National University of Computer and Emerging Sciences, Lahore

FYP Report Title

Name of Student 1 Roll No. 1 BS(CS/DS/SE)

Name of Student 2 Roll No. 2 BS(CS/DS/SE)

Name of Student 3 Roll No. 3 BS(CS/DS/SE)

Supervisor: SupervisorName

Final Year Project

October 11, 2025

Anti-Plagiarism Declaration

This is to declare that the above publication was produced under the:

Title: FYP Report Title

is the sole contribution of the author(s), and no part hereof has been reproduced as it is the basis (cut and paste) that can be considered Plagiarism. All referenced parts have been used to argue the idea and cited properly. I/We will be responsible and liable for any consequence if a violation of this declaration is determined.

Date: ....

Name: ....

Signature: ....

Name: ....

Signature: ....

Name: ....

Signature: ....

Author's Declaration

This states Authors' declaration that the work presented in the report is their own, and has not been submitted/presented previously to any other institution or organization.

Abstract

An Abstract is a short summary of the work being reported. Three to five sentences describing the essence of the work. An abstract is a short, 50-125 words summary of a work. An Abstract should state your work's purpose, findings, and conclusion without commenting on or evaluating the work itself. It should be only one paragraph. Put the abstract on a separate page that follows the title page.

Executive Summary

The executive summary should be one to two pages' of an overview of the information contained in the FYP report. It should give the reader an easy reference, in a very brief form, to the important information contained in the report and explained in more detail in the body of the report. People reading the report will use this section as a reference during presentations.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

This chapter is mandatory. The introduction should contain a brief overview of the problem being addressed and the background information needed for the reader to understand the work being done and the reasoning behind it. The last paragraph of the introduction chapter should contain an outline of the entire report. Summarize each chapter in one line to make the last paragraph.

1.1 Purpose of this Document

Specify the purpose of this Project. The purpose of the document in an FYP report is to clearly and concisely state the main objective and goals of the project, as well as the research question(s) or hypothesis(es) being investigated. It should provide a brief overview of the background and context of the project, and explain why the research is important and relevant. The purpose of the document should be written in a way that is easy for the reader to understand, and should be included at the beginning of the report.

In addition to providing an overview of the project, the purpose of the document should also outline the specific objectives and goals of the research, such as what the project aims to achieve and what the research questions or hypotheses are. This will help the reader understand the scope of the project and what to expect from the report. It is also important to include a brief description of the methods and techniques used in the research, as well as any limitations or assumptions of the project. This will provide the reader with a clear understanding of how the research was conducted and what the results and conclusions are based on. Finally, it should be clear what the report is going to cover, what kind of information will be presented, and in what order.

For Example, ”The purpose of this document is to present the design, implementation, and evaluation of our Final Year Project (FYP) which aims to develop a web-based application for small businesses to manage their inventory and customer orders. The project will focus on creating a user-friendly interface and providing a seamless user experience. The goal of this research is to provide a Minimum Viable Product (MVP) that can be used by small businesses to improve their inventory management and customer order processing. The research question we aim to answer is: Can we develop a web-based application that can improve inventory management and customer order processing for small businesses? This report will present the methodology, design, implementation, testing, and evaluation of our project, as well as the limitations and future work.”

1.2 Intended Audience

Identifying and writing the intended audience for a Final Year Project (FYP) report is an important step in the writing process. The intended audience is the group of people for whom the report is written and who will primarily read and use the report. It is crucial to identify the intended audience early in the writing process as this will guide the report's style, tone, and level of detail.
To identify the intended audience, consider the report's purpose, who will be using the report and for what purpose. For example, the intended audience for a FYP report in Computer Science may be a panel of professors who will be evaluating the project. In contrast, the intended audience for a FYP report in Business may be potential investors or clients.

1.3 Definitions, Acronyms, and Abbreviations

List all important definitions, acronyms, and abbreviations used in this document. For example: **SDG**: Sustainable Development Goal
FYP: Final Year Project
MVP: Minimum Viable Product
UI: User Interface
UX: User Experience
Agile: Agile Development
SCRUM: Scrum Development
REST: Representational State Transfer
ORM: Object-Relational Mapping
CRUD: Create, Read, Update, Delete
API: Application Programming Interface

1.4 Conclusion

The last paragraph of the introduction chapter should contain an outline of the entire report. Summarize each chapter in one line to make the last paragraph.

Chapter 2 Project Vision

For each related work provide a paragraph of introduction and at the end, a paragraph of conclusions. Give a page break after the chapter ends. This chapter is mandatory. Clearly specify the goals and objectives of the project along with the scope of the project. (You can make sub-heading of goals and objectives and the scope of the project).

