GUIDELINES FOR REPORT PRESENTATION
ESSAY, ASSIGNMENT, PROJECT, CAPSTONE PROJECT

(Applicable for student groups supervised by Mr. Mai Van Manh)

Last updated: 25/08/2024

Depending on the nature of each topic and the requirements of the supervisor, the report writing style may vary. However, the general objective of all reports is to provide a comprehensive overview of the group's research and project implementation. The report should include: an introduction to the topic; reasons for selecting it and its practical significance; background knowledge and references related to the work; concepts and theories studied; as well as system design, diagrams, tables, and illustrations of the process. The report should also present the group's proposals, main contributions, or outstanding features of the website/software, the achieved results, and comparisons with previous research/products. Finally, the report should conclude with a summary, references, and appendices (if any).

Suggested Table of Contents for Software Development Projects

- Cover pages and appendices as required by regulations
- Abstract
- Chapter 1 – Introduction
- Chapter 2 – Literature Review
- Chapter 3 – Requirements Analysis
- Chapter 4 – System Design
- Chapter 5 – Development and Testing
- Chapter 6 – Deployment and Evaluation
- Chapter 7 – Conclusion
- References and Appendices as require

For smaller-scale reports (essays), certain chapters may be combined. For example, Chapters 3 and 4 (and even Chapter 5) can be merged depending on report length. Conversely, Chapter 5 can be split into smaller sections for clarity and detail.

Below is a more detailed explanation of the report's table of contents and the elements that must be addressed in the report:

1. Mandatory information pages required by the Faculty/University template.
2. The table of contents, and—where applicable—lists of figures, tables, and equations.

3. Abstract page (Abstract): Provide a concise description of the product, the rationale for its development, and the problem it seeks to solve. For example, if the product is a mobile application, the abstract should describe the app's principal functionality (e.g., allowing users to upload and share images) together with any notable technologies (e.g., using AI to suggest searches or classify images). Summarize the project's results and contributions. This abstract should not exceed one A4 page.

CHAPTER 1. INTRODUCTION

1.1 Background and Rationale: Introduce the problem that the group intends to solve.
Example: "Nowadays, the demand for online language learning is rapidly increasing; however, existing applications are not yet optimized for personalized learning content."

1.2 Project Objectives: Clearly present the specific goals that the product aims to achieve.
Example: "Develop a mobile application that enables learners to practice English listening and speaking skills with real-time feedback from an artificial intelligence (AI) system."

1.3 Scope of the Project: Define the boundaries and limitations of the product.
Example: "This application supports only Android users and does not yet provide iOS support."

1.4 Research Methodology: Describe the methods the group employed for research and product development. This may include literature review, user surveys, and technical approaches such as Agile or Waterfall development.

1.5 Practical Significance: Describe the benefits and values that the group's product can bring to society or to specific user groups. This may involve addressing existing problems or improving particular processes.
Example: "This application holds significant practical value in helping English learners improve their listening and speaking skills through AI-based speech recognition technology. Additionally, it assists teachers in monitoring learners' progress more easily and allows them to adjust teaching methods based on collected data."

1.6 Structure of the Report: Provide an overview of the report's organization.

Example: “Chapter 2 analyzes the system requirements, Chapter 3 presents the system design, and Chapter 4 describes the implementation process, …”

CHAPTER 2. LITERATURE REVIEW

2.1 Related Works

The objective of this section is to evaluate existing products or research available on the market, analyzing their features, advantages, and limitations in comparison with the product your group is developing. Each related work should be presented clearly with the following key elements: product name, developer, main features, technologies used, strengths, and weaknesses. Conclude with comments on the limitations that your group’s product aims to address or improve.

2.2 Relevant Technologies

This section describes in detail the technologies used in your group’s product and explains why these technologies were chosen over other alternatives.

- **General introduction to each technology**: Briefly describe what the technology is and its role within the system. For core technologies in your project, provide a more in-depth explanation of their architecture and underlying principles. Where appropriate, use diagrams or illustrations to clarify the concept and enhance readability (e.g., a diagram comparing Docker’s operating principle with that of traditional virtual machines through a hypervisor).
- **Justification for technology selection**: Explain why your group selected this technology, highlighting its advantages over other possible solutions.
- **Practical examples**: Provide concrete examples of how the technology is applied within your group’s product.

2.3 Theoretical Principles and Models

Present the theoretical principles and software development models that the group applied during the design and implementation of the product. This section should demonstrate the group’s understanding of foundational concepts and how they were applied in practice.

