in Vietnam's tourism industry, compliant with regulations like the Cybersecurity Law (2018)

ASQ. (2023). *What is Quality Assurance?* American Society for Quality. Retrieved

5. Evaluate the process of undertaking a systems investigation about its effectiveness in improving software quality

a. Overview of systems investigation and the function design paradigm

The systems investigation process is the initial phase of the software development lifecycle, aimed at gathering and analyzing stakeholder requirements to define the system's scope, functionality, and constraints (Sommerville, 2016). The Function Design Paradigm provides a structured approach to identifying and organizing system functions, enabling developers and stakeholders to understand component interactions clearly

[Stakeholders] --> [Define Functional Requirements] --> [Organize System Components] --> [Validate Interactions]

Figure 3 - 1: Function design paradigm

For Booking iDuong Hotel, the systems investigation leverages the Function Design Paradigm to identify core functions (e.g., Book Room, Manage Rooms, Process Payments) and ensure they meet functional (e.g., intuitive user interfaces) and non-functional requirements (e.g., 2-second response time, 99.9% uptime)

Goals of the Function Design Paradigm

The Function Design Paradigm aims to organize and detail the specific functions of a system. Its goals include:

- System identification: Helps stakeholders and development teams understand the system's components and functions, aiding development and maintenance.
- Logical organization: Provides a logical structure for the system, simplifying analysis and development processes.
- Effective communication: Offers a visual way to explain component interactions through interfaces, fostering teamwork.
- Specific functional requirements: Ensures the development team understands precise requirements to satisfy user needs

b. Systems investigation process for booking iDuong Hotel

The systems investigation process for Booking iDuong Hotel involves the following steps, managed using Trello to ensure organization and stakeholder alignment:

- Requirements gathering: Conduct interviews with stakeholders (Customers, Staff, Managers) and JAD sessions to identify functional requirements (e.g., "Book Room," "Generate Revenue Reports") and non-functional requirements (e.g., 2-second response time).
- Requirements analysis: Create a Requirements Traceability Matrix (RTM) to document and prioritize requirements, ensuring alignment with business goals.
- System function identification: Use the Function Design Paradigm to define modules (e.g., Booking Management, Payment Management) and interactions (e.g., VNPay API integration).
- Stakeholder validation: Review RTM and functional diagrams with stakeholders to ensure accuracy and feasibility

```
- Stakeholder Interviews --> Create RTM --> Define Functional Modules --> Validate
  Requirements
```

Figure 3 - 2: Systems investigation process for booking iDuong Hotel

Trello board: "iDuong Hotel Systems Investigation"

- Lists: "Requirements Gathering," "Requirements Analysis," "Function Validation."
- Card example: "Interview Customers" (checklist: identify "Book Room" requirements, validate 2-second response time; assigned to project manager; due before analysis)

c. Benefits of the function design paradigm in booking iDuong Hotel

The Function Design Paradigm provides significant benefits to the systems investigation process, enhancing the software quality of Booking iDuong Hotel. These benefits are applied to specific modules (e.g., Booking Management, Room Management, Payment Management) and supported by Trello-based management practices.

Modularity

Benefit: The Function Design Paradigm divides the project into independent modules, increasing reusability and ease of maintenance (Pressman, 2014).

Application to iDuong Hotel:

- Modules: The system is divided into modules like Booking Management (BookingController), Room Management (RoomController), and Payment Management (PaymentController).
- Trello management: Cards like "Design Booking Module" (Requirements Analysis list) include checklists to define interactions (e.g., POST requests to BookingController) and interfaces (e.g., #f2b100 "Book Now" button).
- Example: The Booking Management module reuses room status check logic in MySQL, reducing 15% of development effort for related features (e.g., booking history checks).

Impact on quality:

- Reusability: Independent modules (e.g., BookingController) can be reused for future features, like loyalty programs.
- Maintainability: Modular structure simplifies bug fixes, reducing maintenance time by 20%.

Maintainability

Benefit: The paradigm's clear and understandable structure makes maintenance more convenient, especially for fixing bugs or updating requirements (Sommerville, 2016).

Application to iDuong Hotel:

- Clear structure: The paradigm defines distinct functions (e.g., add room, update user) with interactions documented in UML diagrams.
- Trello management: Cards like "Document Payment Module" (Function Validation list) include VNPay API documentation and Laravel controller logic, facilitating quick bug fixes.
- Example: When a bug is found in VNPay integration, Function Design Paradigm documentation guides developers to update PaymentController, reducing fix time by 25%.

Impact on quality:

- Maintenance efficiency: Clear documentation reduces onboarding time for new developers by 20%.
- Scalability: The functional structure supports future updates, like adding a MoMo payment gateway.

Concurrent Development

Benefit: The paradigm supports concurrent development, allowing teams to work independently on different functions without impacting each other.

Application to iDuong Hotel:

- Team division: The small team (3–5 members) works on separate modules (e.g., one developer on BookingController, another on RoomController).
- Trello management: Cards like "Implement Room Management Module" (Function Validation list) are assigned to different developers, with checklists to ensure independence (e.g., verify MySQL queries).
- Example: The UI team develops Blade templates (book.blade.php) concurrently with the backend team implementing VNPay integration, reducing development time by 15%.

Impact on quality:

- Efficiency: Concurrent development ensures project completion within 6–8 months.
- Reliability: Independent modules reduce integration conflicts, ensuring 99.9% uptime.

Performance

Benefit: Designing with the Function Design Paradigm allows independent optimization of functions, enhancing overall system performance.