2.1 Problem Domain Overview

Describe in full detail what your system does. This section can be further divided into sub-sections, but basically, a paragraph description of the system is required.

2.2 Problem Statement

Provide the **problem** which you are trying to solve.

2.3 Problem Elaboration

Describe the problem in detail and the various sub-problems you would work on.

2.4 Goals and Objectives

Write goals and objectives here.

2.5 Project Scope

Specify the scope of your project here.
For example, The scope of this project is to design and develop a web-based application for small businesses to manage their inventory and customer orders. The application will allow small businesses to:

- Create, read, update, and delete inventory items
- Create, read, update, and delete customer orders
- Generate reports on inventory levels and customer orders
- Send notifications to customers about their orders
- Provide an easy-to-use user interface and a seamless user experience

The project will be developed using a combination of HTML, CSS, JavaScript, and a web-development framework such as React.js or Angular.js. The application will use a RESTful API to communicate with a back-end database, which will be implemented using a ORM library. The project will be developed using Agile development principles, and project management will be done using SCRUM.
The project deliverables will include:

- A functional web-based application for small businesses to manage their inventory and customer orders
- A user manual for the application
- A documentation of the design, implementation, and testing process

The project scope does not include the following:

- Providing hosting or server infrastructure for the application
- Developing mobile applications (iOS or Android)
- Integrating the application with third-party services or APIs

This project scope will be used as a guide throughout the development

2.6 Sustainable Development Goal (SDG)

Identify at least one SDG area that you are targeting in this project. This has to be documented as a paragraph. The complete SDG list is shown in Figure **??**.

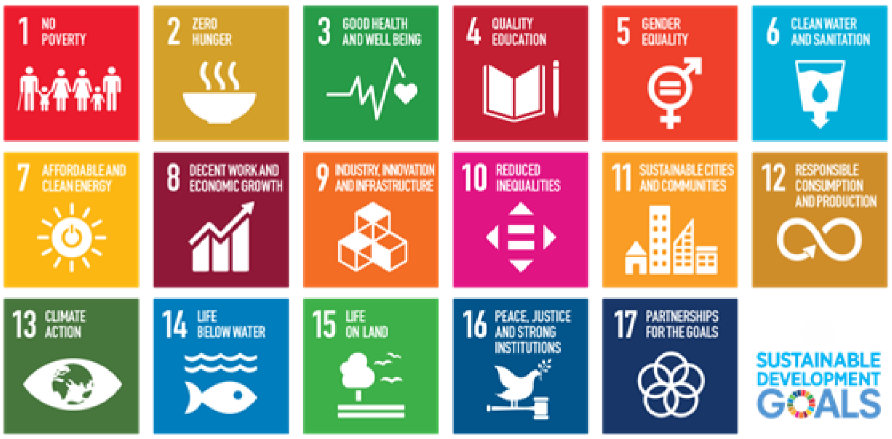
1 NO POVERTY
2 ZERO HUNGER
3 GOOD HEALTH AND WELL BEING
4 QUALITY EDUCATION
5 GENDER EQUALITY
6 CLEAN WATER AND SANITATION
7 AFFORDABLE AND CLEAN ENERGY
8 DECENT WORK AND ECONOMIC GROWTH
9 INDUSTRY, INNOVATION AND INFRASTRUCTURE
10 REDUCED INEQUALITIES
11 SUSTAINABLE CITIES AND COMMUNITIES
12 RESPONSIBLE CONSUMPTION AND PRODUCTION
13 CLIMATE ACTION
14 LIFE BELOW WATER
15 LIFE ON LAND
16 PEACE, JUSTICE AND STRONG INSTITUTIONS
17 PARTNERSHIPS FOR THE GOALS
SUSTAINABLE DEVELOPMENT GOALS

Figure 2.1: This figure represents all the SDGs that can be targets of an FYP; Adding this sub-caption is mandatory; make sure you describe the picture in a way that if a visually impaired person is reading using a text reader, they can understand whats in this picture.

2.7 Constraints

2.8 Business Opportunity

2.9 Stakeholders Description/ User Characteristics

Identify and briefly describe who the users of this system will be, and their roles.

2.9.1 Stakeholders Summary

2.9.2 Key High-Level Goals and Problems of Stakeholders

Notice how a section's heading is changed to "heading 1". It will show up in the table of contents as a section under a chapter. Notice its numbering also. Similarly, subsections can have heading 2 and heading 3 styles.

