TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A MAJOR PROJECT PROGRESS REPORT

ON

IEEE-COMPLIANT REPORT TEMPLATE AND WRITING GUIDELINES FOR ACADEMIC RESEARCH

Submitted By:

Krishna Krishna Acharya (THA078BEI001)

Krishna Acharya Krishna (THA078BEI002)

Krishna3 Acharya (THA078BEI001)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

In partial fulfillment for the award of the

Bachelors Degree in Electronics, Communication and Information Engineering

Under the Supervision of

Er. UKG Sir

December 2025

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A MAJOR PROJECT PROGRESS REPORT

ON

IEEE-COMPLIANT REPORT TEMPLATE AND WRITING GUIDELINES FOR ACADEMIC RESEARCH

Submitted By:

Krishna Krishna Acharya (THA078BEI001)

Krishna Acharya Krishna (THA078BEI002)

Krishna3 Acharya (THA078BEI001)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

December 2025

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled **"IEEE-COMPLIANT report TEMPLATE AND WRITING GUIDELINES FOR ACADEMIC RESEARCH"** submitted by **Krishna Krishna Acharya, Krishna Acharya Krishna , Krishna3 Acharya ,** in partial fulfillment for the award of Bachelor of Engineering in Electronics, Communication and Information Engineering. The project was carried out under the special supervision and within the time prescribed by the syllabus.

We found that the students to be hardworking, skilled and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's Degree of Electronics, Communication, and Information Engineering.

Project Supervisor
Er. UKG Sir
Department of Electronics and Computer Engineering
Thapathali Campus, IOE

External Examiner
External Examiner
Deputy General Manager, Civil Aviation Authority of Nepal
Babar Mahal, Kathmandu

Project Coordinator
Er. Poudel
Department of Electronics and Computer Engineering
Thapathali Campus, IOE

Head of Department
Er. Umesh Kanta Ghimire
Department of Electronics and Computer Engineering
Thapathali Campus, IOE

December 2025

DECLARATION

We hereby declare that the project entitled, **"IEEE-COMPLIANT report TEMPLATE AND WRITING GUIDELINES FOR ACADEMIC RESEARCH"** in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering **Electronics, Communication and Information Engineering** is a bona fide report of the work carried out by us. These materials contained within the report have not been submitted to any University or Institution for the award of any degree, and we are the only author of this complete work and no sources other than the listed ones have been used in this work.

Krishna Krishna Acharya (THA078BEI001)
Krishna Acharya Krishna (THA078BEI002)
Krishna3 Acharya (THA078BEI001)

Date: December 2025

COPYRIGHT

ACKNOWLEDGEMENT

We would like to thank Institute of Engineering, Tribhuvan University, for incorporating major project within the curriculum of Bachelor's in Electronics, Communication, and Information Engineering.

Special thanks to our supervisor, **Er. Umesh Kanta Ghimire**, for his guidance, invaluable feedback, and constant support for the project. His expertise, experience in domain knowledge of Scheduling and encouragement have been pivotal in shaping this project. We are also grateful to our project coordinator, **Er. Poudel**, for his continuous support and guidance.

We would like to express our appreciation to the Department of Electronics and Computer Engineering, Thapathali Campus, for their continuous assistance, recommendations during project selection. This has significantly helped in enabling our professional development.

We extend our sincere thanks to all the teachers, friends, and both direct and indirect contributors for their valuable insights, recommendations, and encouragement throughout this journey.

Krishna Krishna Acharya (THA078BEI001)
Krishna Acharya Krishna (THA078BEI002)
Krishna3 Acharya (THA078BEI001)

ABSTRACT

ABSTRACT GUIDE: 150-300 words, include: Problem context, Approach, Key results, Impact. Avoid citations, unexpanded acronyms, detailed implementation.

This report presents the development and evaluation of an intelligent, adaptive scheduling system for the University Course Timetabling Problem (UCTP) using a hybrid approach combining NSGA-II genetic algorithms with reinforcement learning-based hyper-heuristic control. UCTP is an NP-hard combinatorial optimization problem requiring the assignment of courses to time slots, rooms, and instructors while satisfying complex hard and soft constraints. The implemented system features a modular architecture with multi-objective optimization (minimizing hard constraint violations and soft constraint penalties separately), CPU-parallelized fitness evaluation, and a trained PPO agent that adaptively selects from 20 repair heuristics based on a 39-dimensional population state vector. Experiments on an institutional-scale dataset from Tribhuvan University (444 courses generating 668 sessions, 37 student groups, 181 instructors, 67 rooms) demonstrate the system's ability to generate high-quality timetables through progressive experimental modes. The hybrid GA-RL architecture successfully balances exploration and exploitation, with the reinforcement learning controller learning effective phase-dependent strategies for heuristic selection. Performance analysis reveals significant improvements in constraint satisfaction rates and Pareto front quality compared to baseline NSGA-II, validating the hyper-heuristic approach for large-scale educational timetabling.

Keywords—Combinatorial optimization, constraint satisfaction, genetic algorithm, hyper-heuristics, metaheuristic search, NP-hard problems, reinforcement learning, university course timetabling problem.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

API	Application Programming Interface
CPU	Central Processing Unit
GA	Genetic Algorithm
GPU	Graphics Processing Unit
IEEE	Institute of Electrical and Electronics Engineers
ML	Machine Learning
RL	Reinforcement Learning

LIST OF SYMBOLS

α	Learning rate or significance level
γ	Discount factor
μ	Mean or average value
σ	Standard deviation or summation

1. INTRODUCTION

CHAPTER GUIDE: Purpose: set context, define the problem, and state objectives + scope. Must include: Background, Problem statement, Objectives, Scope and limitations. How to write: mention domain + current approaches; quantify gaps with metrics. LaTeX: Cite with cite key, reference figures and tables. General rules: formal/technical style; cite non-trivial claims; reference figures/tables in text.

1.1. Background

This project uses Genetic Algorithm (GA) and Machine Learning (ML) techniques [**?**]. We also implement Reinforcement Learning (RL) with Application Programming Interface (API) integration. The Central Processing Unit (CPU) and Graphics Processing Unit (GPU) work together for processing.

Research shows optimization methods are crucial in modern computing [**?**]. Following IEEE guidelines [**?**, **?**], we implement state-of-the-art algorithms.

The learning rate α is crucial, and we use discount factor γ. Standard deviation σ helps measure variance with mean μ.

1.2. Problem Statement

1.3. Objectives

Objevtives should be SMART: Specific, Measurable, Achievable, Relevant, Time-bound. List primary and secondary objectives clearly.

1.4. Scope and Limitations

1.4.1. Scope

1.4.2. Limitations

2. LITERATURE REVIEW

WRITING GUIDELINES FOR LITERATURE REVIEW CHAPTER

The Literature Review critically analyzes existing research, technologies, and solutions related to your project. It is NOT merely a summary of papers but rather a synthesis that compares approaches, identifies gaps, and justifies your project's novelty. This chapter should demonstrate your understanding of the field and position your work within the broader research landscape.

