Using LaTeX for High Quality Project Report Generation

A PROJECT REPORT

Submitted in partial fulfilment for the award of the degree of

MS

in

Sofware Engineering

By

Santhos Baala RS, 09MSE038

Under the Guidance of

Dr. Krishna Chandramouli

Associate Professor

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

VIT UNIVERSITY

VIT

UNIVERSITY

(Estd. u/s 3 of UGC Act 1956)

Vellore - 632 014, Tamil Nadu, India

DECLARATION BY THE CANDIDATE

I here by declare that the project report titled, **"Using LaTeX for High Quality Project Report Generation"** submitted by me to VIT University, Vellore in partial fulfilment of the requirement for the award of the degree of **M.S. (Software Engineering)** is a record of bonafide project work carried out by me under the guidance of **Dr. Krishna Chandramouli (Associate Professor)**. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore Signature of the Candidate

Date:

Santhos Baala RS (09MSE038)

VIT

UNIVERSITY

(Estd. u/s 3 of UGC Act 1956)

Vellore - 632 014, Tamil Nadu, India

SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING [SITE]

SOFTWARE ENGINEERING DIVISION

BONAFIDE CERTIFICATE

This is to certify that the project report titled, **"Using LaTeX for High Quality Project Report Generation"**, submitted by **Santhos Baala RS (09MSE038)** to VIT University, Vellore in partial fulfillment of the requirement for the award of the degree of M.S. (Software Engineering) is a record of bonafide work carried out by him/her under my guidance. The project fulfills the requirements as per the regulations of this institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Dr. Krishna Chandramouli

Internal Guide

Associate Professor,

School Of Information Technology & Engineering

Internal Examiner **External Examiner**

Acknowledgements

I take immense pleasure in thanking **Dr. G. Viswanathan**, my beloved chancellor, VIT University, **Dr. Sankaran**, dean, School Of Information Technology and Engineering, **Prof. Jayaram Reddy A**, Program Manager, M.S. Software Engineering and **Dr. Krishna Chandramouli**, project coordinator, M.S. Software Engineering for having permitted me to carry out the project.

I express gratitude to my guide, **Dr. Krishna Chandramouli**, for guidance and suggestions that helped me to complete the project on time. Words are inadequate to express my gratitude to the faculty and staff members who encouraged and supported me during the project. Finally, I would like to thank my ever-loving parents for their blessings and my friends for their timely help and support.

Executive Summary

This document describes how to use the vitmsprojectreport class with LaTeX to produce high quality typeset project report that is suitable for submission to the School of Information Technology and Engineering (SITE). The class can further be extended to various courses and department by modifying the title page and department information.

Table of Contents

List of Figures

List of Tables

1. Introduction

1.1 About the Template

With a basic understanding of the LaTeX language and say 10 to 20 commands, an author can produce beautiful typeset project report quickly, with minimal effort. The purpose of this document is to serve as a user guide for the project report template and document its special behaviours. Examples have also been provided for common tasks such as insertion of images, table, equation, etc., that can be copied as is by the users. It is assumed that the user has a basic knowledge on working of the LaTeX system. Those lacking are strongly urged to lookup some excellent literature through [2].

Sufficient examples have been given in the document and as users request solutions for specific problems that they encounter, more will be added. However, an understanding of the LaTeX system will help the user in unexplainable ways[1]. The main advantage of using the custom class is that, the user need not worry about the final layout. The class may be changed during the development of the project report, the user needs to just place the latest version of the class file, without touching the content[2]. LaTeX, along with the custom class file gives an incredible expressive power its users so that they can focus on the semantics of the document.

1.2 Generic LaTeX Commands

LaTeX contains many commands and with it comes numerous built-in packages that add many features and provide options to customise the output. However, the user is advised to stick to the basic commands, illustrated in this document so that unexpected or incorrect output can be avoided. If, however a specific feature is requested it shall be considered and incorporated into the template. The class file for project report has been extended from the standard *report* class, supplied by LaTeX. Therefore, whatever applied for report also applies for the custom class file.

---

[1] The LaTeX system is a vast ocean. That said, you can mostly get away with copy-paste skills. You have to observe the source code of the examples and learn

[2] It is always recommended to get the latest class file from the repository and compile before project report submission.

1.3 Generating the Final Document

The template development is an ongoing process. It is mature at this point from the point of view of semantics. Layout or default contents (e.g bonafide, acknowledgement, etc.,) may change at a later point in time. Therefore, before generating the final document, please make sure to check out a frozen version of the class file from the repository before compiling the class file.

1.4 Asking Questions

You can post your questions in the forum or if you encounter any problem related to unexpected/incorrect output due to the template send an e-mail to the contributors at the github page.