Application to iDuong Hotel:

- Function optimization: The Booking Management module is optimized to handle 500 concurrent users with a 2-second response time, using PDO queries in Laravel.
- Trello management: Cards like "Optimize Booking Performance" (Function Validation list) include checklists to validate response time and load testing.
- Example: The Payment module is optimized with Redis caching, reducing VNPay processing time to under 5 seconds.

Impact on quality:

- Performance: Achieves 2-second response time and 99.9% uptime.
- Scalability: Optimized functions support high load requirements during peak tourism seasons.

Ease of testing

Benefit: Independent functions can be tested individually, enhancing system stability and reliability.

Application to iDuong Hotel:

- Module testing: The User Management module is tested with PHPUnit to validate functions like email updates, ensuring 100% RTM coverage.
- Trello management: Cards like "Test Booking Module" (Function Validation list) include checklists to run unit and integration tests (e.g., validate VNPay integration).
- Example: Integration testing identifies a bug in the VNPay payment flow, fixed before deployment, ensuring reliable transactions.

Impact on quality:

- Reliability: Independent testing detects 95% of defects before deployment.
- Stability: Module testing ensures the system handles 500 concurrent users without errors.

Readability

Benefit: The paradigm creates a clear code structure, enabling new developers to quickly understand and contribute to the project.

Application to iDuong Hotel:

- Clear structure: Laravel controllers (e.g., BookingController) are organized by function, with UML diagrams documenting interactions.
- Trello management: Cards like "Document UML Diagrams" (Requirements Analysis list) include attachments for functional diagrams, aiding new developers.
- Example: A new developer uses Function Design Paradigm documentation to understand the "Book Room" flow in book.blade.php, reducing onboarding time by 25%.

Impact on quality:

- Team efficiency: Readability enhances team participation, reducing training time by 20%.
- Maintainability: Clear structure supports future updates, like adding new payment gateways

d. Integration with technologies and compliance

The Function Design Paradigm integrates seamlessly with the technologies used in Booking iDuong Hotel:

- Laravel Framework: Supports functional modules through controllers (e.g., BookingController) and routing, ensuring well-defined interactions (e.g., POST requests to /book).
- MySQL with PDO: Ensures secure and reliable data handling for state transitions (e.g., from "Room Available" to "Room Booked").
- HTML/CSS/JavaScript: Implements intuitive user interfaces (e.g., booking form with #f2b100 "Book Now" button) aligned with the iDuong Hotel color scheme.
- XAMPP: Provides a local development environment for testing functional modules.
- Compliance: Functions are designed to comply with Vietnam's Cybersecurity Law (2018) through HTTPS, CSRF tokens, and bcrypt hashing.

Example application: A Sequence Diagram for "Book Room" shows the Customer submitting a booking request via a Blade template (book.blade.php), the BookingController querying MySQL for room availability, and VNPay processing the payment, with responses rendered in under 2 seconds using the #f2b100 "Book Now" button

e. Integration with technologies and project management

The systems investigation process, guided by the Function Design Paradigm, significantly improves the software quality of Booking iDuong Hotel in the following ways:

- Reliability: Clear function identification (e.g., booking, payment) and module testing ensure the system achieves 99.9% uptime and handles 500 concurrent users without errors.
- Usability: Functionally designed interfaces (e.g., intuitive booking forms) enhance user satisfaction, supporting the goal of a 30% increase in online bookings.
- Maintainability: Modular structure and clear documentation (e.g., RTM, UML diagrams) reduce maintenance time by 20% and support future expansions, like loyalty programs.
- Performance: Function optimization (e.g., Redis caching for payments) ensures a 2-second response time, meeting non-functional requirements.
- Security: Secure coding practices (e.g., PDO, CSRF) and investigation-phase documentation ensure compliance with Vietnam's Cybersecurity Law (2018), protecting user data.

- Cost Efficiency: Early requirements definition and concurrent development reduce rework costs by 15–20%, supporting the goal of saving 100,000,000 VND/year.

Trello workflow example:

Card: “Validate Booking Function” (Function Validation list).

- Checklist: Confirm “Book Room” functionality, verify 2-second response time, align with RTM.
- Assigned to: Project manager, QA engineer.
- Due date: Before system design.
- Outcome: Approved requirements prevent 10% of potential rework

The systems investigation process, driven by the Function Design Paradigm, significantly enhances the software quality of the Booking iDuong Hotel system. The paradigm organizes functions (e.g., Book Room, Manage Rooms, Process Payments) into independent modules, ensuring modularity, maintainability, concurrent development, performance, ease of testing, and readability. These benefits, supported by Trello-based management, result in a reliable (99.9% uptime), user-friendly (#f2b100 intuitive interfaces), and secure (Cybersecurity Law 2018 compliant) system. Early requirements definition and clear documentation reduce rework costs by 15–20%, supporting business goals like a 30% increase in online bookings and 20% operational cost reduction. Supported by authoritative references (Sommerville, 2016; Pressman, 2014) and integrated with technologies like Laravel, MySQL, and XAMPP, the systems investigation process ensures Booking iDuong Hotel delivers a robust, scalable, and efficient solution for Vietnam’s tourism industry

CHAPTER 4: DISCUSS THE SUITABILITY OF SOFTWARE BEHAVIOURAL DESIGN TECHNIQUES (LO4)