NOTE: Subsections from 2.7 to 2.9 are optional for Research projects but compulsory for Development projects.

Chapter 3 Literature Review / Related Work

For each related work, provide a paragraph of introduction and, at the end, a paragraph of conclusions. Give a page break after the chapter ends. This chapter is mandatory.

3.1 Definitions, Acronyms, and Abbreviations

3.2 Detailed Literature Review

This section will include a detailed literature review of your problem area. Make different categories for different types of work done in the past. In addition to textual descriptions, make a summary table that describes each paper that you have read, along with references.

3.2.1 Related Research Work 1 / Related Development Project (Please make sure to add an appropriate heading here and not keep this same heading, this is your info only.)

- Summary of the research item / Development Project (1 or 2 paragraphs)
- Critical analysis of the research item / Development Project (Strengths and Weaknesses) 1 paragraph
- Relationship to the proposed work - 1 paragraph

3.2.2 Example LR 1: for Development Project

The project ABC [1] focused on building a real-time customer service dashboard designed to streamline query management and improve response efficiency within a mid-sized software firm. The system was developed using a modular architecture, integrating ticket tracking, live chat support, and performance analytics into a single interface. Built with React for the frontend and Node.js for the backend, the dashboard also featured role-based access control and SLA monitoring tools. During testing, the dashboard demonstrated a 35% reduction in query resolution time and improved internal coordination between support agents and technical teams. The project was deployed in phases, starting with internal use and later expanding to client-facing support channels.

One of the key strengths of this development effort was its adaptability to existing workflows. The dashboard was designed to integrate seamlessly with the company's CRM and internal communication tools, minimizing disruption during rollout. Its real-time analytics allowed supervisors to monitor agent performance and customer satisfaction trends. However, the project faced limitations in scalability during peak traffic hours, and the initial version lacked multilingual support, which restricted its usability for international clients. These issues were partially addressed in later iterations but remain areas for future enhancement.

This development project directly supports the proposed research by providing a practical foundation for evaluating customer service efficiency. While the research explores theoretical models and AI integration, the dashboard serves as a tangible implementation that can be enhanced with AI-driven features such as automated query routing and sentiment analysis. The performance metrics gathered during this project will inform the research's evaluation framework and help validate proposed improvements in service delivery and operational management.

3.2.3 Example LR 2: for Research and Development Project

Smith et al. [2] investigated the integration of AI-powered chatbots within corporate customer service frameworks, focusing on automating Tier-1 support queries. Their study involved deploying a Natural Language Processing (NLP) based chatbot across three multinational firms and evaluating its performance over a six-month period. The results demonstrated a 40% reduction in average response time and a 25% increase in customer satisfaction ratings. The chatbot was trained using historical customer interaction logs and was integrated with existing CRM systems to ensure contextual accuracy and personalized responses. Additionally, the study emphasized the scalability of chatbot systems, noting their ability to handle thousands of simultaneous queries without compromising quality, especially during peak service hours when traditional support teams struggled to maintain SLA compliance.

One of the strengths of this research lies in its empirical approach, offering quantifiable improvements in service metrics. The integration with CRM systems added depth to the chatbot's responses, enhancing the overall customer experience. However, the study's limitations include its narrow focus on Tier-1 queries, with minimal exploration of escalation protocols or emotional intelligence in handling sensitive customer issues. It also lacks discussion on the long-term adaptability of the chatbot to evolving customer expectations and organizational changes.

This research is directly relevant to the proposed work, which aims to develop a hybrid customer service model that combines AI efficiency with human oversight. While they focused primarily on automation, the proposed work seeks to extend their framework by incorporating feedback loops, escalation mechanisms, and sentiment analysis to handle more complex queries. The foundational insights from this study will inform the design and evaluation metrics of the proposed system, particularly in terms of response time, customer satisfaction, and operational scalability.

3.3 Literature Review Summary Table

Please provide a compact summary of your literature review as provided in the table 3.1, for the development project. The Table above explains a way to present your literature review for development

Table 3.1: This summary Table is for a development project, and you must identify each column clearly. Must need at least 15 relevant studies

Application	Features	Relevance	Limitations
Rapid Miner [3]	User interface design	Provides information on user requirements and design considerations	Limited sample size and geographic diversity of participants
TensorFlow [4]	Competitive analysis	Identifies potential competitors and opportunities for differentiation	Limited to apps in the same category and not considering other possible alternatives
Abc Keras [5]	Technical infrastructure	Helps to identify and plan for necessary technical resources and infrastructure	Assumes all requirements are known and does not evaluate the possibility of new requirements emerging in the future
Abc Xyz [1]	User experience testing	Provides feedback on user experience and identifies areas for improvement	Limited sample size and testing environment not fully representative of real-world conditions

projects. If you have a research project, the summary should be written in the below table 3.2 format.

