Assignment template using IEEEtran.cls for IEEE Journals and Transactions

FirstName LastName
Department of Computer Science
UiT The Arctic University of Norway
Tromsø, Norway
aabbb1234@uit.no

FirstName LastName
Department of Computer Science
UiT The Arctic University of Norway
Tromsø, Norway
aabbb1234@uit.no
userName@github

FirstName LastName
Department of Computer Science
UiT The Arctic University of Norway
Tromsø, Norway
aabbb1234@uit.no
userName@github

***Abstract*—This document describes the most common article elements and how to use the IEEEtran class with LaTeX to produce files suitable for submission to the IEEE. IEEEtran can produce conference, journal, and technical note (correspondence) papers with suitable class options. This template is meant as a general guideline on how to write a report, and to give some tips about what you should and should not be writing. You may move or cut sections depending on the assignment or your needs.**

While writing this report template, I enjoyed reading Michael Alley's 'The Craft of Scientific Writing'. I recommend having it by your side when you are stuck writing- it happens to all of us [1].

Make sure you clean up the template text before submitting your report. ;)

***Index Terms*—Assignment submission, LaTeX, paper, template, typesetting.**

I. Introduction

This section should be brief. Describe the assignment and the requirements in your own words. Avoid listing the requirements directly.

There are many opinions on first person speaking when writing a technical report. In general:

- 1. First-person report is good at:
  - Expressing individual work, personal opinions and ideas.
  - Creates an informal and personal tone.
  - Reduces cluttering of your writing
- 2. An objective report is good at:
  - Emphasizing your work

Mixing these in your reports is not uncommon, and people like Einstein, Feynman, and Curie frequently used both forms in their texts.

Here are some examples of how to start your introduction:

1) This report describes the design and implementation of a list ADT using a linked list. It will go into detail about the design choices made and discuss the benefits and trade offs of those choices.
2) Boids is a computer model created by Craig Reynolds that simulates the flocking behavior of birds [2]. In this report, we present an implementation of the model using the Python programming language.
3) SQL is a widely used querying language used to process queries into table-based databases. This text details the implementation of a simplified server that implements a subset of the SQL language built over sqlite.

A. Outline

The rest of this paper is organized as follows:

Section II outlines concepts and background information relevant to the rest of the paper.

Section III is a high-level description of your solution to the assignment.

Section IV goes into a detailed explanation of your implementation that you described in Section III.

Section V discusses the methodology for your experiments and includes results from your experiments.

Section VI is the most important section in a report; this is where you show that you understand the theory behind your solution and can reflect on it with the choices you made.

Section VII concludes and summarizing possible future work.

II. Theoretical Background

This section is where you provide information on the theoretical aspects of your design. You can usually assume that the theory required to solve the assignment is known to the reader, but if you want to clarify terms or go into detail about specific points in the theory (if you are doing something slightly different, or a detail of it is of notable importance to your implementation), consider writing a few words about it here.

We want you to write a short section explaining the most important background information required to read and understand the rest of the report.

Here is an example of a subsection covering a topic.

A. Virtual memory

The basic concept of virtual memory is that you map the virtual address the processes use to a unique physical address in physical memory. This means that two processes can access the same virtual address in their address space but get two different results since the addresses point to different places in the physical memory. This again means that each process can use all of its 32-bit address space while still ensuring that no other processes can access its data [2]

Using figures in technical backgrounds is encouraged, if that makes the concept easier to explain. Usually, you want figures/images as Scaleable Vector Graphics(SVG) or Portable Document Format(PDF), especially for your graphs. Sometimes that is not doable, and you can use portable network graphics(PNG) or similar. The following snippet shows how to import figures.

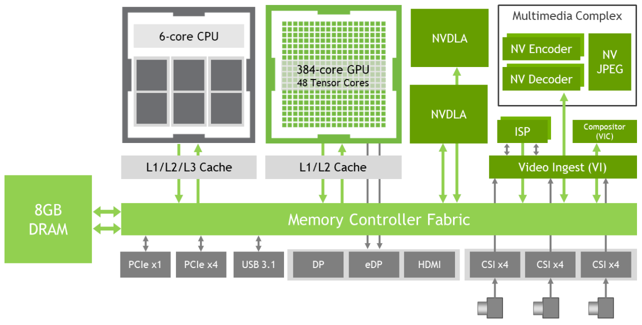

Fig. 1. Block diagram of the Jetson Xavier NX

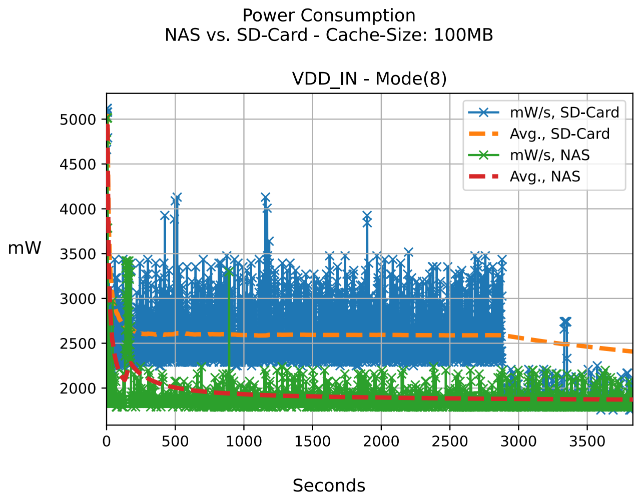

Fig. 2. Total power consumption compared between NAS and SD-Card
Note the NAS-experiment did not complete in time, and the measurements for the NAS fit to the SD-card measurements