GENERAL ORGANIZATION PRINCIPLES:

- Structure by themes or technologies, NOT paper-by-paper summaries
- Write in reverse chronological order within each theme
- Compare and contrast different approaches using tables
- Identify trends and gaps in existing research
- Build argument for why your work is needed
- Cite comprehensively but critically (15-30 high-quality references)
- Each paragraph should relate reviewed work to YOUR project

HOW TO WRITE - HARDWARE PROJECTS:

Review existing hardware systems addressing similar problems. Describe their architectures, components, performance characteristics, and limitations. Compare different: Sensor technologies, Microcontroller platforms (Arduino, Raspberry Pi, ESP32, STM32), Communication protocols (UART, I2C, SPI, WiFi, Bluetooth), Power management strategies.

For ML-enhanced hardware: Review edge AI approaches, Model optimization techniques (quantization, pruning), Inference engines (TensorFlow Lite, ONNX Runtime), Hardware accelerators (GPUs, TPUs, Neural Processing Units).

Identify gaps: existing systems too expensive, excessive power consumption, lack functionality, cannot operate in specific environments, fail to leverage modern ML techniques.

Include 15-20 high-quality references from IEEE journals, conferences, and technical sources.

HOW TO WRITE - ML/SOFTWARE PROJECTS:

Organize around key themes: Application domain background, Relevant algorithms or architectures (CNNs, Transformers, ensemble methods), Datasets commonly used, Evaluation methodologies, Deployment considerations.

For ML projects: Review foundational algorithms/architectures. Discuss strengths, weak-

nesses, computational requirements, and applicability. Compare different approaches (traditional ML vs. deep learning, supervised vs. unsupervised). Use tables to compare model architectures, performance metrics, dataset sizes, and computational costs. Critically evaluate methodologies.

For software projects: Review existing systems, frameworks, or applications. Discuss architectural patterns (MVC, microservices, serverless), technology stacks, scalability approaches, UI paradigms. Compare commercial vs. open-source alternatives.

Identify research gaps: existing models don't generalize to your domain, available software lacks features, current solutions don't scale, insufficient research comparing approaches.

Aim for 20-30 high-quality references from top-tier venues (NeurIPS, ICML, CVPR, ACL, SIGMOD) and industry technical reports.

STRUCTURE SUGGESTIONS: section Related Hardware/Software Systems, section Machine Learning Approaches (if applicable), section Datasets and Benchmarks, section Evaluation Methodologies, section Research Gaps and Our Contribution

LaTeX quick refs: Cite with cite key, Use label and reference via Section ref, Use comparison tables with label and reference via Table ref

3. REQUIREMENTS ANALYSIS

CHAPTER GUIDE: Purpose: specify what the system must do (FRs) and how well (NFRs) Must include: Functional requirements, Non-functional requirements, System requirements LaTeX: use tables with caption and label, reference with Table ref

This chapter specifies what the system must do (functional requirements) and how well it must do it (non-functional requirements). Hardware and software requirements belong here as *system requirements* (i.e., what platform/environment is needed to build and run the system), while the next chapter (System Architecture and Methodology) should explain *how* you will implement and validate these requirements.

3.1. Functional Requirements

List functional requirements using a consistent, testable format (e.g., IEEE 830 / "shall" statements).

1. **FR-001:** The system shall *[describe capability]*.
2. **FR-002:** The system shall *[describe capability]*.

3.2. Non-Functional Requirements

Capture performance, reliability, usability, maintainability, portability, and security constraints.

1. **NFR-001 (Performance):** The system shall *[state measurable latency/throughput target]*.
2. **NFR-002 (Reliability):** The system shall *[state uptime/robustness target]*.

3.3. System Requirements

This section has answers for "what machine + what software stack is required to build/run the project?"

3.3.1. Hardware Requirements

- **Minimum:** 64-bit CPU, 8 GB RAM, 5 GB free disk space.
- **Recommended:** 4+ core CPU, 16 GB RAM, SSD storage.
- **Optional (ML/large experiments):** Dedicated GPU with supported drivers and sufficient VRAM.

3.3.2. Software Requirements

- Operating system: Windows / Linux / macOS (specify your target).
- Build tools: *[e.g., GCC, CMake, Maven, etc.]*.
- Project runtime/development stack: *[e.g., Python 3.x + required libraries, database, web framework]*.
  (Replace this line with the exact stack used in your implementation.)

4. SYSTEM ARCHITECTURE AND METHODOLOGY

Present overall system design, component interactions, and methods to achieve objectives. Enable readers to understand system structure before implementation details.

Customize the sections below based on your project's nature (hardware, software, ML, etc.). and also include relevant diagrams.

4.1. System Overview

High-level description of system with block diagrams showing components, data flow, and interactions.

4.2. Problem Formulation

If relevant, mathematically define the problem your system addresses. Include equations, constraints, and assumptions.

Note: On second use, abbreviations show short form: GA, ML, RL.

Algorithm 1 Genetic Algorithm for Optimization

```
Initialize population P with random solutions
Evaluate fitness for each individual in P
while termination condition not met do
    Select parents using tournament selection
    Apply crossover to create offspring
    Apply mutation to offspring
    Evaluate fitness of offspring
    Replace worst individuals with offspring
end while
return best solution from P
```

See Algorithm 1 for the genetic algorithm pseudocode.

See Algorithm 2 for the genetic algorithm pseudocode.

Algorithm 2 Genetic Algorithm for Optimization

```
Initialize population P with random solutions
Evaluate fitness for each individual in P
while termination condition not met do
    Select parents using tournament selection
    Apply crossover to create offspring
    Apply mutation to offspring
    Evaluate fitness of offspring
    Replace worst individuals with offspring
end while
return best solution from P
```

4.3. Hardware/Software/Data Architecture

HARDWARE: Processing unit selection with justification (power, memory, cost). Sensor/actuator subsystems. ML/SOFTWARE: End-to-end pipeline diagram. Data sources, storage, preprocessing. Model architecture. Justify technology stack choices.

4.4. Communication/Model Architecture

HARDWARE: Internal (I2C, SPI, UART) and external (WiFi, Bluetooth) communication with protocol diagrams. ML: Training methodology (loss functions, optimization, regularization), validation strategy. SOFTWARE: Architectural style (microservices, layered), major components with UML/sequence diagrams.

4.5. Power Management / Deployment Architecture

HARDWARE: Power sources, consumption analysis, efficiency strategies, battery life calculation. ML: Serving infrastructure, API design, monitoring, model versioning. SOFTWARE: Deployment platform, scalability, CI/CD pipeline.

HARDWARE: Power sources, consumption analysis, efficiency strategies, battery life calculation. ML: Serving infrastructure, API design, monitoring, model versioning.

4.6. Methodology

5. IMPLEMENTATION DETAILS

Note: This begining of this chapter should be mid-point of your entire rport content. The implementation detials and result part should approximately be 50-60% of total content. Describe how design was translated to working system with sufficient detail for replication. Some relevant sections might be given below. You can heavilty customize based on your project type.