1.5 Contributing

The project is hosted on GitHub [3] and uses the GIT repository management system. Contributors can fork the project and send pull requests to the repository for the changes to be merged. The patch will be evaluated and if found to be good, merged into the master branch of the repository. If however, a separate template is required for other departments and courses, the project can be forked and maintained separately. Issues, pertaining to the template, such as rendering faults, can be posted in the issues section of the repo. The contributors shall also put up a list of items that need improvement or new features in progress that people can contribute to.

2. Auto-Generated Pages

The parts of the document, unique to the project report are generated by a predefined template, using simple commands. The order in which the commands are issued determine the corresponding order of these individual pages. The arguments that need to be passed to these commands are explained in the following sections. The sequence of commands to generate such pages have already been placed in the example document and it is recommended not to touch those, expect when inserting your project specific information.

2.1 The Title Page

The user can generate the title page using the `\maketitlepage` command. The detailed syntax is given in Figure 2.1.

```
\maketitlepage
{Project Title}
{Name}
{RegNo}
{Guide Name}
{Guide Title}
```

Figure 2.1: The syntax of \maketitlepage command

Note that you should always set the title page to be the first page of the document, so issue this command right after the document begins. The template also collects information like your name and regno when you declare the title page so that it can be auto-inserted into other pages like declaration and bonafide. The following subsections d

2.1.1 Project Title

Your project title should be placed here. If your title is long and you are not satisfied with the way the lines break, use \\ to insert line breaks at suitable locations. Note the inclusion of blank space after the backslashes.

E.g. `{A very very\\ long long title}`.

2.1.2 Regno

Supply your register number here. A long name will automatically push the register number to the next line. However, to be safe, prepend your RegNo with \\.
E.g. `{\\09MSE038}`.

2.1.3 Guide Name

Your guide's name goes here. Make sure to add your guide's title too, Prof., Dr., etc. Also, be sure to add a non-breaking space "~" after the title so that unintended line breaks are prevented. Follow the regular convention of first name, last name.
E.g. `{Prof.~Robert}` or `{Dr.~John}`.

2.1.4 Guide Title

Your Guide's title(designation) goes here. Do not abbreviate anything.
E.g `{Assistant Professor (Senior)}`.

2.2 Declartion Page

Use `\makedeclarationpage` command, rest is automatically generated!

2.3 Bonafide Page

Use `\makebonafidepage` command, rest is automatically generated!

2.4 Acknowledgement Page

Use `\makeackpage` command to generate the page. The generated text is more than sufficient for acknowledgement. However, if there is a request for custom writeup for the acknowledgement page, we'll consider that[1]. There are three arguments to this page as given in Figure 2.2. Make sure to prepend the names with appropriate titles, Mr, Dr, etc., and remember to give a non-breaking space character just after the title as you did for your guide's name in 2.1.3

[1] We don't want to kill creativity!

```
\makeackpage
{Dean Name}
{Program Manager Name}
{Year Co-Ordinator Name}
```

Figure 2.2: The syntax of \makeackpage command.

2.5 Executive Summary Page

The command `\makeexecsummarypage` creates the executive summary page. The only argument to this function is your executive summary. Keep it down to 1-2 paragraphs. An example is given in Figure 2.3.

```
\makeexecsummarypage{
This document describes how to use the vitmsprojectreport
class with \LaTeX\ to produce high quality typeset project
report that is suitable for submission to the School of
Information Technology and Engineering (SITE). The class
can further be extended to various courses and department
by modifying the title page and department information.
}
```

Figure 2.3: Example usage of the \makeexecsummarypage command.

2.6 Single Command to Generate All the Starting Pages

The shortcut `\makestartingpages`, renders all the starting pages. The format of the parameters that applied to individual commands mentioned previously apply here to.

```
\makestartingpages
{Project Title}
{Name}{Regno}
{Guide Name}{Guide Title}
{Dean Name}
{Program Manager Name}
{Year Co-Ordinator Name}
{Executive Summary Contents}
```

Figure 2.4: Single command to render all the starting pages.

3. Text Elements

The chapters, sections and subsections in the document are created with the standard commands `\chapter`, `\section` and `\subsection` respectively. The text sections are automatically numbered using arabic numerals and also added to the table of contents. You can create a deeper level using `\subsubsection`. It is recommended to keep the depth restricted to the subsection level. The template doesn't guarantee proper output for beyond that. The following sections illustrate on creating various text structures in the document and cross-references.