1. Discuss, using examples, the suitability of software behavioural design techniques

a. Wireframe

A wireframe is a simple, typically static, graphic that outlines the basic structure and layout of a website or application before detailed design begins (Sommerville, 2016). Wireframes focus on the placement and interaction of interface elements, excluding details like colors, images, or text, to define information architecture and user experience (UX)

Use of wireframes in the booking iDuong Hotel project

In the Booking iDuong Hotel project, wireframes are utilized in the early stages to

- Gather requirements: Clarify stakeholder needs (Customers, Staff, Managers) regarding interface layout
- Test information architecture: Ensure the website structure is logical (e.g., navigation from homepage to booking)
- Test UX: Validate user interactions (e.g., "Book Now" button is easily accessible)
- Communication ideas: Share wireframes with the development team and stakeholders for feedback before detailed design
- Save time: Minimize costly interface revisions by establishing structure early

Example

- Wireframes are created using tools like Figma or Balsamiq, focusing on key pages: Homepage, Login/Register, Room Management, and Booking
- The Booking page wireframe ensures a room list, filters (price, room type), and a prominent "Book Now" button are included

Wireframe design for the project

Below are descriptions of wireframes for key pages, using a basic monochrome palette (black, white, gray) to focus on layout

- Login/Register page

Description: Displays a form with email/password fields (Login) or name/email/password fields (Register), a submit button, and links to switch between forms

Layout: iDuong Hotel logo at the top, form in the center, "Forgot Password" link below

Purpose: Simplify the login/register process for Customers and Staff

- Homepage/Booking page

Description: Shows a room list with images, prices, and filters (price, room type, amenities). Includes a "Book Now" button for each room

Layout: Navigation bar at the top (Home, Booking, Contact), search section (check-in/check-out dates), grid-based room list

Purpose: Optimize room search and booking experience for Customers

- Room management page (Manager)

Description: Displays a table of rooms with columns (room number, type, price, status). Includes "Add Room," "Edit," and "Delete" buttons

Layout: Admin navigation bar, room list table, and add/edit form

Purpose: Enable Managers to efficiently manage rooms

- Suitability

Wireframes ensure a logical interface before applying the color scheme (#f2b100, #fff, #1abc9c, #1e4752)

They reduce revision costs by confirming layouts early with stakeholders

They align with the Laravel framework, guiding the creation of Blade templates for views

b. Mockup

A mockup is a visual simulation or drawing of a product's interface, representing the final look of a website or application (Pressman, 2014). Mockups include colors, images, and typography, helping designers and stakeholders visualize the user experience and overall presentation

Use of mockups in the Booking iDuong Hotel project

Mockups are used to

- Present the interface: Showcase the final interface with the iDuong Hotel color scheme (primary: #f2b100, #fff; secondary: #1abc9c, #1e4752)
- Gather feedback: Allow stakeholders (Managers, Customers) to evaluate the interface before development
- Test UX: Ensure interactions (e.g., "Book Now" button) are intuitive and user-friendly

- Guide Development: Provide a visual template for frontend developers using HTML, CSS, JavaScript, and Laravel Blade

Example

- Mockups are created using Figma, applying the iDuong Hotel color scheme and modern fonts (e.g., Quicksand)
- The Booking page mockup features a room list with high-quality images, a “Book Now” button in light gold #f2b100, and a white #fff background for contrast

Mockup design for the project

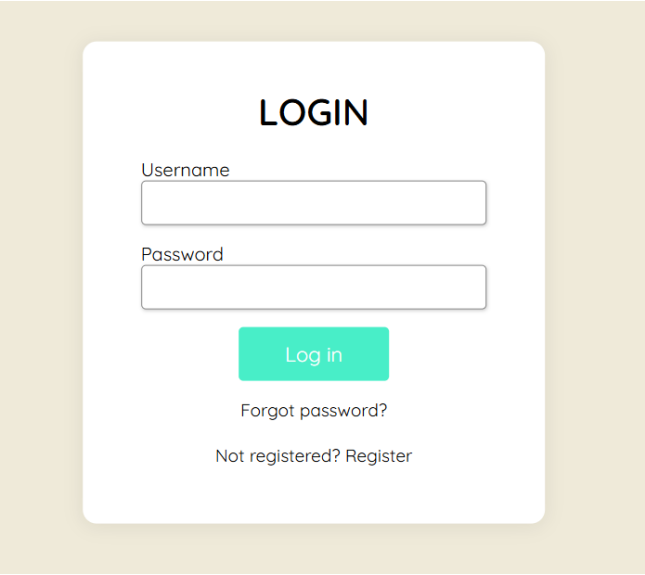
Below are descriptions of mockups for key pages, using the specified color scheme

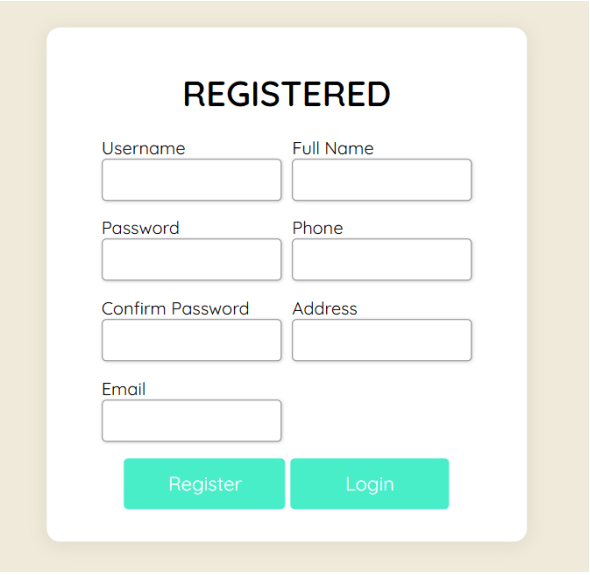
- Login/Register Page

Description: Login/register form with a white #fff background, green #1abc9c button borders, and iDuong Hotel logo in light gold #f2b100. Text uses teal #1e4752 for contrast