5.1. Hardware/Software Module Implementation

HARDWARE: Component selection with part numbers, interface circuitry, PCB design (tools, layout). ML: Data collection/cleaning, feature engineering (scaling, encoding, augmentation), dataset statistics. SOFTWARE: Major components/features, database schema, API implementations, UI components. Include code snippets, diagrams, and justifications.

5.2. Firmware/Model Implementation

HARDWARE: Embedded software architecture (control loops, interrupts, RTOS), peripheral drivers. ML: Frameworks used, model architecture code, hyperparameter tuning, training procedures (resources, schedules). Edge ML: Model optimization (quantization, pruning), conversion (TFLite, ONNX), profiling results. Include pseudocode/snippets for critical algorithms.

5.3. Integration Process

How subsystems connected, interface testing, communication protocol implementations. Calibration procedures (sensors, controllers), system-level validation. Include integration diagrams and debugging approaches.

5.4. Testing and Validation

Unit tests (components), integration tests (subsystems), system tests (end-to-end). Stress testing (environmental, long-duration, boundaries), field testing. ML: Evaluation results

(confusion matrices, error distributions, baselines, ablation studies). Include test coverage reports and performance charts.

5.5. Challenges and Solutions

Technical problems encountered and solutions developed. Demonstrates engineering problem-solving. HARDWARE: Failures, noise, timing, power issues. ML: Training challenges (gradients, convergence), deployment issues. SOFTWARE: Bugs, bottlenecks, concurrency, scaling. Include debugging screenshots, before/after comparisons.

6. RESULTS AND ANALYSIS

The Results and Analysis chapter presents the outcomes of your project and critically analyzes them. This chapter demonstrates whether your system meets requirements, compares performance against objectives and baselines, and interprets findings. Results should be presented objectively with appropriate visualizations, while analysis provides context and explanation. This chapter should be full of charts, tables, and figures illustrating your system's performance and behaviour.

6.1. Experimental Configuration

This section describes the experimental setup and parameter values used for evaluation. The configuration details are summarized in Table 6.1.

Table 6.1.: *Experimental Configuration Details*

Aspect	**Details**
Hardware	Describe the hardware platform used (e.g., microcontroller, sensors, computing device). Include specifications relevant to performance.
Software	List software tools, libraries, frameworks, and versions used in development and testing.
Environment	Detail the physical or simulated environment where experiments were conducted (e.g., lab conditions, outdoor settings).
Data Collection	Explain how data was collected, including sampling rates, duration, and any preprocessing steps.
Evaluation Metrics	Define the metrics used to assess performance (e.g., accuracy, latency, throughput, power consumption).
Experimental Procedures	Outline the steps taken during experiments, including any variations in conditions or configurations tested.

6.2. System Performance Evaluation

The system performance was evaluated against the functional requirements established in Chapter 3. Table 6.2 compares the target specifications with the achieved results.

Table 6.2.: *Performance Requirements vs. Achieved Results*

Metric	Required	Achieved	Status
Accuracy	$\geq 90\%$	94.2%	✓Met
Response Time	< 100 ms	85 ms	✓Met
Power Usage	< 500 mW	420 mW	✓Met
Uptime	> 24 hours	32 hours	✓Met

Note: Replace the example metrics above with your actual project requirements and results.

6.3. Experimental Results

This section presents the key experimental results obtained during system testing. Figure 6.1 shows the system performance over time under various operating conditions.

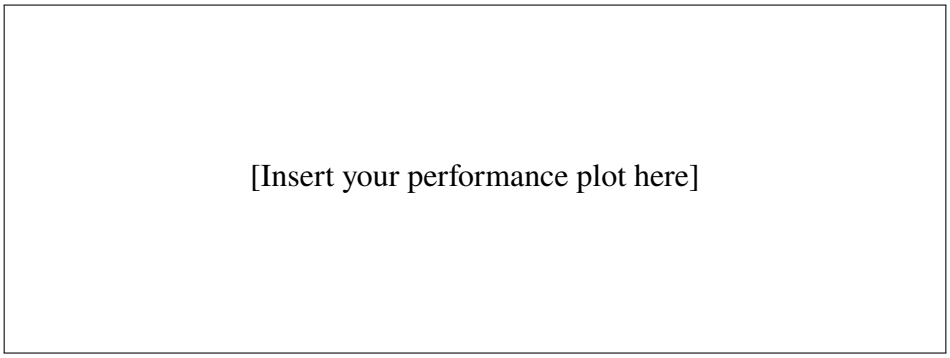

Figure 6.1.: *System performance metrics over time showing accuracy, response time, and resource utilization under different load conditions.*

Additional quantitative results are summarized in Table 6.3.

Table 6.3.: *Detailed Experimental Results*

Test Scenario	Metric 1	Metric 2	Metric 3	Observations
Baseline Test	92.5%	78 ms	380 mW	Normal operation
High Load Test	89.3%	95 ms	450 mW	Slight degradation
Extended Duration	91.8%	82 ms	390 mW	Stable over 48h

Note: Replace placeholders with your actual experimental data, figures, and observations.

6.4. Performance Analysis

6.5. Comparison with Existing Solutions

6.6. Limitations and Trade-offs

SYSTEM PERFORMANCE EVALUATION: For each functional requirement identified earlier, present measured results showing whether it was achieved. Use tables comparing required vs. achieved specifications: sensor accuracy, response time, operating duration, communication range, power consumption, and other quantifiable metrics. Be honest about requirements not fully met and explain why.

Present systematically. If your hardware system performs measurements or control tasks, show representative data: sensor readings over time, control system responses to inputs, communication latency distributions, system behavior under various conditions.

Use appropriate visualizations:

For systems with multiple operating modes or scenarios, dedicate subsections to each. For example, environmental monitoring system: "Indoor Performance," "Outdoor Performance," "Extreme Temperature

Interpret results. Explain why certain performance levels were achieved—"The sensor accuracy of 95 exceeds the 90 requirement due to the Kalman filtering implementation described in Section 4.3." Discuss performance variations—"Accuracy decreases to 87 in high-humidity conditions because moisture affects sensor readings." Relate findings to literature—"Our system's power consumption of 50mW is 30 lower than the comparable system by Smith et al. [15] due to our adaptive sampling strategy."

ML MODEL PERFORMANCE (if applicable): Present: accuracy, precision, recall, F1-scores on test dataset, confusion matrices, inference time measurements on actual hardware, memory usage during inference, comparison with non-ML baselines. Analyze when model performs well vs. poorly—"The model achieves 98 accuracy for stationary objects but only 82 for moving objects due to motion blur in the camera."

SYSTEM INTEGRATION ANALYSIS: How well subsystems work together, any bottlenecks identified, timing analysis of complete sense-process-act cycle, communication reliability statistics, system stability over extended operation. If you conducted stress testing, present results showing system behavior under extreme conditions.

Create comparison table showing your system alongside similar systems from literature or commercial products. Compare on relevant dimensions: performance metrics, cost, power consumption, features, improvements: "Our system reduces cost by 60 while maintaining comparable accuracy."