3.1 Chapters, Sections and Sub-Sections

Chapters are created with the `\chapter` command. The chapter title is passed as an argument. Each chapter is automatically placed in a separate page. Sections and Sub-Sections under a chapter are created with the `\section` and `\subsection` respectively. The syntax of all the commands are given in Figure 3.1.

```
\chapter{The Chapter Title}
\section{The Section Title}
\subsection{The Sub-Section Title}
```

Figure 3.1: Commands to create text structures.

If you want to place a section inside a chapter, place it under that chapter. Similarly, a sub-section meant to be placed inside a section is placed under the corresponding section. That's how levels are created. The sequence of commands in Figure 3.1 will create a sub-section, under a section, which in turn is under a chapter.

3.2 Lists

List is inherited directly from vanilla LaTeX list. You can look them up from any tutorial for LaTeXlists, preferably from [2]. We give a short example of an unordered list and an ordered list over here. Changing the list style, bullet shape, etc., is your choice. *Note that an empty list(list with no items) will throw an error during compilation.* More information on the list structure is available at http://en.wikibooks.org/wiki/LaTeX/List_Structures#Enumerate.

3.2.1 Unordered

An unordered list is generated by the `\begin{itemize}...\end{itemize}` environment with `\item` commands inside. The listing in Figure 3.2 generates an unordered list, placed to the right of the same figure.

```
\begin{itemize}
\item C
\item C++
\item Java
\end{itemize}
```

- C
- C++
- Java

Figure 3.2: Unordered list example.

3.2.2 Ordered

Ordered lists can be generated by the `\begin{enumeration}...\end{enumeration}` environment with `\item` commands inside. The listing in Figure 3.3 generates an orders list, placed to the right of the same figure.

```
\begin{enumerate}
\item C
\item C++
\item Java
\end{enumerate}
```

1. C
2. C++
3. Java

Figure 3.3: Ordered list example.

The default numbering scheme of the template is arabic and it is recommended not to change that. The numbering scheme for nested ordered lists is automatically managed.

3.2.3 Description List

Sometimes, it is required to render something like a definition list. Although semantically it isn't a list, it follows the same syntax structure. Such special type of lists can be created with the `\begin{description}...\end{description}` environment and the `\item[...]` command. The usage of one such special list is illustrated in Figure 3.4 along with the output.

```
\begin{description}
\item[C] The first item's description
\item[C++] The second item's description
\item[Java] The third item's description
\end{description}
```

C The first item's description

C++ The second item's description

Java The third item's description

Figure 3.4: Description list example.

3.3 Footnotes

A footnote is an ancillary piece of information, something that is additional but less important. They are placed at the bottom of the page[1]. The template automatically adds a horizontal rule to separate it from the main flow of the document. Each footnote is numbered, specific to the chapter in which it resides. A footnote is added through the `\footnote` command, with the actual contents of the footnote as the argument. Footnotes are meant to be added along the regular flow of the document. It is illustrated by Figure 3.5.

```
This is the regular flow of document\footnote{This piece of
text is treated as a footnote}.
```

Figure 3.5: An example footnote.

There may be multiple pages in the same page. The placement of the footnote, flow into the next page, numbering, etc., are all automatically managed.

3.4 Cross-References

Other elements in the document can be referenced via the `\ref` command at the location where a reference is to be inserted and the target identifies itself with a `\label` command.

[1] This is a sample footnote

3.5 External References

TODO

4. Floating Structures

Elements that are embedded in a text document are termed as floating structures. These include figures, tables, equations, algorithms, etc. In the following sections, examples have been provided on how to use these elements and how the template behaviour can be controlled through the various options available.

4.1 Figures

Figures are handled in the standard LaTeX manner. An minimalistic example is given in Figure 4.1. The figure is placed inside the `\begin{figure}...\end{figure}` environment. The options passed to the environment is the figure placement priority preference, h - at the current location, t - top and b - bottom. `\centering` indicates that the figure and caption should be placed at the centre. `\includegraphics` is self explanatory, with the width option and the source file. The source file extension is optional, left blank, LaTeX chooses the best available format automatically. The rest of the section contains image inclusion and subfigure examples. The output generated is Figure 4.2.

```
\begin{figure}[htb]
\centering
\includegraphics[width=0.5\textwidth]{images/105003.jpg}
\caption{The big cat}
\label{fig:big_cat}
\end{figure}
```

Figure 4.1: Using 50% of the textbox to place the image.

4.1.1 Sub-Figures

Figures inside figures. The subfigures are suitable when you want to explain a progressive concept through images. For example, a solar eclipse, amoeba fisson, etc. Figure 4.3 partially lists the code for rendering subfigures[1], Figure 4.4 is the output.