- **Explanation of principles or models**: Introduce the theories behind the principles or models adopted, such as microservices, asynchronous communication, message brokers, decoupled architectures, single-page applications (SPA), and progressive web apps (PWA).
- **Application in the product**: Illustrate specifically how these models or principles were integrated into your group's product, with real examples.
- **Illustrations**: For such models and principles, clear and easy-to-understand diagrams should be included. Consider creating your own diagrams using drawing tools to ensure clarity and originality.

CHAPTER 3. REQUIREMENTS ANALYSIS (This chapter may be merged with Chapter 4 if the combined content is not too lengthy).

This chapter constitutes the first critical step in the software development process, where groups determine what the system must accomplish based on user or organizational needs. It not only helps readers understand the objectives of the system but also serves as the foundation for subsequent design and implementation.
The content of this chapter should clarify fundamental questions such as:

- Who are the target users of the system?
- What tasks must the system perform?

3.1 User Requirements

Analyze the requirements of end-users. These may include both direct and indirect users of the system, such as administrators, customers, or staff members. List the main user groups of the system and describe their roles. In some cases, a stakeholder diagram may be used to illustrate the related groups. Identify the specific requirements of each user group.
Example: Customers need to register accounts, browse products, add items to a shopping cart, and complete payments (with details provided, not just short bullet points).

3.2 Functional Requirements

Detail the specific functionalities the system will provide.

Example: User registration and login, listening and pronunciation practice with instant feedback, learning progress tracking.

3.3 Non-Functional Requirements

Include aspects such as performance, security, and scalability.

Example: “The application must support at least 1,000 concurrent users and maintain a response time under one second.”

3.4 Use Case Diagrams and Other UML Diagrams

Each diagram included in the report must be carefully designed to ensure clarity and readability, with text that is sharp and large enough to be legible when printed. Do not include unnecessary details in diagrams or screenshots. Every diagram must be accompanied by a detailed textual explanation—avoid simply inserting diagrams without any description.

3.5 Additional Content

Groups may supplement this chapter with other relevant materials related to their requirements analysis process.

CHAPTER 4. SYSTEM DESIGN

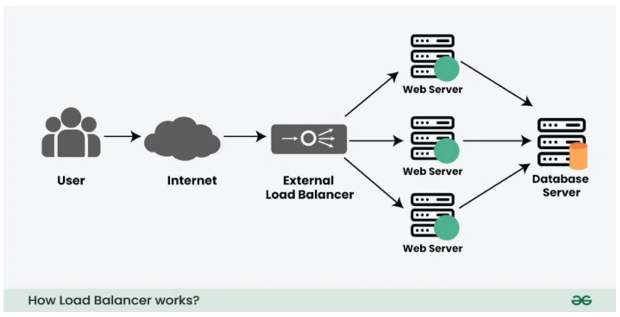
The System Design chapter plays a crucial role in defining the components and architecture of the system after requirements have been clearly analyzed. It translates the requirements into technical models, describing how the system will function, how its components interact with each other, and how data will be managed.

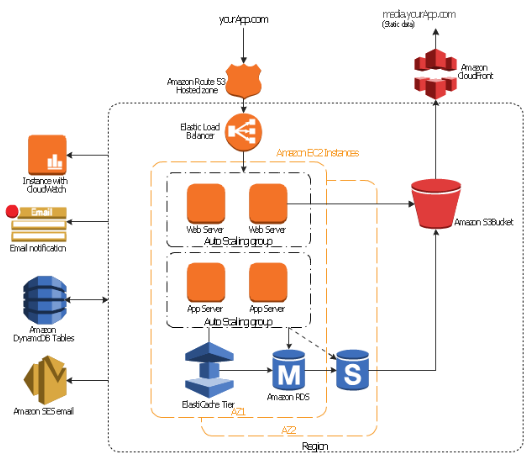
4.1 System Architecture

This section describes the overall architecture of the system, including how its major components are organized and how they communicate.

- Overall Architecture: Present the main components of the system such as the Frontend, Backend, Database, APIs, and any integrated third-party services.
- Client-Server or Microservices Architecture: Choose the most suitable architectural style and explain the rationale behind this choice.

- Architecture Diagram: Provide a diagram illustrating the system architecture to help readers clearly understand the interactions between components.





In this chapter, it is mandatory to include a system architecture diagram similar to the examples mentioned above. The diagram must list the system's components and show how they interact, enabling readers to easily visualize the overall system structure. Alongside the diagram, the report must also include textual explanations that describe each component and clarify their underlying principles of operation.