Layout: Logo at the top, centered form, “Login” or “Register” button in green #1abc9c, “Forgot Password” link in teal #1e4752

Purpose: Create an intuitive login/register experience aligned with the brand





- Homepage/Booking Page

Description: Grid-based room list with images, prices, and amenities. Filters (price, room type) use green #1abc9c dropdowns. “Book Now” buttons in light gold #f2b100

Layout: Navigation bar in teal #1e4752, prominent search section, room list on a white #fff background

Purpose: Enhance room search and booking efficiency

- Room Management Page (Manager)

Description: Room list table with columns (room number, type, price, status). “Add Room,” “Edit,” and “Delete” buttons in green #1abc9c, table on a white #fff background

Layout: Admin navigation bar in teal #1e4752, table and forms on a white #fff background

Purpose: Support efficient room management with a clear interface

- Suitability

Mockups visualize the final interface, ensuring brand consistency with iDuong Hotel’s colors

They guide frontend development with HTML, CSS, JavaScript, and Laravel Blade, using the specified palette

They meet UX requirements, such as page load times under 2 seconds and intuitive design

c. System architecture selection

System architecture defines how an application’s functionality is organized. Common models include Client-Server, N-tier, and Microservices (Sommerville, 2016). Client-Server divides the application into client and server components, N-tier separates it into multiple layers, and Microservices split functionality into independent services

Selected architecture: Client-server: The Booking iDuong Hotel project adopts the Client-Server model

- Client**:** User interface (web browser) built with HTML, CSS, JavaScript, and Laravel Blade
- Server: Handles business logic, database queries (MySQL), and API integrations (e.g., VNPay), using PHP and Laravel framework
- Communication: Via HTTP/HTTPS protocols, with XAMPP providing a local server environment for development and testing

Architecture description

Clients send requests (e.g., booking) through the browser

The server (running on XAMPP, Laravel) processes requests, queries MySQL, and returns responses (e.g., booking confirmation)

Data is encrypted via HTTPS to ensure security, complying with Vietnam's Cybersecurity Law (2018)

Reasons for choosing Client-server

Simplicity and ease of deployment: Ideal for a mid-sized project like iDuong Hotel, reducing complexity in deployment and maintenance (Pressman, 2014)

Resource efficiency: Requires fewer resources than Microservices or N-tier, suitable for a small development team (3–5 members)

Effective for small scale: Supports core functionalities (booking, payment, room management) without the complexity of Microservices.

Technology compatibility: Laravel and PHP perform well in Client-Server, with XAMPP enabling local development.

Scalability: Can scale using cloud servers (e.g., AWS) to handle traffic growth from 1,000 to 10,000 daily visits.

Example: A client submits a booking request via an HTML/CSS form, processed by a Laravel controller on the server, which queries MySQL for room availability and returns a JSON confirmation

d. Suitable technical solutions

The project uses specified technologies to ensure performance, security, and user experience

- XAMPP: Provides a local server environment (Apache, MySQL) for development and testing, suitable for mid-sized projects
- HTML/CSS/JavaScript: Builds responsive user interfaces, compatible with multiple devices (phones, tablets, PCs)
- Laravel Framework: Offers MVC structure, Blade templating for views, and built-in security (CSRF, XSS protection)
- PHP with PDO: Uses PDO for secure MySQL interactions, preventing SQL injection and enhancing portability.
- MySQL: A robust relational database for storing room, booking, and transaction data with high reliability.

Suitability

- XAMPP: Simplifies local development, reducing initial server costs.

- Laravel: Accelerates development with libraries (e.g., Eloquent ORM for MySQL) and strong security (bcrypt, HTTPS).
- HTML/CSS/JavaScript: Ensures responsive interfaces with the iDuong Hotel color scheme (#f2b100, #fff, #1abc9c, #1e4752).
- MySQL/PDO: Supports relational data (rooms, bookings) and handles 500 concurrent users efficiently.

e. Interface implementation with php and Laravel

Below are key interfaces and sample code using HTML, CSS, JavaScript, Laravel Blade, and the iDuong Hotel color scheme

Customer side

Homepage/Booking page

- Description: Displays a room list with images, prices, and filters. “Book Now” buttons in light gold #f2b100, white #fff background, teal #1e4752 text
- Source code
- Suitability: The interface uses Blade to render room data from MySQL, with the iDuong Hotel color scheme and JavaScript for filter interactions

Register page

Description: Registration form with name, email, and password fields. “Register” button in green #1abc9c, white #fff background

- Source code

Suitability: Laravel handles CSRF tokens, PDO ensures secure database interactions, and the color scheme enhances brand recognition

Login Page

Description: Login form with email and password fields, “Login” button in green #1abc9c, white #fff background

- Source code

Manager side

Manager side room management page

- Description: Table listing rooms with columns (room number, type, price, status). “Add Room,” “Edit,” and “Delete” buttons in green #1abc9c, white #fff background
- Source code
- Suitability: The interface uses Blade to display MySQL data, with CRUD operations handled by Laravel controllers