Table 3.2: This summary Table is for a research project, and you must identify each column clearly. Must need at least 15 relevant studies

Author	Method	Results	Limitations
Smith et al. [2]	Case study with a sample size of 50	Positive effects of XYZ on ABC, 93.74% accuracy	Small sample size
Johnson et al. [3]	Literature review utilizing keyword-based search	A comprehensive overview of XYZ in the field of DEF, 12.8% precision	Limited to specific subfield
Williams et al. [6]	Survey of 1000 participants using a structured questionnaire	Strong correlation between XYZ and GHI, R-squared = 0.72	Self-reported data
Jones et al. [4]	Experimental design with 100 participants, Utilized SVM with the combination of PCA	XYZ significantly improves JKL in certain populations, AUC = 0.86	Limited generalizability

3.4 Conclusion

Write the conclusion of the literature review. The columns in the table depending upon your problem and should be specific to your project as described in 3.1, 3.2, respectively. Proper citations and references are necessary. The template has already an integrated IEEE references format. Go to Google scholar, copy the bib text of your reference, and paste the reference into the fypbib.bib file. Then all you have to do is use [2].

NOTE: For development projects, describe related or similar work done by other teams and details of their methods/algorithms. For a research project, a detailed literature survey is expected.

Chapter 4 Software Requirement Specifications

Describe all modules of requirements and design in clear English text along with the necessary diagram and figures. Anyone reading your report should be able to reproduce your system/results after reading it. It describes functional requirements, design constraints, and other factors necessary to provide a complete and comprehensive description of the requirements for the software.
For each chapter, provide a paragraph of introduction and at the end, a paragraph of conclusions. Make sure no heading/subheading is blank. Write text to introduce each section as well. NOTE: This Chapter is optional for Research projects but compulsory for Development projects.

4.1 List of Features

List all important features of your system here.

4.2 Functional Requirements

The functional requirements fully describe the external behavior of the system. Identify and list each functionality and give a brief description, along with the user of each functionality.

4.3 Quality Attributes

4.4 Non-Functional Requirements

This section should describe all the non-functional requirements, including reusability, performance (how many maximum users can access it at a time), extensibility, etc.

4.5 Assumptions

List down all the assumptions made for the specification.

4.6 Use Cases

This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements, or if they stress or illustrate a specific, delicate point of the architecture.
The table 4.1 provides a simple use case.

Table 4.1: A sample use case is given in the above table. You have to follow this tabular format for use cases. Following is the description of the content required in this section. No separate heading is required or use case tables. Keep them all under a generic use case heading. Also, In the previous version, usecase tables dont have cpation, in this version, we do have.

Name	Sample Use Case Name Here		
Actors	Admin, Business Owner, Store Manager		
Summary	The user shall provide their email and password on the login form, and after successful verification, redirect the user to the home page.		
Pre-Conditions	The user must be in the database records, either added by any of the authorized users or added manually by a developer. The user must not already be logged in.		
Post-Conditions	The user's session is successfully established and shall be redirected to the home page.		
Special Requirements	None		
Basic Flow			
Actor Action		**System Response**	
1	The user opens the login page.	2	The login page is displayed asking for email and password.
2	The user enters valid email and password.	4	The system verifies the email and password, establishes a session for the user and redirects the user to the home page.
Alternative Flow			
3	The user enters invalid email or password.	4-A	The system responds with an error message: Incorrect email or password entered.

This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.

Summary: The description briefly conveys the role and purpose of the use case. A single paragraph will suffice for this description.

Basic Flow: This use case starts when the actor does something. An actor always initiates use cases. The use case describes what the actor does and what the system does in response. It is phrased in the form of a dialog between the actor and the system.

The use case describes what happens inside the system, but not how or why. If information is exchanged, be specific about what is passed back and forth. For example, it is not very illuminating to say that the actor enters customer information. It is better to say the actor enters the customer's name and address. A Glossary of Terms is often useful to keep the complexity of the use case manageable you may want to define things like customer information there to keep the use case from drowning in details.

Simple alternatives may be presented within the text of the use case. If it only takes a few sentences to describe what happens when there is an alternative, do it directly within the **Flow of Events** section. If the alternative flow is more complex, use a separate section to describe it. For example, an Alternative

Flow subsection explains how to describe more complex alternatives.