4.2 Database Design

Describe how data is organized in the system, including tables and their relationships. Explain the choice of database type (relational—such as MySQL or PostgreSQL—or NoSQL—such as MongoDB) and the rationale for that choice. Describe the system's principal entities and their attributes, and the relationships among entities. Provide details for each table: columns/fields, primary keys, foreign keys, and constraints. Use an ERD (or another suitable diagram) to illustrate entities and relationships. Note: if the diagram is very large, resize it appropriately so that the text remains legible and the layout remains aesthetically pleasing. Conversely, if the diagram is too small, adjust it so that, when embedded in the report, labels are still large enough to read; avoid scaling the entire diagram down to a tiny image that is difficult or impossible to decipher.

4.3 User Interface Design

Present how the system interacts with users through its interface, including screen structure, functionality, and usability.

- **Layout**: Present the system's main screens (e.g., Home, Account Management, Product pages).

- **User Flow**: Describe how users will progress through and interact with system functions.
- **Wireframes or Mockups**: Provide interface sketches to illustrate the design visually.

4.4 API and Service Design

Describe the APIs and services that the system will expose to other components or to external services.

- **API Design**: Describe the APIs provided by the system, including HTTP methods (GET, POST, PUT, DELETE) and endpoint URLs.
- **Data Exchange Model**: Present how data is transmitted via the APIs (e.g., JSON, XML).
- **API Documentation**: Provide detailed usage instructions, including input parameters and outputs.

4.5 Security Design

Present the security measures designed to protect the system against threats such as cyberattacks and data breaches.

- **Authentication and Authorization**: Describe how users sign in and how access rights are enforced.
- **Data Encryption**: Explain how sensitive data is encrypted (e.g., payment information).
- **Attack Mitigation Measures**: Describe protections against attacks such as SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF).

4.6 Performance and Scalability Design

Ensure the system can operate efficiently as the user base grows and can scale when necessary.

- **Performance Optimization**: Describe solutions such as caching and load balancing to improve response time.
- **Scalability**: Explain how the system can scale horizontally (adding servers) or vertically (upgrading servers).
- **High Availability Design**: Propose redundancy and failover measures to prevent service disruption in case of incidents.

CHAPTER 5. DEPLOYMENT & TESTING

This chapter focuses on how the system is implemented and configured based on the proposed design, as well as describing the group's software development process. This part is particularly important because it demonstrates how the design phase is transformed into practical deployment. The following aspects may be addressed:

- Development Environment and Tools: Present the tools and development environments used (e.g., IDEs, Git version control systems).
- Programming Languages and Frameworks: Describe in detail the programming languages and frameworks selected.
- Example: "The mobile application was developed using React Native, while the backend was built with Node.js and Express."
- Source Code Structure: Provide a brief explanation of the project's source code organization.
- Example: "The frontend components are divided into separate modules for easier management."
- Challenges During Development: Present the difficulties encountered during programming and how they were resolved.
- Example: "Optimization issues in the speech recognition model required switching from standard TensorFlow to TensorFlow Lite."
- Database Deployment: Describe the procedures for setting up and deploying the database, ensuring that the system can store and retrieve data efficiently.
- Software Deployment: Describe the process of deploying the application on servers or cloud services (if applicable).
  - Backend Deployment: Explain the installation and configuration of the backend, including setting up servers (e.g., Nginx, Apache) and related services.
  - Frontend Deployment: Describe how the web interface (SPA) is deployed on servers or hosting services such as Netlify or Vercel.
  - Integration with Database and APIs: Present the process of connecting the application to the database and APIs.
  - Example: Deploying a ReactJS application on Netlify while the Node.js backend connects to a database hosted on AWS RDS.

CHAPTER 6. DEPLOYMENT AND EVALUATION

This chapter focuses on testing activities to ensure that the system functions accurately and reliably. Testing not only helps detect errors but also verifies that the system fulfills the defined requirements. The following aspects may be included:

- **Testing Strategy**: Explain the testing methods employed (unit testing, integration testing, system testing).
- **Automated Testing**: Describe how automated testing was set up, if applicable. Example: "Jenkins was used to configure CI/CD pipelines and automate unit testing."
- **Performance Testing:** Evaluate the system's performance under heavy load. Example: "The system can process 200 speech recognition requests per second with an average response time of 300ms."
- **Usability Testing**: Present the results of end-user testing and improvements made based on feedback.
- **Bug Tracking and Fixing**: Explain how issues are managed and resolved (e.g., using JIRA, GitHub Issues).
- **Performance and Scalability Testing**: Ensure that the system performs efficiently under high load and can scale as needed. Measure response times, processing speeds, and system capacity under different conditions. Types of performance testing include:
  - **Load Testing**: Assess system behavior under normal load conditions.
  - **Stress Testing**: Evaluate how the system performs under extreme load.
  - **Endurance Testing**: Examine system performance during prolonged continuous operation.