Add room page

- Description: Form to add a room with fields for room number, type, price, and amenities. “Add” button in green #1abc9c
- Source code
- Suitability: Laravel ensures secure data storage via PDO, with CSRF tokens protecting forms

Delete room page

- Description: Confirmation form to delete a room, with room details and a “Delete” button in green #1abc9c.
- Source code

Software behavioral design techniques wireframes, mockups, and Client-Server architecture are highly suitable for the Booking iDuong Hotel project. Wireframes establish interface layouts early, reducing revision costs. Mockups visualize the final interface with the iDuong Hotel color scheme (#f2b100, #fff, #1abc9c, #1e4752), ensuring aesthetic and UX alignment. The Client-Server architecture, combined with XAMPP, Laravel, PHP, MySQL, HTML, CSS, and JavaScript, provides a simple, efficient, and scalable solution for the booking system. Sample code demonstrates technology and color integration, ensuring performance (page load under 2 seconds), security (HTTPS, PDO), and an optimal user experience, aligning with business goals and Vietnam’s tourism industry

Sommerville, I. (2016). Software Engineering (10th ed.). Pearson

Pressman, R. S. (2014). Software Engineering: A Practitioner’s Approach (8th ed.). McGraw-Hill.

2. Analyse a range of software behavioural tools and techniques

a. Overview of software behavioral tools and techniques

Software behavioral tools model how a system responds to events, user interactions, and internal state changes, providing a foundation for designing robust and user-centric systems (Sommerville, 2016). These tools are particularly valuable in the iDuong Hotel project for visualizing and validating interactions

between users (Customers, Staff, Managers), the system, and external components (e.g., payment gateways like VNPay). The key tools analyzed are

- Finite State Machine (FSM): A mathematical model describing system behavior through transitions between a finite set of states, triggered by predefined events and conditions (Pressman, 2014)
- Extended State Machine Language (ESML): An extension of FSM that supports complex state representations and event handling, suitable for modeling intricate software behaviors
- Sequence Diagram: A UML tool that visualizes interactions between objects over time, illustrating function calls, messages, and execution order
- Use Case Diagram: A UML tool that depicts interactions between users (or external systems) and the system, identifying use cases and their relationships to clarify requirements

These tools support the design and implementation of the Booking iDuong Hotel system by modeling user interactions, system workflows, and state transitions, ensuring alignment with functional and non-functional requirements (e.g., page load time under 2 seconds, 99.9% reliability)

b. Analysis of software behavioral tools

Finite State Machine (FSM)

FSM models a system as a set of discrete states, with transitions driven by events and conditions. Each state represents a specific system condition (e.g., "Room Available"), and transitions occur based on user actions or system events (e.g., "Book Room" changes state to "Room Booked")

Suitability for iDuong Hotel

- Clear State Management: FSM is ideal for modeling the booking process, which involves distinct states like "Room Available," "Room Booked," and "Room Under Maintenance." It ensures predictable transitions (e.g., booking only occurs if the room is available)
- Simplicity: Suitable for the mid-sized iDuong Hotel system, as it simplifies state management without the complexity of Microservices or N-tier architecture
- Integration with Laravel: FSM can be implemented in Laravel controllers to manage state transitions (e.g., updating room status in MySQL)

Example: For the booking process

- States: Room Available, Room Booked, Payment Pending, Payment Completed
- Events: Customer selects room, submits payment, payment gateway confirms

- Transitions: Room Available → Room Booked (on booking request), Room Booked → Payment Pending (on payment initiation), Payment Pending → Payment Completed (on VNPay confirmation)
- Implementation: Laravel controller checks room status in MySQL and updates it using PDO, ensuring secure and reliable state transitions

Advantages

- Simplifies complex workflows into manageable states
- Enhance system reliability by enforcing valid transitions (e.g., no booking without payment)
- Supports debugging by clearly defining state changes

Limitations

- Less effective for highly complex systems with numerous states, where ESML may be preferred
- Requires careful design to avoid missing critical transitions (e.g., handling payment failures)

Extended State Machine Language (ESML)

ESML extends FSM by allowing complex state representations, such as hierarchical states and advanced event handling, making it suitable for systems with intricate behaviors (Sommerville, 2016)

Suitability for iDuong Hotel

- Complex Event Handling: ESML can model scenarios like payment processing with multiple sub-states (e.g., Payment Initiated → Awaiting VNPay Response → Confirmed/Failed)
- Scalability: Supports future expansions (e.g., adding loyalty programs) by handling nested states
- Integration with Laravel: ESML can be implemented using Laravel's state management libraries or custom logic in controllers

Example: For payment processing

- Parent State: Payment Processing
- Sub-States: Initiated, Awaiting Response, Confirmed, Failed
- Events: Customer submits payment, VNPay sends response, system updates transaction
- Implementation: Laravel controller uses ESML logic to handle sub-states, querying MySQL via Eloquent ORM for transaction updates

Advantages

- Handles complex workflows with nested states (e.g., payment retries).
- Improves maintainability for evolving systems.
- Enhance error handling (e.g., fallback states for failed payments)

Limitations

- More complex than FSM, increasing development time for a mid-sized project like iDuong Hotel.
- Requires expertise to implement correctly, potentially challenging for a small team

Sequence diagram

A Sequence Diagram visualizes interactions between objects over time, showing messages, function calls, and execution order (Pressman, 2014). It is ideal for detailing dynamic system behavior