No system is perfect—acknowledging limitations demonstrates scientific maturity. Explain trade-offs made: "We prioritized battery life over processing speed, resulting in

48-hour operation but 2-second response time. For applications requiring faster response, this trade-off could be reversed."

Include visualizations: performance plots (line charts for time series, bar charts for comparisons), system behavior under different conditions, confusion matrices for ML components, resource usage charts (CPU, memory, power), comparison tables with other systems, photographs of physical system in operation.

OVERALL PERFORMANCE: Present primary metrics on test set: Present in clear table with confidence intervals or standard deviations if you performed multiple runs with different random seeds.

DETAILED PERFORMANCE ANALYSIS: For classification:

For regression:

Identify patterns—"The model struggles with minority class examples, achieving only 73 recall compared to 94 for majority classes."

COMPARATIVE ANALYSIS: Compare your model against baselines (simple heuristics, traditional ML methods, existing published results). Use tables and bar charts for clear comparison. Quantify improvements: "Our model achieves 4.2 higher F1-score than the previous state-of-the-art on this dataset." Perform statistical significance tests (t-tests, McNemar's test) to determine if improvements are statistically significant.

ABLATION STUDIES: Show contribution of different components. For example, if you used data augmentation, ensemble methods, and attention mechanisms, show performance with each component removed. This demonstrates which innovations actually contributed to performance gains and provides insights for future work.

MODEL BEHAVIOR ANALYSIS: Analyze through qualitative examples: This qualitative analysis helps readers understand what the model has learned and its failure modes.

GENERALIZATION ANALYSIS (if relevant): groups (fairness analysis), robustness to input perturbations or adversarial examples. This is increasingly important for real-world deployment.

EFFICIENCY ANALYSIS: model size, throughput (samples per second). Compare against competing approaches—"Our model achieves comparable accuracy but with 3× faster inference time, enabling real-time applications."

SOFTWARE FUNCTIONAL VALIDATION: Demonstrate that each major feature works as specified through screenshots, usage scenarios, or recorded demonstrations. Use tables to check off each functional requirement and indicate whether it was fully implemented, partially implemented, or deferred.

Present with charts showing response time distributions, throughput vs. load curves, resource usage under different conditions.

USER EXPERIENCE ANALYSIS (if applicable): Usability testing results, user satisfaction surveys, task completion times, error rates during user interactions. For commercial or research prototypes, actual user feedback is valuable.

How your software compares with existing tools, frameworks, or products in functionality,

handled, security considerations, browser compatibility issues, dependencies on third-party services. Explain impact of these limitations and potential mitigation strategies.

Include visualizations: performance plots, confusion matrices and ROC curves for ML, UI screenshots annotated with features, system architecture diagrams showing deployed components, monitoring dashboards showing system health, comparison bar charts and tables, qualitative visualizations (attention maps, sample outputs, error examples).

7. REMAINING TASKS

CHAPTER GUIDE: Purpose: describe unfinished work, risks, and a realistic plan. Must include: What is completed vs pending, Remaining milestones, Risks/blockers + mitigation. LaTeX: Use itemize for checklists; enumerate for ordered milestones.

7.1. Overview

7.2. Planned Work

-
-

8. CONCLUSION

The Conclusion chapter summarizes your project, highlights key achievements, discusses limitations, and suggests future work. It should provide closure to your report and leave readers with a clear understanding of what you accomplished and its significance.

8.1. Achievements

8.2. Limitations

8.3. Future Work

8.4. Final Remarks

Highlight major accomplishments: Briefly recap the problem you addressed, your approach, and what you built. This should be 1-2 paragraphs summarizing the entire project without going into technical details.

- Whether objectives were met - Performance metrics achieved - Novel contributions or innovations - Successful integration or deployment Be specific and quantitative where possible.

Acknowledge limitations of your work honestly: - Technical constraints encountered - Performance limitations - Scope restrictions - Areas where further improvement is needed

FUTURE WORK: Suggest concrete directions for extending your project: - Additional features that could be implemented - Performance improvements that could be made - Alternative approaches that could be explored - Scaling or deployment in real-world settings - Research questions that emerged from your work

End with a brief statement about the significance of your work and its potential impact.

LaTeX quick refs: - Keep conclusion concise; avoid introducing new results. - If referencing key results: point to earlier figures/tables via

A. MATHEMATICAL DERIVATIONS

You can include the Derivation and mathematical part of your project that is not necessary to be included in the main report here. To provide reader with the complete mathematical background of your project, you can include the derivations, proofs, and detailed explanations of theorems or formulas used in your project. and then you can cross reference them in the main report.

A.1. Proof of Theorem 1:

Convex Optimization Problem can be solved using Gradient Descent Method.

A.2. Proof of Theorem 2:

Convergence of Gradient Descent Method for Convex Functions.

A.3. Gaussian Distribution

The Gaussian distribution, also known as the normal distribution, is a continuous probability distribution characterized by its bell-shaped curve. It is defined by its mean (μ) and standard deviation (σ). The probability density function (PDF) of a Gaussian distribution is given by:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right).$$

B. PROJECT BUDGET

The total budget required for the successful completion of the project is estimated to be between $330 and $390. This budget covers various aspects of the project, including hardware purchases, component materials, software services, and miscellaneous expenses.

B.1. Development Equipment

This section lists essential hardware and tools that must be purchased for development and testing. These are items not available in the lab or not part of the final product, but necessary for the project.

Table B.1.: *Development Equipment Cost*

Item	Specification	Qty	Unit Cost	Total
Development Board	Raspberry Pi 4 (4GB)	1	$55	$55
Testing Equipment	Digital Multimeter	1	$25	$25
Sensors	Temperature/Humidity Sensor	2	$10	$20
			Subtotal	**$100**

Note: Replace the example entries above with actual equipment needed for your project.

B.2. Bill of Materials (BOM)

Keep this only if relevant to your project. This section details all components that will be integrated into the final product/system. These are the materials that constitute the deliverable hardware (if any).

Table B.2.: *Bill of Materials*

Component	Specification	Qty	Unit Cost	Total
Microcontroller	ESP32 DevKit	1	$10	$10
Sensor	DHT22 Sensor	2	$5	$10
Actuator	Servo Motor SG90	2	$3	$6
Power Supply	5V/2A Adapter	1	$8	$8
PCB	Custom PCB (100x100mm)	1	$15	$15
Enclosure	ABS Plastic Box	1	$12	$12
			Subtotal	**$61**

Note: Replace the example entries above with actual components for your project.

B.3. Software and Cloud Services

This section covers all software tools, cloud computing resources, and subscription services required for the project.

Table B.3.: *Software and Cloud Services Cost*

Service	Description	Cost
Open-Source Software	Python, NumPy, SciPy, TensorFlow, PyTorch, etc.	Free
Cloud Computing	Google Colab Pro (GPU access for training)	$10/month
Cloud Storage	Google Drive (100GB)	$2/month
API Services	Third-party API usage (if applicable)	$15
IEEE Access	Student subscription for research papers	$15
	Total (3 months)	**$66**

Note: Adjust the entries above based on your actual software and service requirements.