III. Design

This is where you describe how you solved the assignment, at least on paper. Give a high-level view of your design. As a rule of thumb, if you find yourself describing code, you need to go to a higher abstraction level. If you for some reson want to write code you can use the ‘listings’ package as seen in Figure 3

This section is also a good place to put illustrations to enhance the text. There are multiple tools out there to create good illustrations. draw.io is a strong tool that can be run in a browser. There are stronger, free tools, such as Yed Graph Editor, for more advanced users.

```c
/* Simple copy from src to dest */
char *strcpy(char *restrict dest,
        const char *restrict src)
{
    char *destorig = dest;
    for (;; dest++, src++) {
        char copiedchar = *dest = *src;
        if (!copiedchar) break;
    }
    printf("Strings are orange\n");
    return destorig;
}
```

Fig. 3. An implementation of strcpy in C

While illustrations can help make your report clearer and look more polished, avoid using them to fill up space if you can convey the same information clearly using just text.

Examples of what the design section should cover:

- The interface of the list ADT supports six methods. These are create_list(), destroy_list(), add_list(), remove_list(), iterate_list() and sort_list(). When a list is created, it is provided with a comparator method that is used to handle sorting…
- The Boids simulation consists of a set of entities called Boids, Each boid moves independently according to a set of criteria, specified in three rules. Firstly, boids avoid crashing into obstacles, including other boids. Secondly, boids attempt to maintain the same speed and heading as nearby boids. Finally, all boids attempt to move closer to each other to form a cohesive flock.
- The server parses incoming data requests into an SQL query and runs them on its database. The result is then processed into JSON and returned to the client.

Remember to avoid low-level details! An expert should in theory be able to implement your design in any programming language based on what you write in this section.

IV. Implementation

This is where you go into detail about your specific implementation. Questions you should answer here are things such as “How does your implementation match your design?” and “Are there any bugs, and do you have any ideas about what may be causing them?”. What sort of difficulties did you experience when working, and how did you overcome them? If you found a clever solution to the problem, this is also the place to write about that.

A. Technical Details

You may want to include a short section giving high-level details about your implementation, such as the programming language used and other information you find relevant for your report. In most cases however, this section is unnecessary, as the assignment usually decides those details for you. Even if you have the freedom of choice, consider whether this information is really relevant to the report, which should avoid low-level implementation details most of the time.

V. Experiments and Results

A core pillar of computer science is testing. In this section, you should include your methodology for testing your implementation. How do you know your implementation meets the requirements? What sort of performance metrics have you chosen to benchmark your solution, and how did you go about performing tests to gather those metrics?

You should present the results of your tests here, either using an illustration and/or a table of results. These will be valuable in the discussion section. The following is an example for how to format a table of results in LaTeX.

	Precision					
Classifier	(1)	(2)	(3)	(1&2)	(1&3)	All
Perceptron	0.78	0.82	0.24	0.81	0.77	0.83
Decision Tree	0.65	0.79	0.56	0.75	0.65	0.73
One-Class SVM	0.74	0.72	0.50	0.80	0.73	0.85
Isolation Forest	0.54	0.51	0.52	0.53	0.54	0.53

TABLE I
PRECISION RESULTS OF CLASSIFIERS FOR DIFFERENT FEATURE SETS

If presenting data and benchmarks are not important to the assignment, you may likely find yourself cutting this section. If you decide to do so, make sure you at least mention your correctness criteria and testing methodology somewhere else.

VI. Discussion

The discussion section is the most important section in a report, This is where you show that you understand the theory behind the solution, and also a chance to argue the pros and cons of your solution. You should discuss about the results of your measurements and why you think they are the way they are. Also bring up tradeoffs, and why you made the choices you did; show that you understand the alternatives, and why they may be a good idea (or not) for the particular problem the assignment asked you to solve.

Here is an example of a discussion subsection:

A. Recovery of a simulated crash

When a node recovers from a simulated crash, it will check if it's neighbors is still connected to it. If not, it will try and start an internal join to its previous successor. This works as long as the previous successor is still active in the network. The case where the previous successor is not active, is not dealt with, and will result in the node not being able to recover.

VII. Conclusion

Here you sum up the report and reiterate the results. Does not need to be very long, a few sentences is fine.

References

[1] Michael Alley. *The Craft of Scientific Writing*. eng. Fourth edition. New York, NY: Springer New York, 2018. ISBN: 1441982876.

[2] Andrew S. Tanenbaum. *Modern operating systems*. eng. Fifth edition. Hoboken: Pearson, 2024. ISBN: 9781292459660.

Appendix
Proof of the Zonklar Equations

Use `\appendix` if you have a single appendix: Do not use `\section` anymore after `\appendix`, only `\section*`. If you have multiple appendixes use `\appendices` then use `\section` to start each appendix. You must declare a `\section` before using any `\subsection` or using `\label` (`\appendices` by itself starts a section numbered zero.)