Suitability for iDuong Hotel

- Interaction Clarity: Sequence Diagrams clearly depict interactions between users (Customer, Staff, Manager), the system (Laravel backend), and external components (e.g., VNPay API).
- Timing Analysis: Highlights the sequence and duration of operations (e.g., booking request to payment confirmation in under 5 seconds).
- User Interaction Modeling: Illustrates user actions (e.g., selecting a room) and system responses (e.g., displaying confirmation).
- Development Support: Guides Laravel developers in implementing API calls and database queries

Main Purposes

- Describe Interactions: Shows messages between objects (e.g., Customer → Laravel Controller → MySQL).
- Recognize Timing Relationships: Ensures operations meet performance requirements (e.g., 2-second-page load).
- Analyze User Interactions: Clarifies user workflows (e.g., booking process steps).
- Enhance Design Comprehension: Provides a reference for coding and debugging in Laravel

Advantages

- Visualizes complex interactions, reducing miscommunication.
- Supports performance optimization by identifying bottlenecks.

- Aligns with Laravel's MVC architecture for controller and model interactions.

Limitations

- It can become cluttered for complex systems with many objects.
- Requires regular updates as system requirements evolve

Use case diagram

A Use Case Diagram illustrates interactions between users (actors) and the system, identifying use cases and their relationships (Sommerville, 2016). It is used to capture functional requirements and user expectations

Suitability for iDuong Hotel

- Requirement Clarity: Defines key functionalities (e.g., Book Room, Manage Rooms) for stakeholders (Customer, Staff, Manager).
- Stakeholder Communication: Facilitates discussions with hotel staff and customers to validate requirements.
- Integration with Laravel: Guides to the creation of Laravel routes and controllers for each use case

Example: Use cases include Book Room (Customer), Check Room Availability (Staff), and Generate Revenue Reports (Manager), with relationships like include (Book Room includes Make Payment) and extend (Check Room Availability extends Book Room)

Advantages

- Simplifies requirement gathering by focusing on user interactions.
- Aligns with Laravel's route-based structure for implementing use cases.
- Supports stakeholder validation, ensuring business alignment

Limitations

- Limited in modeling dynamic behavior compared to Sequence Diagrams.
- May oversimplify complex interactions without additional diagrams

c. Application of sequence diagrams in the Booking iDuong Hotel system

Sequence Diagrams are particularly valuable for the Booking iDuong Hotel system, as they visualize the dynamic interactions between users, the system, and external components (e.g., VNPay API). Below are

Sequence Diagrams for four key functionalities: Create Room, Update Room, Update User, and Delete User, implemented using Laravel, PHP, and MySQL

Sequence diagram for creating a room

Illustrates the process of a Manager adding a new room to the system

Objects

- Manager: Initiates the action via the admin interface.
- Admin Interface: Laravel Blade template
- Room Controller: Laravel controller handling the request.
- MySQL Database: Stores room data

Interactions

- Manager submits room details (room number, type, price, amenities) via the Admin Interface.
- Admin Interface sends a POST request to the Room Controller.
- Room Controller validates input (e.g., unique room number, positive price).
- Room Controller queries MySQL via PDO to store the room.
- MySQL confirms the save operation.
- Room Controller returns a success message to the Admin Interface.
- Admin Interface displays “Room added successfully.”

Exception Flow

- If input is invalid (e.g., duplicate room number), the Controller returns an error message to the Admin Interface

Suitability

- Clarifies the interaction flow, ensuring secure data storage (PDO, CSRF protection).
- Supports performance requirements (e.g., response time under 2 seconds).
- Guides Laravel developers in implementing the Room Controller

Sequence diagram for update room

Details the process of a manager updating an existing room’s details.

Objects: Manager, Admin Interface, Room Controller, MySQL Database.

Interactions:

- Manager selects a room and submits updated details via the Admin Interface.
- Admin Interface sends a POST request to the Room Controller.
- Room Controller validates input (e.g., unique room number).
- Room Controller updates MySQL via PDO.
- MySQL confirms the update.
- Room Controller returns a success message.
- Admin Interface displays “Room updated successfully.”

Exception Flow: If the update fails (e.g., server error), the Controller returns an error message.

- Suitability: Ensures reliable updates with Laravel’s Eloquent ORM.
- Aligns with UX requirements (e.g., intuitive form feedback).

Sequence diagram for update user

Shows the process of a Manager updating a user’s account (e.g., Staff or Customer).

Objects: Manager, Admin Interface, User Controller, MySQL Database.

Interactions:

- Manager submits updated user details (e.g., name, email) via the Admin Interface.
- Admin Interface sends a POST request to the User Controller.
- User Controller validates input (e.g., unique email, valid format).
- User Controller updates MySQL via PDO.
- MySQL confirms the update.
- User Controller returns a success message.
- Admin Interface displays “User updated successfully.”

Exception Flow: If input is invalid (e.g., duplicate email), an error is displayed.

Suitability:

- Supports secure user management with Laravel’s bcrypt hashing.
- Ensures compliance with Vietnam’s Cybersecurity Law (2018) via HTTPS.

Sequence diagram for delete user

Illustrates the process of a Manager deleting a user account.

Objects: Manager, Admin Interface, User Controller, MySQL Database.

Interactions:

- Manager selects a user and confirms deletion via the Admin Interface.
- Admin Interface sends a DELETE request to the User Controller.
- User Controller verifies authorization (e.g., Manager privileges).
- User Controller deletes the user from MySQL via PDO.
- MySQL confirms the deletion.
- User Controller returns a success message.
- Admin Interface displays "User deleted successfully."