CHAPTER 7: CONCLUSION

This chapter provides an analysis and interpretation of results, together with evaluations and reflections on the group's findings. It primarily summarizes the content presented in previous chapters and may be divided into three subsections:

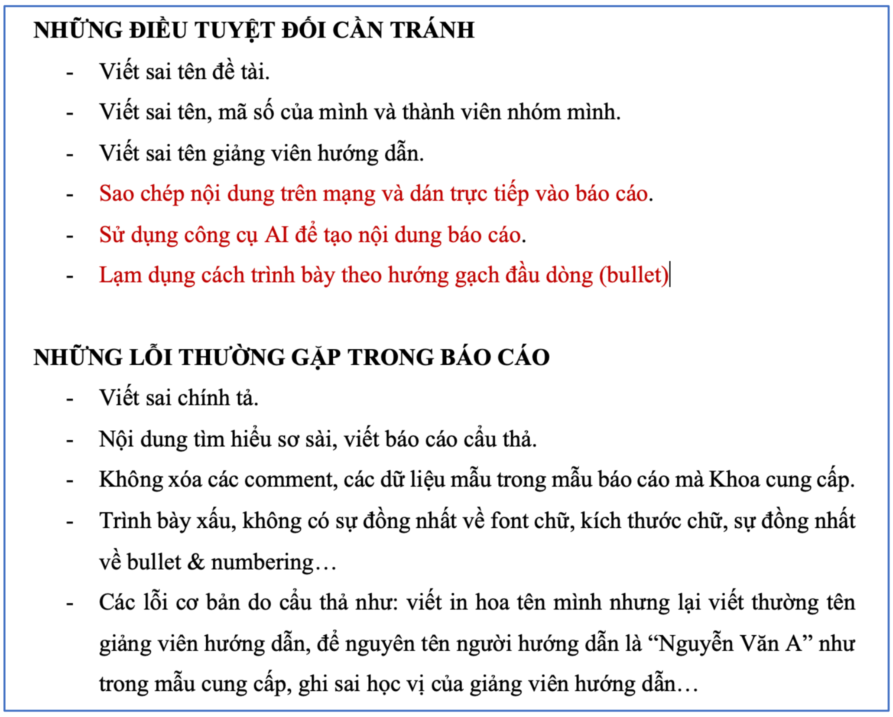
7.1 Achieved Results: Present the results of the group's product clearly and in detail. Analyze and interpret the outcomes, clarifying their significance and linking them to the research objectives.

7.2 Strengths and Limitations. Identify the advantages and shortcomings of the developed system.

7.3 Future Directions.

ABSOLUTE DON'Ts

- Misspelling the university name, faculty name, project title, or the names and student IDs of team members.
- Writing the supervisor's name or academic title incorrectly.
- Copying content directly from the internet and pasting it into the report.
- Using AI tools to generate report content.
- **Overusing bullet-point formatting.**



NHỮNG ĐIỀU TUYỆT ĐỐI CẦN TRÁNH

- Viết sai tên đề tài.
- Viết sai tên, mã số của mình và thành viên nhóm mình.
- Viết sai tên giảng viên hướng dẫn.
- Sao chép nội dung trên mạng và dán trực tiếp vào báo cáo.
- Sử dụng công cụ AI để tạo nội dung báo cáo.
- Lạm dụng cách trình bày theo hướng gạch đầu dòng (bullet)

NHỮNG LỖI THƯỜNG GẶP TRONG BÁO CÁO

- Viết sai chính tả.
- Nội dung tìm hiểu sơ sài, viết báo cáo cẩu thả.
- Không xóa các comment, các dữ liệu mẫu trong mẫu báo cáo mà Khoa cung cấp.
- Trình bày xấu, không có sự đồng nhất về font chữ, kích thước chữ, sự đồng nhất về bullet & numbering…
- Các lỗi cơ bản do cẩu thả như: viết in hoa tên mình nhưng lại viết thường tên giảng viên hướng dẫn, để nguyên tên người hướng dẫn là "Nguyễn Văn A" như trong mẫu cung cấp, ghi sai học vị của giảng viên hướng dẫn…

Example: Overuse of bullet points can significantly reduce the quality of a report. In essays, projects, or capstone reports, students should present content in well-structured paragraphs, each consisting of multiple continuous sentences, rather than relying excessively on bullet lists.