[1] Refer the source code of this document for more

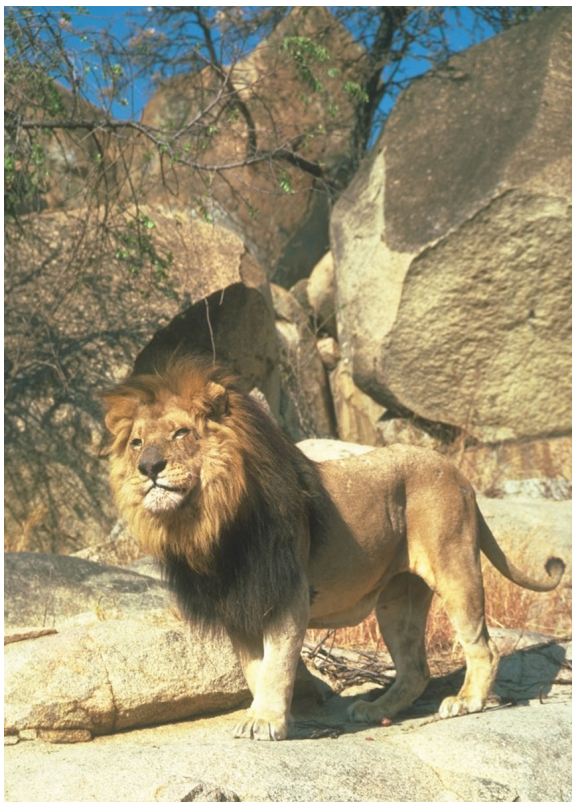


Figure 4.2: The big cat

4.2 Tables

4.3 Algorithms

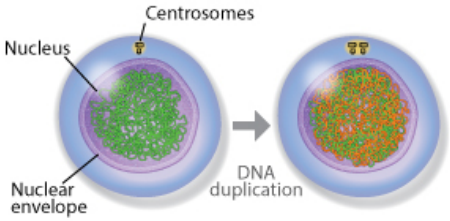
In this template, algorithms are treated as figures for the sake of simplicity. They too get listed in the list of figures. Similar to a figure, algorithms need to be placed in the `{algorithm}` environment. The actual algorithm is represented by the statements inside the `{algorithmic}` environment. It defines its own set of commands. We strongly recommend the algorithmic environment compared to other packages offering similar functionality since it has been tested on the template and has a compact vocabulary. Figure 4.5 lists the algorithmic code for generating the output in Figure 4.6. Learn more about the algorithmic commands from [1].

Another informal way of defining algorithms is to use an enumeration list, suppose you are not concerned with the specifics of your algorithm or it simply needs to be descriptive. Figure 4.8 is one such listing, generated by the sample source in Figure 4.7[2].

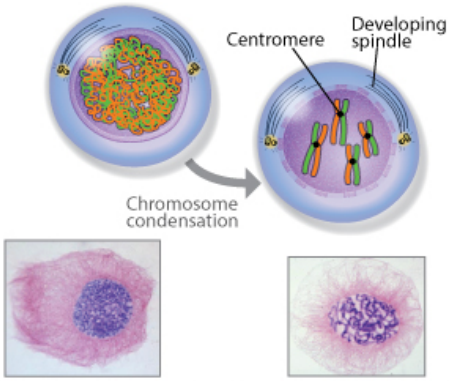
[2]Refer to the complete listing in this document's source code

```
\begin{figure}[htb]
\centering
\begin{subfigure}[b]{0.4\textwidth}
\includegraphics[width=\textwidth]{images/interphase.png}
\caption{Interphase}
\label{fig:mitosis_interphase}
\end{subfigure}
% Spacing %
\hspace{0.1\textwidth}
% Spacing %
\begin{subfigure}[b]{0.4\textwidth}
\includegraphics[width=\textwidth]{images/prophase.png}
\caption{Prophase}
\label{fig:mitosis_prophase}
\end{subfigure}
% Spacing %
\linebreak\linebreak
% Spacing %
\begin{subfigure}[b]{0.4\textwidth}
\includegraphics[width=\textwidth]{images/prometaphase.png}
\caption{Prometaphase}
\label{fig:mitosis_prometaphase}
\end{subfigure}
% Spacing %
\hspace{0.1\textwidth}
% Spacing %
...
...
...
% End of subfigures
\caption{The steps of mitosis.}
\label{fig:mitosis}
\end{figure}
```

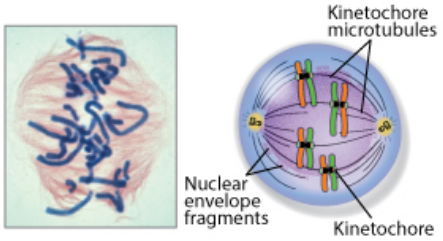
Figure 4.3: Subfigure example.