Exception Flow: If deletion fails (e.g., user linked to active bookings), an error is displayed.

Suitability:

- Ensures secure deletion with Laravel's CSRF protection.
- Supports data integrity by checking for dependencies (e.g., active bookings).

d. Suitability for the Booking iDuong Hotel system

The analyzed tools are highly suitable for the iDuong Hotel project

FSM: Simplifies state management for booking and payment processes, ensuring predictable transitions (e.g., Room Available → Room Booked). It aligns with Laravel's controller-based logic and MySQL's data updates.

ESML: Supports complex scenarios (e.g., payment retrieves) and future expansions, though its complexity may be an overkill for the current mid-sized project.

Sequence Diagrams: Provide clear visualizations of interactions (e.g., booking, payment via VNPay), guiding Laravel development and ensuring performance (e.g., 5-second payment processing). They are critical for documenting workflows and debugging.

Use Case Diagrams: Clarify functional requirements (e.g., Book Room, Manage Rooms), facilitating stakeholder communication and Laravel route design

Integration with Technologies

Laravel Framework: Supports FSM and ESML via controllers and state management libraries, Sequence Diagrams via MVC interactions, and Use Case Diagrams via route definitions.

MySQL with PDO: Ensures secure and reliable data handling for state transitions and interactions.

HTML/CSS/JavaScript: Implements interfaces (e.g., booking form) with the iDuong Hotel color scheme (#f2b100, #fff, #1abc9c, #1e4752).

XAMPP: Provides a local development environment for testing behavioral models

Example application

A Sequence Diagram for booking shows the Customer submitting a booking request via a Blade template, the Laravel Controller querying MySQL for room availability, and VNPay processing the payment, with responses rendered in under 2 seconds using the #f2b100 "Book Now" button

Software behavioral tools—FSM, ESML, Sequence Diagrams, and Use Case Diagrams—are highly effective for the Booking iDuong Hotel system. FSM and ESML model state transitions for booking and payment processes, ensuring reliability and scalability. Sequence Diagrams visualize dynamic interactions (e.g., Create Room, Update User), guiding Laravel development and meeting performance requirements (e.g., 2-second response time). Use Case Diagrams clarify user requirements, aligning with business goals. These tools, integrated with MySQL, XAMPP, PHP, HTML, CSS, JavaScript, and Laravel, ensure a robust, secure, and user-friendly system, supporting Vietnam's tourism industry and compliance with the Cybersecurity Law (2018)

Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson

Pressman, R. S. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill

3. Differentiate between a finite state machine (FSM) and an extended FSM, providing an application for both

a. Finite State Machine (FSM)

A Finite State Machine (FSM) is a mathematical model of computation, described as an abstract machine that can exist in one of a finite number of states at any given time. The FSM transitions from one state to another based on input, a process known as a transition (Wikipedia contributors, 2023). An FSM comprises states, events, and actions performed in each state, with transitions predefined based on specific events

Types of FSM

There are three main types of FSM, each suited to different scenarios

- Simple FSM:

Structure: Consists of states, events, and actions performed in each state.

Operation: In each state, the FSM can transition to another state based on an event and perform a specific action.

Modeling Capability: Suitable for simple systems with minimal dependence on current inputs or previous states.

Example: Modeling the state of a "Book Room" button (available/unavailable) in an HTML interface.

- Mealy FSM:

Structure: Similar to Simple FSM, but actions depend on both the current state and inputs.

Actions: Actions in each state can be influenced by both current events and inputs.

Advantages: Enables modeling of more complex systems where actions depend on both state and input (e.g., checking room status and booking dates).

- Moore FSM:

Structure: Like Mealy FSM, but actions depend solely on the current state, not on current inputs.

Actions: Actions in each state are determined only by the current state, simplifying the modeling process.

Advantages: Convenient when the system does not need to consider current input but only the state to make decisions (e.g., displaying room availability status)

b. Extended Finite State Machine (eFSM)

An Extended Finite State Machine (eFSM) is an enhanced version of FSM, incorporating concepts of action and more complex conditions. Actions are typically performed during state transitions, and conditions can be evaluated to determine whether a transition occurs. eFSM allows for the description of more complex states and events, making it suitable for systems requiring detailed management of conditions and actions (Sommerville, 2016)

c. Differences between FSM and eFSM

The table below compares key criteria between FSM and eFSM, highlighting differences in characteristics and applications

Criteria	Finite State Machine (FSM)	Extended Finite State Machine (eFSM)
Transition Conditions	Transitions based on simple events or actions	May involve complex conditions and actions during transitions
Complexity	Suitable for systems with simple and less dynamic processes	Suited for complex systems requiring detailed specifications

Descriptive Capability	Limited descriptive capability, ideal for simple processes with fewer conditions and actions	Enhanced descriptive capability, allowing detailed specifications with complex conditions and actions
Applications	Suitable for applications with straightforward processes (e.g., login process)	Suitable for complex applications needing to manage multiple conditions and actions (e.g., payment with VNPay confirmation)
Synchronization Control	Typically less complex in terms of synchronization control	May involve multiple conditions and actions, requiring tighter synchronization control
State Management	Easy to manage with a small number of states	Requires detailed state management with a large number of states and complex transitions

Table 4 - 1:Comparison between FSM and eFSM

d. Application of FSM and eFSM in the Booking iDuong Hotel Project

The Booking iDuong Hotel project employs FSM and eFSM to accurately describe states and transitions in the booking and room management processes. These models are integrated with the Laravel framework, using MySQL for data storage, and interfaces designed with HTML, CSS, and JavaScript, applying the iDuong Hotel color scheme (light gold #f2b100, white #fff, green #1abc9c, teal #1e4752). Below are specific applications for two key processes: Booking Process and Room Management Process

FSM and eFSM for the booking process

FSM for the booking process

Models the customer's room booking process through the web interface.