A picture is sometimes worth a thousand words, though there is no substitute for clean, clear prose. If it improves clarity, feel free to paste graphical depictions of user interfaces, process flows or other figures into the use case. If a flow chart is useful to present a complex decision process, by all means use it! Similarly for state-dependent behavior, a state-transition diagram often clarifies the behavior of a system better than pages upon pages of text. Use the right presentation medium for your problem, but be wary of using terminology, notations or figures that your audience may not understand. Remember that your purpose is to clarify, not obscure.

First Alternative Flow: More complex alternatives are described in a separate section, referred to in the Basic Flow subsection of Flow of Events section. Think of the Alternative Flow subsections like alternative behavior each alternative flow represents alternative behavior usually due to exceptions that occur in the main flow. They may be as long as necessary to describe the events associated with the alternative behavior. When an alternative flow ends, the events of the main flow of events are resumed unless otherwise stated.

Second Alternative Flow: There may be, and most likely will be, a number of alternative flows in a use case. Keep each alternative flow separate to improve clarity. Using alternative flows improves the readability of the use case, as well as preventing use cases from being decomposed into hierarchies of use cases. Keep in mind that use cases are just textual descriptions, and their main purpose is to document the behavior of a system in a clear, concise, and understandable way.

Special Requirements: A special requirement is typically a nonfunctional requirement that is specific to a use case, but is not easily or naturally specified in the text of the use case's event flow. Examples of special requirements include legal and regulatory requirements, application standards, and quality attributes of the system to be built including usability, reliability, performance or supportability requirements. Additionally, other requirements such as operating systems and environments, compatibility requirements, and design constraints should be captured in this section.

Pre-Conditions: A pre-condition of a use case is the state of the system that must be present prior to a use case being performed.

Post-Conditions: A post-condition of a use case is a list of possible states the system can be in immediately after a use case has finished.

4.7 Hardware and Software Requirements

List the hardware and software requirements that will be required to develop and deploy the project.

4.7.1 Hardware Requirements

4.7.2 Software Requirements

4.8 Graphical User Interface

This section should give the GUI dumps of each screen, with reference to the users. The navigation flow of each user is also required, and each GUI should mark the functionality/use case that it covers.

4.9 Database Design (if required; this means if you are using noSQL, you will not provide ER but the other design element and data dictonary should still be there. Explain and elaborate your DB design)

4.9.1 ER Diagram

4.9.2 Data Dictionary

4.10 Risk Analysis

List and explain the risks that may be encountered during the project. For e.g.: technical risks, business risks, etc.

Chapter 5 Proposed Approach and Methodology

The proposed approach could be a Framework, Heuristic, Algorithm, Protocol, or Mathematical-model. For each chapter, provide a paragraph of introduction and in the end a paragraph of conclusions. Not the programming code but the algorithmic and procedural details especially related to the hidden/ backend algorithms that are not covered in the design.

NOTE: This chapter is optional for Development projects but compulsory for Research projects.

Chapter 6 High-Level and Low-Level Design

Provide the high- and low-level design of your system in this chapter. Select the design that is appropriate for your project.

NOTE: This Chapter is compulsory for both R&D projects and Development projects.

6.1 System Overview

Provide a general description of the software system, including its functionality and matters related to the overall system and its design (perhaps including a discussion of the basic design approach or organization). Feel free to split this discussion up into subsections (and sub-subsections, etc. ...).

6.2 Design Considerations

This section describes many issues that need to be addressed or resolved before attempting to devise a complete design solution.

6.2.1 Assumptions and Dependencies

Describe any assumptions or dependencies regarding the software and its use. These may concern such issues as:

- Related software or hardware
- Operating systems
- End-user characteristics
- Possible and/or probable changes in functionality

Do not just add random generic statements, be clear and concise about what you are writing.

6.2.2 General Constraints

Describe any global limitations or constraints that have a significant impact on the design of the system's software (and describe the associated impact). Such constraints may be imposed by any of the following (the list is not exhaustive):

- Hardware or software environment
- End-user environment
- Availability or volatility of resources

- Standards compliance
- Interoperability requirements
- Interface/protocol requirements
- Data repository and distribution requirements
- Security requirements (or other such regulations)
- Memory and other capacity limitations
- Performance requirements
- Network communications
- Verification and validation requirements (testing)
- Other means of addressing quality goals
- Other requirements described in the requirements specification

6.2.3 Goals and Guidelines

Describe any goals, guidelines, principles, or priorities which dominate or embody the design of the system's software. Such goals might be:

- The KISS principle (”Keep it simple stupid!”)
- Emphasis on speed versus memory use
- Working, looking, or ”feeling” like an existing product

For each such goal or guideline, unless it is implicitly obvious, describe the reason for its desirability. Feel free to state and describe each goal in its own sub subsection if you wish.