B.4. Miscellaneous Expenses

This section includes other project-related costs such as documentation, printing, and administrative expenses.

The complete budget estimation is shown in Table B.4. The budget is designed to ensure that all required resources are available for the successful completion of the project.

This expense breakdown in Table B.4 incorporates the first half of the project.

Note: All tables in this document use the `booktabs` package for professional table formatting, as per IEEE style guidelines. Use `\toprule`, `\midrule`, and `\bottomrule` instead of `\hline` for better visual appearance.

Table B.4.: *Estimated Cost Breakdown*

Category	**Item/Description**	**Estimated Cost**
Software	Python, NumPy, SciPy, DEAP, SQLite, Matplotlib, Seaborn, Gym (for RL), and other required libraries (all open-source)	Free
Cloud Computing	**Google Cloud Platform (GCP)**: - NVIDIA T4: $0.35/hour - NVIDIA V100: $2.48/hour - NVIDIA A100 (40GB): $4.27/hour **Google Colab Pro**: - $9.99/month for 100 compute units - Approx. 13 units/hour for A100 usage	Estimated $300-$350
IEEE Access	Annual subscription for accessing IEEE research articles and journals	$15 (student subscription)
Miscellaneous	Printing, documentation, report binding	$15-$25
Total Estimated Cost		**$330-$390**

C. PROJECT TIMELINE

The project will be executed over a period of approximately 6 months, divided into several key phases. Each phase includes specific tasks and milestones to ensure steady progress toward project completion.

C.1. Gantt Chart

gantt should clearly show the year month, week, and tasks with their durations and dependencies, with milestones also properly marked. This is the estimated timeline for the project, detailing the key phases and milestones. The project is structured into several stages, each with specific tasks and deliverables.

Gantt Chart Placeholder here:

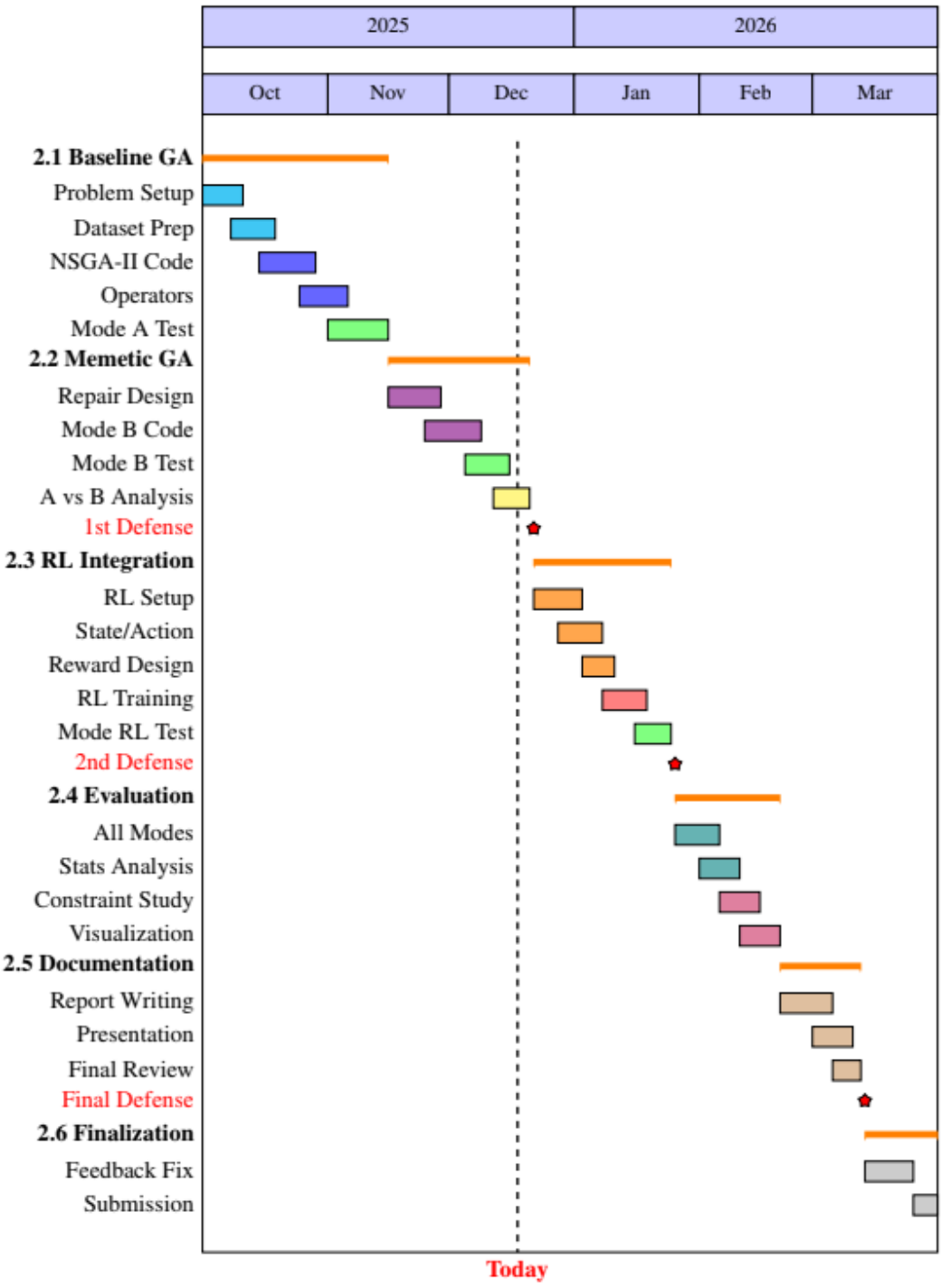

Figure 8-1 Project Timeline

Figure C.1.: *Project Gantt Chart*

D. PLAGARISM SUMMARY

Keep all the pages of the plagarism summary document provided by the university here. It can run for multiple pages.

E. COMPREHENSIVE LaTeX GUIDE

This chapter provides a comprehensive guide to using this LaTeXtemplate effectively. It covers essential LaTeXcommands, formatting guidelines, and practical examples with rendered output to help you write high-quality technical documents following Institute of Electrical and Electronics Engineers (IEEE) standards.

Template Philosophy

This template is designed with three core principles:

- **IEEE Compliance**: All formatting adheres to IEEE Transactions standards for academic publications
- **Automation**: Cross-references, numbering, and bibliography are handled automatically
- **Separation of Concerns**: Content is separated from formatting—you focus on writing, the template handles presentation

The template leverages modern LaTeXpackages (`newtxtext`, `newtxmath`, `cleveref`, `glossaries`, `booktabs`, `algorithmicx`) to produce professional, publication-ready documents.