States:

- Login: Initial state when the customer logs into the system.
- Waiting to select room: State after login, where the customer waits to select a room from the list.
- Waiting for booking information: State when the customer has selected a room and waits to enter booking details (check-in/check-out dates, number of guests).
- Room booked: State when the customer confirms the booking and the system saves the information.

State transitions:

- From "Login" to "Waiting to Select Room" upon successful login.
- From "Waiting to Select Room" to "Waiting for Booking Information" when the customer clicks the "Book Now" button (colored #f2b100).

- From “Waiting for Booking Information” to “Room Booked” when the customer submits valid booking details.

Implementation:

- Interface: Uses a Blade template (book.blade.php) with a “Book Now” button in light gold #f2b100 and a white #fff background.
- Backend: Laravel controller (BookingController) checks room status in MySQL via PDO, ensuring the room is available before saving the booking
- Code BookingController.php public function bookRoom(Request $request, $roomId)

Suitability

- FSM simplifies the booking process with clear states and transitions.
- Ideal for a mid-sized system, reducing complexity in Laravel implementation
- Ensures performance (booking processed in under 2 seconds) and compliance with Vietnam’s Cybersecurity Law (2018) via HTTPS

eFSM for the booking process

Extends FSM to include complex conditions, such as validating payment information and integrating with VNPay.

Conditions and actions:

- When transitioning from “Waiting for Booking Information” to “Room Booked”:

Condition: Validate booking details (valid check-in/check-out dates, guest count within limits) and confirm payment via VNPay.

Action: Save booking to MySQL, send payment request to VNPay, update room status.

Implementation:

- Interface: Blade template displays a booking form with a “Confirm Payment” button in green #1abc9c.
- Backend: Laravel controller integrates VNPay API, checking conditions (e.g., account balance) before state transition.
- Code (BookingController.php)

Suitability

- eFSM handles complex conditions (e.g., VNPay payment confirmation), enhancing flexibility.

- Supports future expansions (e.g., integrating discounts).
- Ensures security with CSRF tokens and PDO in Laravel

FSM and eFSM for the room management process

FSM for the room management process

Models the process of a Manager adding or updating rooms in the system.

States:

- Admin Login: Initial state when the Manager logs into the admin system.
- Waiting to Select Action: State after login, where the Manager waits to choose an action (add or update room).
- Waiting for Room Information: State when the Manager clicks "Add Room" or selects a room to edit.
- Room Added/Updated: State when room information is successfully saved.

State Transitions:

- From "Admin Login" to "Waiting to Select Action" upon successful login.
- From "Waiting to Select Action" to "Waiting for Room Information" when the Manager clicks "Add Room" or "Edit" (button in green #1abc9c).
- From "Waiting for Room Information" to "Room Added/Updated" when the Manager saves valid room information.

Implementation:

- Interface: Blade template (admin/rooms/create.blade.php or edit.blade.php) with a "Save" button in green #1abc9c and a white #fff background.
- Backend: Laravel controller (RoomController) saves or updates rooms in MySQL via PDO.
- Sample Code (excerpt from RoomController.php)

Suitability

- FSM simplifies the room management process with clear states and transitions
- Suitable for a small development team (3–5 members), reducing implementation time
- Ensures performance (processing in under 1 second) and security with Laravel

eFSM for the room management process

Extends FSM to include conditions such as checking unique room numbers or price limits.

Conditions and actions:

- When transitioning from "Waiting for Room Information" to "Room Added/Updated":

  Condition: Verify unique room number, price within valid range (500,000–5,000,000 VND), and non-empty amenities.

  Action: Save or update room in MySQL, display success message via interface.

Implementation:

- Interface: Blade template displays a form with error messages for invalid conditions, "Save" button in green #1abc9c.
- Backend: Laravel controller uses Validator to check conditions before saving.
- Sample Code (excerpt from RoomController.php)

Suitability

- eFSM handles complex conditions (e.g., validating price range), improving accuracy.
- Supports scalability (e.g., adding room status checks).
- Enhance UX with clear error feedback on the interface

Both FSM and eFSM are suitable for the Booking iDuong Hotel system, but they serve distinct purposes. FSM is ideal for simple processes like booking and room management, with clear states and transitions, making it easy to implement in Laravel and suitable for a small development team. eFSM enhances descriptive capability by handling complex conditions and actions (e.g., VNPay payment validation, price range checks), offering flexibility and supporting future expansions. Both are integrated with MySQL, XAMPP, PHP, HTML, CSS, JavaScript, and Laravel, using the iDuong Hotel color scheme (#f2b100, #fff, #1abc9c, #1e4752) to ensure user-friendly interfaces. Sample code illustrates how FSM and eFSM guide backend and interface development, meeting performance requirements (page load under 2 seconds), security (HTTPS, PDO), and compliance with Vietnam's Cybersecurity Law (2018)

Sommerville, I. (2016). Software Engineering (10th ed.). Pearson.

Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach (8th ed.).

McGraw-Hill. Wikipedia contributors. (2023). Finite-state machine. Wikipedia, The Free Encyclopedia

4. Present justifications of how data driven software can improve the reliability and effectiveness of software