6.2.4 Development Methods

Briefly describe the method or approach used for this software design. If one or more formal/published methods were adopted or adapted, then include a reference to a more detailed description of these methods. If several methods were seriously considered, then each such method should be mentioned, along with a brief explanation of why all or part of it was used or not used.

6.3 System Architecture

This section should describe both the internal architecture of the modules, as well as the external architecture of the system with other systems, if any. Diagrammatic architecture is compulsory.

This section should provide a high-level overview of how the functionality and responsibilities of the system were partitioned and then assigned to subsystems or components. Do not go into too much detail about the individual components themselves (there is a subsequent section for detailed component descriptions). The main purpose here is to gain a general understanding of how and why the system was decomposed, and how the individual parts work together to provide the desired functionality.

At the top-most level, describe the major responsibilities that the software must undertake and the various roles that the system (or portions of the system) must play. Describe how the system was broken down into its components/subsystems (identifying each top-level component/subsystem and the roles/responsibilities assigned to it). Describe how the higher-level components collaborate with each other in order to achieve the required results. Don't forget to provide some sort of rationale for choosing this particular decomposition of the system (perhaps discussing other proposed decompositions and why they were rejected). Feel free to make use of design patterns, either in describing parts of the architecture (in pattern format), or for referring to elements of the architecture that employ them.

If there are any diagrams, models, flowcharts, documented scenarios or use-cases of the system behavior and/or structure, they may be included here (unless you feel they are complex enough to merit being placed in the Detailed System Design section). Diagrams that describe a particular component or subsystem should be included within the particular subsection that describes that component or subsystem.

Note: This section (and its subsections) really applies only to newly developed (or yet-to-be developed) portions of the system. If there are parts of the system that already existed before this development effort began, then you only need to describe the pre-existing parts that the new parts of the system depend upon, and only in enough detail sufficient to describe the relationships and interactions between the old parts and the new parts. Pre-existing parts that are modified or enhanced need to be described only to the extent that is necessary for the reader to gain a sufficient understanding of the nature of the changes that were made.

6.3.1 Subsystem Architecture

If a particular component is one which merits a more detailed discussion than what was presented in the System Architecture section, provide that more detailed discussion in a subsection of the System Architecture section (or it may even be more appropriate to describe the component in its own design document). If necessary, describe how the component was further divided into subcomponents, and the relationships and interactions between the subcomponents (similar to what was done for top-level components in the System Architecture section).

If any subcomponents are also deemed to further merit discussion, then describe them in a separate subsection of this section (and in a similar fashion). Proceed to go into as many levels/subsections of discussion as needed in order for the reader to gain a high-level understanding of the entire system or subsystem (but remember to leave the gory details for the Detailed System Design section).

If this component is very large and/or complex, you may want to consider documenting its design in a separate document and simply including a reference to it in this section. If this is the option you choose, the design document for this component should have an organizational format that is very similar (if not identical to) this document.

6.4 Architectural Strategies

Explain all strategies that you use for designing of the Architecture. Describe any design decisions and/or strategies that affect the overall organization of the system and its higher-level structures. These strategies should provide insight into the key abstractions and mechanisms used in the system architecture. Describe the reasoning employed for each decision and/or strategy (possibly referring to previously stated design goals and principles) and how any design goals or priorities were balanced or traded-off. Such decisions might concern (but are not limited to) things like the following:

- Use of a particular type of product (programming language, database, library, etc. ...)
- Reuse of existing software components to implement various parts/features of the system
- Future plans for extending or enhancing the software
- User interface paradigms (or system input and output models)
- Hardware and/or software interface paradigms
- Error detection and recovery
- Memory management policies
- External databases and/or data storage management and persistence
- Distributed data or control over a network
- Generalized approaches to control
- Concurrency and synchronization
- Communication mechanisms

- Management of other resources

6.4.1 Architectural Strategies Strategy-1 name or description

6.4.2 Architectural Strategies Strategy-2 name or description

Each significant strategy employed should probably be discussed in its subsection or (if it is complex enough) in a separate design document (with an appropriate reference here, of course). Make sure that when describing a design decision you also discuss any other significant alternatives that were considered and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose). Sometimes it will be most effective to employ the ”pattern format” for describing a strategy.