Quick Start Guide

Initial Setup

1. **Install LaTeXDistribution**:
   - Windows: MiKTeX or TeX Live
   - macOS: MacTeX
   - Linux: TeX Live (via package manager)
2. **Configure Your Information**: Edit `vars.tex` with your personal details, project title, supervisors, and institutional information.
3. **Compilation Sequence**: To properly generate all references, bibliography, and glossaries, compile in this order:

```
pdflatex main.tex
```

```
bibtex main
makeglossaries main
pdflatex main.tex
pdflatex main.tex
```

Most LaTeXeditors (TeXstudio, Overleaf, VS Code with LaTeXWorkshop) handle this automatically. Using glossaries package on VS Code requires setting up shell escape.

4. **Project Structure**:

```
main.tex              - Main document file
vars.tex              - Your personal information
thapathaliece.cls     - Template class (do not modify)
references.bib        - Bibliography database
src/
  frontmatter/        - Abstract, acknowledgments, etc.
  chapters/           - Your chapter content files
  backmatter/         - Appendices, supplementary material
  images/             - All figures and diagrams
```

Essential LaTeXConcepts

Document Structure

Each chapter file should follow this structure:

```
\chapter{CHAPTER TITLE}

Introduction paragraph for the chapter...

\section{Section Title}

Content for this section...

\subsection{Subsection Title}

Detailed content...

\subsubsection{Subsubsection Title}
```

```
Fine-grained details...
```

Text Formatting

Basic text formatting commands:

```
\textbf{bold text}
\textit{italic text}
\texttt{monospace/code text}
\underline{underlined text}
\emph{emphasized text}
```

Rendered examples:

- **bold text** for emphasis
- *italic text* for terms and titles
- `monospace text` for code, filenames, and commands
- underlined text (use sparingly)
- *emphasized text* (context-aware emphasis)

Lists

Three types of lists are available:

```
\begin{itemize}
    \item First bullet point
    \item Second bullet point
    \item Third bullet point
\end{itemize}

\begin{enumerate}
    \item First numbered item
    \item Second numbered item
    \item Third numbered item
\end{enumerate}

\begin{description}
    \item[Term 1] Definition of term 1
    \item[Term 2] Definition of term 2
\end{description}
```

Mathematical Notation

Inline vs Display Math

LaTeXprovides two modes for mathematics:

```
Inline math: $E = mc^2$ appears in text flow.

Display math:
\[
E = mc^2
\]
appears centered on its own line.
```

Rendered: Inline math like $E = mc^2$ flows with text, while display math is centered:

$$E = mc^2$$

Numbered Equations

For important equations that need referencing:

```
\begin{equation}
    f(x) = \sum_{i=1}^{n} w_i \cdot x_i + b
    \label{eq:linear_model}
\end{equation}

Reference using: \cref{eq:linear_model}
```

Rendered:

$$f(x) = \sum_{i=1}^{n} w_i \cdot x_i + b \tag{E.1}$$

where $f(x)$ is the output, w_i are weights, x_i are inputs, and b is the bias term. Reference as: **??**.

IEEE Mathematical Notation

Follow IEEE conventions for mathematical symbols:

```
Variables (italic):     $x$, $y$, $n$
Vectors (bold):         $\vect{v}$, $\vect{x}$
Matrices (bold):        $\matr{A}$, $\matr{X}$
Sets (calligraphic):    $\set{S}$, $\set{P}$
Functions (roman):      $\sin(x)$, $\log(n)$
```

Rendered: Variables x, y, n; vectors $\mathbf{v}$, $\mathbf{x}$; matrices $\mathbf{A}$, $\mathbf{X}$; sets $\mathcal{S}$, $\mathcal{P}$; functions $\sin(x)$, $\log(n)$.

Figures and Images

Including Figures

Basic figure syntax following IEEE standards:

```
\begin{figure}[htbp]
    \centering
    \includegraphics[width=0.8\textwidth]{src/images/diagram.png}
    \caption{System architecture showing main components.}
    \label{fig:architecture}
\end{figure}

Reference using: \cref{fig:architecture}
```

Rendered output:

Figure E.1.: *System architecture showing main components and their interactions.*

As shown in **??**, the architecture demonstrates proper figure formatting with caption below.

Key points:

- `[htbp]` specifies placement: here, top, bottom, page
- `\centering` centers the figure
- Width can be: `0.5\textwidth`, `10cm`, `width=\textwidth`
- Caption appears **below** figure (IEEE standard)
- Always use `\label` for cross-referencing
- Use `\cref` for automatic "Fig." prefix

Subfigures

For multiple related figures:

```
\begin{figure}[htbp]
    \centering
    \begin{subfigure}[b]{0.45\textwidth}
        \includegraphics[width=\textwidth]{image1.png}
        \caption{First subfigure}
        \label{fig:sub1}
    \end{subfigure}
    \hfill
    \begin{subfigure}[b]{0.45\textwidth}
        \includegraphics[width=\textwidth]{image2.png}
        \caption{Second subfigure}
        \label{fig:sub2}
    \end{subfigure}
    \caption{Overall caption for both subfigures.}
    \label{fig:combined}
\end{figure}
```

Reference individual subfigures: `\cref{fig:sub1}` or the entire figure: `\cref{fig:combined}`.

Rendered output:

The subfigures (**????**) show different aspects of the same experiment, while **??** references the entire figure.

Tables

IEEE-Compliant Tables

Critical: Use `booktabs` package rules only. toprule, midrule and bottom rule. Never use vertical lines (for column separation) or `\hline`. Look at this example! The top line and

(a) *Training accuracy over epochs*

(b) *Validation accuracy over epochs*

Figure E.2.: *Model performance during training showing both training and validation metrics.*

bottom lines are thick while the middle line is thin. This consistency is required throughout the report. Always label and cross reference all tables in the document.

Table E.1.: *Experimental parameters and their values.*

Parameter	**Value**	**Unit**
Population Size	100	individuals
Generations	500	iterations
Mutation Rate	0.05	probability

```
\begin{table}[htbp]
    \centering
    \caption{Experimental parameters and their values.}
    \label{tab:parameters}
    \begin{tabular}{lcc}
        \toprule
        \textbf{Parameter} & \textbf{Value} & \textbf{Unit} \\
        \midrule
        Population Size    & 100            & individuals   \\
        Generations        & 500            & iterations    \\
        Mutation Rate      & 0.05           & probability   \\
        \bottomrule
    \end{tabular}
\end{table}
```

```
Reference using: \cref{tab:parameters}
```

Key points:

- Caption appears **above** table (IEEE standard)
- Use `\toprule`, `\midrule`, `\bottomrule` only
- Column alignment: `l` (left), `c` (center), `r` (right)
- Bold headers: `\textbf{Header}`
- Never use `|` for vertical lines

Code and Pseudocode in Academic Reports

IEEE Standard for Formal Reports

Important Note: For formal academic reports following IEEE standards, pseudocode using the `algorithmicx` package is **strongly preferred** over direct code embedding.