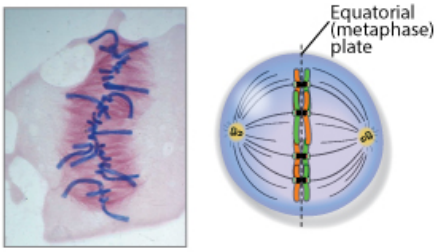
(a) Interphase



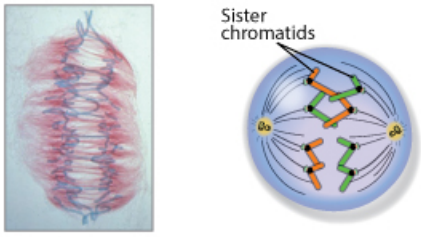
(b) Prophase



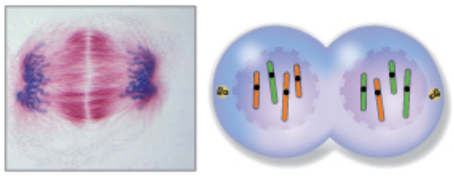
(c) Prometaphase



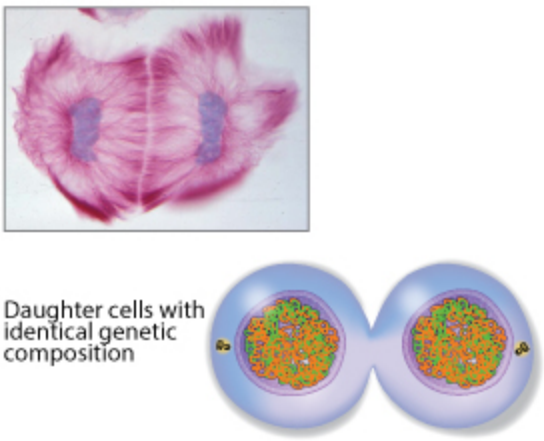
(d) Metaphase



(e) Anaphase



(f) Telophase



(g) Cytokinesis

Figure 4.4: The steps of mitosis.

```
\begin{algorithm}
\begin{algorithmic}[1]
\Procedure{Euclid}{$a,b$}\Comment{The g.c.d. of a and b}
\State $r\gets a\bmod b$
\State $a\gets b$
\State $b\gets r$
\State $r\gets a\bmod b$
\State \textbf{return} $b$\Comment{The gcd is b}
\EndProcedure
\end{algorithmic}
\caption{Euclid's Algorithm}
\label{algo:euclid_algorithm}
\end{algorithm}
```

Figure 4.5: Euclid's algorithm, written in algorithmic.

1: **procedure** EUCLID(a, b) ▷ The g.c.d. of a and b
2: $r \leftarrow a \bmod b$
3: $a \leftarrow b$
4: $b \leftarrow r$
5: $r \leftarrow a \bmod b$
6: **return** b ▷ The gcd is b
7: **end procedure**

Figure 4.6: Euclid's Algorithm

```
\begin{algorithm}
\begin{enumerate}
    \item Initialize a population array of particles with
     random positions and velocities on $d$ dimensions,
     in the problem space.
    ...
    ...
    ...
    \item Loop to 2, until the stoping criterion is satisfied.
\end{enumerate}
\caption{Steps to perform Brownian motion.}
\label{alto:brownian_motion_steps}
\end{algorithm}
```

Figure 4.7: Listing to generate an enumerated list of steps.

1. Initialize a population array of particles with random positions and velocities on d dimensions, in the problem space.
2. For each particle, evaluate the desired optimization fitness function in d variables.
3. Compare particles fitness evaluation with particles $pbest$. Then set $pbest$ value (to make it) equal to the current value, and the $pbest$ location equal to the current location in d-dimensional space.
4. Compare fitness evaluation with the populations overall previous best. If current value reduces the error to achieve global minima than previous $gbest$, then update gbest value.
5. Update the velocity and position of the particle according to equation 1 and 2.
6. Loop to 2, until the stoping criterion is satisfied.

Figure 4.8: Steps to perform Brownian motion.

5. Equations

6. References

[1] Szasz Janos. *The algorithmicx package*. Indiana State University, USA, 2005. http://cs.indstate.edu/CS695/algorithmicx.pdf.

[2] Stefan Kottwitz. *LaTeX beginner's guide*. Packt, Birmingham, UK, 2011.

[3] RS Santhos Baala. Vit ms project report class, 2014. https://github.com/santhosbaala/vitmsprojectreport.