6.5 Domain Model/Class Diagram

In this subsection, add the Class Diagram of your system. Class diagrams represent the structure design of your system.

6.6 Policies and Tactics

Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures) but which nonetheless affect the details of the interface and/or implementation of various aspects of the system. Such decisions might concern (but are not limited to) things like the following:

- Choice of which specific product to use (compiler, interpreter, database, library, etc. ...)
- Engineering trade-offs
- Coding guidelines and conventions
- The protocol of one or more subsystems, modules, or subroutines
- The choice of a particular algorithm or programming idiom (design pattern) to implement portions of the system’s functionality
- Plans for ensuring requirements traceability
- Plans for testing the software
- Plans for maintaining the software
- Interfaces for end-users, software, hardware, and communications

- Hierarchical organization of the source code into its physical components (files and directories).
- How to build and/or generate the system's deliverables (how to compile, link, load, etc. ...)

6.6.1 Policy/tactic-1 name or description

6.6.2 Policy/tactic-2 name or description

Each particular policy or set of tactics employed should probably be discussed in its own subsection, or (if it is large or complex enough) in a separate design document (with an appropriate reference here of course). Make sure that when describing a design decision that you also discuss any other significant alternatives that were considered, and your reasons for rejecting them (as well as your reasons for accepting the alternative you finally chose). For this reason, it may frequently be convenient to use one of the more popular ”pattern formats” to describe a given tactic.

For this particular section, it may become difficult to decide whether a particular policy or set of tactics should be discussed in this section, or in the System Architecture section, or in the Detailed System Design section for the appropriate component. You will have to use your own ”best” judgment to decide this. There will usually be some global policies and tactics that should be discussed here. Still, decisions about interfaces, algorithms, and/or data structures might be more appropriately discussed in the same (sub) section as its corresponding software component in one of these other sections.

Chapter 7 Implementation and Test Cases

For each chapter, provide a paragraph of introduction and at the end, a paragraph of conclusions. (Not the programming code but the algorithmic and procedural details especially related to the hidden/ backend algorithms that are not covered in the design)

7.1 Implementation

Whatever implementation that you have done so far, please elaborate here. Give clear details of the algorithms that were implemented along with the platform and the APIs which were used. NOTE: For FYP-1, this chapter can be changed to description of prototype developed.

7.1.1 Implementation of First Component/Algorithm

Write the implementation of the first component of your system here.

7.2 Test case Design and description

This section will be added in FYP-II. Summarize the common attributes of test cases. This may include input constraints that must be true for every input in the set of associated test cases, any shared environmental needs, any shared special procedural requirements, and any shared case dependencies. The following scheme is recommended for describing test cases in detail. The Sample test case is provided in table 7.1.

Table 7.1: Sample Test case No.1

<Software Component Name>			
<Reference>			
Test Case ID:	*Reference Number*	**QA Test Engineer:**	*Name of personnel*
Test case Version:	*Version number*	**Reviewed By:**	*Testing Team lead*
Test Date:	*Date*	**Use Case Reference(s):**	*Relation to use cases*
Revision History:	*Refer to previous test case identity (if any)*		
Objective:	*Need and scope of the testing*		
Product/Ver/ Module:	*Refer to overall system being built and the place of this test case in it.*		
Environment:	*Necessary and desired properties of the test environment.*		
Assumptions:	*Assumptions that might affect the testing process.*		
Pre-Requisite:	*Necessary condition that needs to be fulfilled before the test case.*		
Step No.	**Execution description**	**Procedure result**	
	Events being tested.	*Mention software response.*	
Comments:			
Passed Failed Not Executed			

7.3 Test Metrics

Summarize here the common ground of attributes of test case metrics as explained in Table 7.2.

Table 7.2: Sample Test case Matric.No.1

Metric	**Purpose**
Number of Test Cases	Total number of test cases that you have developed for your system.
Number of Test Cases Passed	The number of test cases that successfully passed
Number of Test Cases Failed	The number of test cases that failed
Test Case Defect Density	(No of test cases failed * 100) No of test cases executed
Test Case Effectiveness	No of defects detected using test cases *100 Total number of defects detected
Traceability Matrix	Traceability is the ability to determine that each the feature has a source in requirements and each requirement has a corresponding implemented feature.

7.3.1 Sample Test case Metric.No.2

Similar to the above table, add here for test metrics 2,3,4.

Chapter 8 User Manual

This chapter will be added in FYP-II. Design a complete user manual of your application and add that in this chapter. NOTE: This Chapter is optional for Research projects but compulsory for Development projects.