Why Pseudocode is Preferred

- **Language-agnostic**: Focuses on logic rather than implementation details
- **Professional presentation**: Consistent with academic publication standards
- **Better readability**: Semantic structure with clear control flow
- **Space efficient**: Avoids lengthy implementation details
- **IEEE compliant**: Follows academic formatting conventions

When Direct Code May Be Acceptable

Direct code listings are generally **discouraged** in formal academic reports. However, they may be appropriate in:

- **Appendices**: For reference or reproducibility
- **Technical documentation**: Implementation guides or manuals
- **Software-focused papers**: When code is the primary contribution
- **API demonstrations**: Showing specific library usage

Inline Code References

For mentioning function names, variables, or commands in text:

```
Use the \texttt{calculate\_fitness()} function to evaluate solutions.
Set the \texttt{population\_size} parameter to 100.
```

Rendered: Use the `calculate_fitness()` function to evaluate solutions. Set the `population_size` parameter to 100.

Algorithms and Pseudocode

Algorithm Environment

For formal algorithms, use `algorithmicx` with `algpseudocode`:

```
\begin{algorithm}[htbp]
\caption{Genetic Algorithm for Optimization}
\label{alg:genetic}
\begin{algorithmic}[1]
\Require Population size $N$, generations $G$
\Ensure Optimized solution $\mathbf{x}^*$
\State Initialize population $P \gets \text{random}(N)$
\For{$g = 1$ to $G$}
    \State Evaluate fitness for all individuals in $P$
    \State $P' \gets \text{selection}(P)$
    \State $P'' \gets \text{crossover}(P')$
    \State $P \gets \text{mutation}(P'')$
\EndFor
\State \Return best individual from $P$
\end{algorithmic}
\end{algorithm}

Reference using: \cref{alg:genetic}
```

Rendered output:

Algorithm 3 Genetic Algorithm for Optimization

Require: Population size N, generations G
Ensure: Optimized solution $\mathbf{x}^*$
1: Initialize population $P \leftarrow \text{random}(N)$
2: **for** $g = 1$ to G **do**
3: Evaluate fitness for all individuals in P
4: $P' \leftarrow \text{selection}(P)$
5: $P'' \leftarrow \text{crossover}(P')$
6: $P \leftarrow \text{mutation}(P'')$
7: **end for**
8: **return** best individual from P

Key commands:

- `\State`: Single statement
- `\If...\EndIf`: Conditional blocks
- `\For...\EndFor`: Loops
- `\While...\EndWhile`: While loops
- `\Require`: Input requirements
- `\Ensure`: Output guarantee
- `\Return`: Return statement

Citations and References

Adding References

Add entries to `references.bib` in BibTeX format:

```
@article{smith2020machine,
    author  = {Smith, John and Doe, Jane},
    title   = {Machine Learning for Optimization},
    journal = {IEEE Transactions on Neural Networks},
    year    = {2020},
    volume  = {31},
    number  = {5},
    pages   = {1234--1245}
}
```

Citing References

Use `\cite` command:

```
Machine learning has shown promising results \cite{smith2020machine}.
Multiple citations can be combined \cite{ref1,ref2,ref3}.
```

Rendered: Citations appear as numbers in square brackets: [1], [2–5].

Acronyms and Glossaries

Defining Acronyms

Add definitions to `src/frontmatter/abbreviations.tex`:

```
\newacronym{ml}{ML}{Machine Learning}
\newacronym{ai}{AI}{Artificial Intelligence}
\newacronym{nn}{NN}{Neural Network}
```

Using Acronyms

```
\gls{ml}      - First use: Machine Learning (ML)
              - Later uses: ML
\glspl{nn}    - Plural form: NNs
\Gls{ai}      - Capitalized: Artificial Intelligence (AI)
\acrshort{ml} - Always shows: ML
\acrlong{ml}  - Always shows: Machine Learning
```

Benefits:

- Automatic expansion on first use
- Consistent abbreviation usage throughout document
- Automatic generation of abbreviations list
- Prevents undefined acronym errors

Cross-Referencing

Modern Cross-References with Cleveref

This template uses `cleveref` package for intelligent cross-referencing:

```
\cref{fig:arch}        → Fig. 1.1
\cref{tab:results}     → Table 2.3
\cref{eq:fitness}      → Eq. (3.4)
\cref{alg:nsga}        → Algorithm 4.1
\cref{sec:method}      → Section 5.2
\cref{chap:intro}      → Chapter 1

Multiple references:
\cref{fig:a,fig:b,fig:c}    → Fig. 1.1, 1.2, and 1.3
\cref{tab:x,eq:y}           → Table 2.1 and Eq. (3.5)
```

Advantages over manual referencing:

- Automatic prefix (Fig., Table, Eq., etc.)
- Automatic number updating when content changes
- Intelligent grouping of multiple references
- Hyperlinks to referenced elements (in PDF)

Special Characters and Typography

Dashes

LaTeXdistinguishes between three types of dashes:

```
Hyphen:     state-of-the-art (single -)
En-dash:    pages 10--20 (double --)
Em-dash:    results---as shown---improved (triple ---)
```

Rendered: state-of-the-art, pages 10–20, results—as shown—improved.

Quotation Marks

Use proper typographic quotes:

```
``double quotes''
`single quotes'
```

Rendered: “double quotes” and ‘single quotes’.

Never use straight quotes copied from Word or text editors.

Special Symbols

Common special characters require backslashes:

```
\%    - Percent sign
\$    - Dollar sign
\&    - Ampersand
\_    - Underscore
\#    - Hash/pound
\{    - Left brace
\}    - Right brace
\~{}  - Tilde
```

Best Practices and Workflow

File Organization

- **One chapter per file**: Keep chapters in separate files for easier management
- **Descriptive filenames**: Use names like `methodology.tex`, `results.tex`
- **Image organization**: Store images in `src/images/` with subdirectories if needed
- **Version control**: Use Git to track changes and collaborate

Compilation Tips

- **Clean auxiliary files**: Regularly delete `.aux`, `.log`, `.toc` files if errors persist
- **Full recompile**: Run complete sequence (pdflatex → bibtex → makeglossaries → pdflatex × 2)
- **Check errors**: Read error messages carefully—they indicate file and line number
- **Incremental compilation**: Compile frequently to catch errors early

Common Mistakes to Avoid

- **UTF-8 characters**: Avoid smart quotes, em-dashes, and special characters copied from Word
- **Unclosed braces**: Every `{` must have matching `}`
- **Missing labels**: Always add `\label` to figures, tables, equations, and algorithms
- **Unreferenced elements**: Every figure, table, and equation must be referenced in text
- **Manual numbering**: Never type "Fig. 1" manually—use `\cref`
- **Vertical lines in tables**: Use `booktabs` rules, not `|` or `\hline`

Quality Checklist

Before final submission, verify:

- ☐ Document compiles without errors or warnings
- ☐ All figures, tables, and equations are referenced in text
- ☐ All citations appear in bibliography
- ☐ Abbreviations are defined and used consistently
- ☐ Cross-references show numbers (no "??")
- ☐ Table of contents has correct page numbers
- ☐ All code listings have captions and proper formatting
- ☐ Mathematical notation follows IEEE conventions
- ☐ No UTF-8 encoding errors in bibliography
- ☐ PDF bookmarks and hyperlinks work correctly

Advanced Features

Custom Commands

The template provides custom commands for convenience:

```
\vect{v}      - Bold vector notation
```

```
\matr{A}      - Bold matrix notation
\set{S}       - Calligraphic set notation
\ie           - i.e.,
\eg           - e.g.,
\etc          - etc.
```

Units with siunitx

For proper unit formatting:

```
\SI{5}{\meter}
\SI{100}{\kilo\gram}
\SI{3.5}{\giga\hertz}
\SIrange{10}{20}{\celsius}
```

Rendered: 5 m, 100 kg, 3.5 GHz, 10–20°C.