OTHER COMMON MISTAKES IN REPORTS

- Spelling errors: Students must carefully proofread for spelling mistakes before submitting the report for grading or even when sending it to the supervisor for review/feedback.
- Superficial content: Reports that are carelessly written, lacking depth and serious effort, are unacceptable.
- Failure to remove template artifacts: Leaving comments or sample data from the report template provided by the Faculty.

- Poor formatting: Inconsistencies in font style, font size, bullet & numbering formats, and line spacing.
- Careless errors: For example, writing one's own name in uppercase while writing the supervisor's name in lowercase, leaving the placeholder "Nguyễn Văn A" unchanged in the template, or recording the supervisor's academic title incorrectly.

RECOMMENDED PRACTICES

- Thorough proofreading: Carefully check for spelling errors before submission and before sending the report to your supervisor.
- Cover page presentation: Design the cover page meticulously. Write the project title in UPPERCASE, check the university logo, and replace it with a higher-resolution version if necessary, scaling it appropriately to fit the page.
- Formatting standards: Use Times New Roman, font size 13pt, line spacing 1.5, and justify the text. Headings should use a slightly larger size (e.g., 15pt).
- Table of contents: Generate the table of contents automatically. Limit it to three levels; avoid four or more unless truly necessary (e.g., section 4.1.2.4).
- Lists of figures/tables/equations: Provide these only if such items exist in the report. Do not include blank pages titled "LIST OF TABLES" when no tables are present.
- Paragraph writing: Prefer continuous prose. Use bullet points sparingly and always introduce or explain them. For example, in the "application functions" section, avoid merely listing features. Instead, name each function in a bullet point and follow it with descriptive paragraphs.
- Original writing: Do not copy-paste content from blogs. Read, understand, and rephrase in your own words.
- Figures, tables, diagrams: Always include captions in italic, centered, and numbered (e.g., Figure 1.2, Table 3.1, or sequentially Figure 1, Figure 2). Use a font 1–2pt smaller than body text.
- Tables: Use a font about 2pt smaller than the main text. Center content both vertically and horizontally where appropriate. Table headings should be bold and UPPERCASE; header cells may use a light gray background for clarity.
- Chapter layout: Each chapter must begin on a new page; never start a new chapter on the same page as the previous one.
- Source code inclusion: Limit code snippets. For special cases, present them in a readable code font with proper formatting. Do not paste screenshots from IDEs. Avoid black backgrounds in code blocks. Always add explanations for code snippets.

- Software installation details: Do not describe installation steps in excessive detail. Keep explanations concise (e.g., “To run Java, install the JDK, which can be downloaded from the official website, then configure environment variables.”).
- Introducing visuals: Every figure, table, or diagram must be introduced and explained in the text. Avoid adding screenshots without description.
- Example: Instead of simply labeling a screenshot “Login screen,” describe its functions, user actions, and sequence of interactions.
- Printing: Reports should be printed single-sided. Color printing is unnecessary unless for final theses or when emphasizing content in figures.
- Exploratory topics: For research-oriented assignments (e.g., “database connection methods in Java”), include comparisons and analysis of approaches—explain which method is preferable and under what circumstances.
- Completeness and detail: Ensure deliverables are fully functional and detailed.
- Example: A project titled “Exploring database connections in Java with a demo website” should not be limited to a trivial database with one record. A better approach is to create a database with multiple student records and implement functions such as listing, adding, deleting, and updating. Enhancements with frameworks like Bootstrap or jQuery are encouraged.
- Capstone projects: For topics like “building an online mobile phone store,” the submitted product must be a functioning website that meets requirements, with multiple categories (Samsung, Apple, Nokia, etc.), each containing several products with varying prices. Each product should ideally include features such as an image slideshow. Avoid presenting incomplete products and excusing missing features by saying “we can add those quickly later.” Remember: the submitted product is graded as-is. If it is incomplete, your grade will reflect that.

Analogy: This is similar to startup pitch competitions. Even if the product idea is excellent, a poor presentation may fail to convince investors. Likewise, your project must be as polished as possible when submitted on the e-learning platform and during oral defense sessions.

Important Note: From the moment these guidelines are provided, all groups are responsible for reading, understanding, and applying them in their reports. In addition, students are encouraged to conduct further research to present their reports in the best possible manner.