Chapter 9 Experimental Results and Discussion

This chapter will be added in FYP-II. Give proper analysis and discussion of experimental results (in plain English text) along with tables of results. For each chapter provide a paragraph of introduction and in the end a paragraph of conclusions.

NOTE: This Chapter is optional for Development projects but compulsory for Research projects.

Chapter 10 Conclusions

10.1 aa

10.1.1 dfs

10.1.1.1 dsd

This chapter is mandatory. Give conclusions and summary of the work done. What were your findings and what were the results? Discuss in detail whether the scope of your project was entirely covered or not and whether the objectives of the project were met or not. What challenges did you face and what has been left out and why.

Sum up all the conclusions of all the chapters here to make a final conclusion chapter. Do not repeat any text, just summarize it in different words. Give recommendations for future work also.

For FYP-1 it is mandatory to list down a plan of the work to be done for FYP-2.

Bibliography

[1] I. Guyon, A. Saffari, G. Dror, and G. Cawley, "Analysis of the IJCNN 2007 agnostic learning vs. prior knowledge challenge," *Neural Networks*, vol. 21, pp. 544–550, December 2008.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[3] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. John Wiley and Sons, 2000.

[4] C.-C. Chang and C.-J. Lin, *LIBSVM: A Library for Support Vector Machines*, 2001. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] I. Guyon, A. Saffari, G. Dror, and G. Cawley, "Agnostic learning vs. prior knowledge challenge," in *Proceedings of International Joint Conference on Neural Networks*, August 2007.

[6] "Agnostic Learning Vs. Prior Knowledge Challenge by Internation Joint Conference on Neural Networks (IJCNN)." Available: http://www.agnostic.inf.ethz.ch [Accessed November 02, 2018].

Chapter A First Appendix if Required

If you want to add appendices then make sure you provide a proper title. Add different appendices by using the chapter command as shown for this one. They will automatically be added to the table of contents.

A.1 References

See how .bib file is prepared. The bibliography would be automatically added by Latex. So here is how you cite papers (you'll need to view the .bib file to understand). [3] is the correct way to make one citation and [2, 4] to make multiple citations. You can see various reference styles for manuals [4], conferences [5], websites [6] and journal articles [1]. Also, note that all citations appear in the order in which they appear within text. They are not sorted according to author names.

A.2 Equations

Make sure you use the math environment for writing equations. Again, make sure you do not hard code equation numbers. Use label and ref commands of Latex to number them and reference them. Here is an example of (A.1)

$$E = mc^2 \tag{A.1}$$

Note, how the equation is referenced by its equation number within brackets.

A.3 Figures

To insert a figure, you can use the following code as template. Here the figure is referenced using ref command of Latex. So Figure A.1 is the FAST logo. DO NOT HARD CODE FIGURE NUMBERS. Let Latex manage them for you.

Figure A.1: Fast Logo is the Caption.

A.4 Tables

Note how tables can be added in Latex. Also, make sure you use label and ref commands to define and use table numbers. You can use Table A.1 as a template to make tables, add captions to them and reference them. DO NOT HARD CODE TABLE NUMBERS. Let Latex manage their numbering for you.

If you are having problems in managing Latex tables, you may use Lyx `https://www.lyx.org/`.

Table A.1: Give a Caption to the Table.

No.	**Column 1**	**Column 2**
1.	Some data	More data
2.	Some values	More values

A.5 Pseudo Code

When you need to explain an existing algorithm of an algorithm you devised, then you should use pseudo code notation. Given below, algorithm 1, is an example. Note how the numbering is managed by the Latex itself.

```
Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while not at end of this document do
    read current;
    if understand then
        go to next section;
        current section becomes this one;
    else
        go back to the beginning of current section;
    end
end
```

Algorithm 1: How to write algorithms

A.6 Code of Programming Languages

If you need to add C++, Java or any other programming language's code, then you can use "listings" package. Following is an example. Normally, the code is only added in the appendices to avoid clutter in the document.

```
#include <stdio.h>
#define N 10
/* Block
 * comment */

int main()
{
    int i;

    // Line comment.
    puts("Hello world!");

    for (i = 0; i < N; i++)
    {
        puts("LaTeX is also great for programmers!");
    }

    return 0;
}
```

A.7 Recommendations for LaTeX

Here we are providing you with only two Latex source files. main.tex and fypbib.bib file. You can also start a project at `overleaf.com` and upload the zip file of this template there and share the Latex source between your group and supervisor. This way all of you can work on the Latex files together online.