Getting Help

Resources

- **This document**: Read all guideline chapters thoroughly
- **LaTeX Stack Exchange**: `tex.stackexchange.com` for specific questions
- **Overleaf Documentation**: Comprehensive LaTeXtutorials
- **Template class file**: See `thapathaliece.cls` for implementation details

Common Issues

Bibliography not appearing Run: pdflatex → bibtex → pdflatex × 2

Acronyms not expanding Run: pdflatex → makeglossaries → pdflatex × 2

Undefined references (??) Compile multiple times until stable

Missing images Check file paths and extensions (.png, .png, .jpg)

UTF-8 errors Replace special characters in .bib file with LaTeXequivalents

WRITING GUIDELINES

This section provides comprehensive guidance on technical writing following IEEE standards and best practices for academic publications [**?**, **?**].

Avoiding Common Mistakes

Grammar and Style

- **Avoid**: "The data was analyzed" → **Use**: "The data were analyzed" (data is plural)
- **Avoid**: "very unique", "more optimal" → **Use**: "unique", "optimal" (absolute adjectives)
- **Avoid**: "In this paper" → **Use**: "In this report" or "In this work"
- **Avoid**: Contractions (don't, can't) → **Use**: Full forms (do not, cannot)

Technical Writing

- Define terms before using them extensively
- Maintain consistency in terminology throughout
- Use specific numbers rather than vague terms: "15% improvement" not "significant improvement"
- Explain acronyms at first use, even if well-known in the field
- Cross-reference related sections appropriately using `\cref{}` commands

COMPILING AND TROUBLESHOOTING

This section provides practical guidance on compiling your report and resolving common issues.

Compilation Process

Standard Compilation Sequence

To properly compile this report with all references, use this sequence:

1. **pdfLaTeX**: First pass to generate auxiliary files
2. **bibtex**: Process bibliography references
3. **makeglossaries**: Generate abbreviations list
4. **pdfLaTeX**: Second pass to include bibliography
5. **pdfLaTeX**: Third pass to resolve all cross-references

Command Line Compilation

Open terminal in report directory and run:

```
pdflatex main.tex
bibtex main
makeglossaries main
```

```
pdflatex main.tex
pdflatex main.tex
```

Using LaTeXEditors

Most LaTeXeditors have "Build" buttons that handle the sequence automatically:

- **TeXstudio**: Press F5 or click "Build & View"
- **Overleaf**: Compiles automatically on save
- **VS Code**: With LaTeXWorkshop extension, use "Build LaTeXproject"

Common Compilation Errors

Missing Package Errors

Error: `! LaTeXError: File 'package.sty' not found`
Solution: Install the missing package:

- TeX Live: `tlmgr install package`
- MiKTeX: Will prompt to install automatically
- Or install full LaTeXdistribution

Undefined Reference Warnings

Warning: `LaTeXWarning: Reference 'fig:example' undefined`
Solution: Run pdfLaTeXmultiple times (2–3) after adding new labels. References require multiple passes to resolve.

Citation Undefined

Warning: `Citation 'smith2020' undefined`
Solution:

- Check that citation key exists in `references.bib`
- Run bibtex step
- Run pdflatex again

Glossary Not Appearing

Problem: Abbreviations list is empty or doesn't appear
Solution:

- Ensure you've used `\gls{}` commands in text
- Run `makeglossaries` command

- Run pdfLaTeXagain
- Glossaries only shows acronyms actually used in document

Font Errors

Error: Font shape warnings or missing font files
Solution:

- Ensure newtx package is installed: `tlmgr install newtx`
- Update LaTeXdistribution to latest version
- Run `updmap` or `updmap-sys` to refresh font database

File Organization Issues

File Not Found Errors

Error: `! LaTeXError: File 'src/chapters/intro.tex' not found`
Solution:

- Verify file exists in correct directory
- Check for typos in filename or path
- Use forward slashes (/) not backslashes (\) in paths
- Ensure no special characters in filenames

Image Not Found

Error: `! Package pdftex.def Error: File 'img/figure.png' not found`
Solution:

- Check that image file exists in specified path
- Verify file extension is correct (.pdf, .png, .jpg)
- Use relative paths from main .tex file location
- Ensure image is in supported format

Managing Large Documents

Compilation Time

For large theses with many figures:

- Use `\includeonly{chap/chapter_name}` to compile only specific chapters during editing
- Keep this in main .tex file, comment out when not needed
- Final compilation should include all chapters

Example:

```
% Uncomment to compile only specific chapters:
% \includeonly{chap/methodology,chap/results}
```

Draft Mode

For faster compilation during editing:

```
\documentclass[draft]{scrreprt}
```

Draft mode:

- Shows boxes instead of images
- Marks overfull boxes clearly
- Compiles faster
- Remember to remove `draft` option for final version

Quality Checks

Before Final Submission

Run these checks:

1. **Complete compilation**: Full pdfLaTeX→ bibtex → makeglossaries → pdfLaTeX× 2
2. **No warnings**: Address all LaTeXwarnings in log
3. **References check**: All citations appear in bibliography
4. **Cross-references**: No "??" in document
5. **Figures**: All figures appear correctly, not missing
6. **Page numbers**: TOC page numbers match actual pages
7. **Spelling**: Run spell checker
8. **Formatting**: Consistent throughout document

PDF Quality

Verify final PDF:

- **File size**: Reasonable (< 50 MB typically)
- **Hyperlinks**: Internal links work correctly
- **Bookmarks**: Chapter bookmarks appear in PDF viewer
- **Fonts**: All fonts embedded properly
- **Images**: Clear and readable at 100% zoom

Best wishes for your writing,
Krishna Acharya,

REFERENCES

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[2] J. A. Smith, “Optimization methods in modern computing,” *IEEE Trans. Comput.*, vol. 69, no. 4, pp. 523–538, 2020.

[3] IEEE, “Ieee author center,” https://ieeeauthorcenter.ieee.org/, 2022, accessed: 2025-12-15.

[4] ——, *IEEE Editorial Style Manual*, Institute of Electrical and Electronics Engineers, 2021, available: https://www.ieee.org/content/dam/ieee-org/ieee/web/org/conferences/style_references_manual.pdf.

[5] M. Alley, *The Craft of Scientific Writing*, 4th ed. New York, NY: Springer, 2018.