SHORT TITLE

# YOUR THESIS TITLE, WHICH CAN BE AS LONG AS YOU WANT ON THE TITLE PAGE

BY

JANE DOE, B.Eng.

a Report
submitted to the Department You Belong To
and the School of Graduate Studies
of McMaster University
in partial fulfilment of the requirements
for the degree of
Masters of Engineering

| | |
|---|---|
| Masters of Engineering (YYYY) | McMaster University |
| (Department You Belong To) | Hamilton, Ontario, Canada |

| | |
|---|---|
| TITLE: | Your Thesis Title, Which Can Be As Long As You Want On the Title Page |
| AUTHOR: | Jane Doe<br>B.Eng. (Software Engineering & Game Design),<br>McMaster University, Hamilton, Canada |
| SUPERVISOR: | Your Supervisor |
| NUMBER OF PAGES: | xii, 12 |

# Lay Abstract

A lay abstract of not more 150 words must be included explaining the key goals and contributions of the thesis in lay terms that is accessible to the general public.

# Abstract

Abstract here (no more than 300 words)

*Your Dedication*

*Optional second line*

# Acknowledgements

Acknowledgements go here.

# Contents

# List of Figures

# List of Tables

# Notation, Definitions, and Abbreviations

## Notation

| | |
|---|---|
| $A \leq B$ | A is less than or equal to B |

## Definitions

| | |
|---|---|
| **Challenge** | With respect to video games, a challenge is a set of goals presented to the player that they are tasks with completing; challenges can test a variety of player skills, including accuracy, logical reasoning, and creative problem solving |

## Abbreviations

| | |
|---|---|
| **AI** | Artificial intelligence |

# Declaration of Academic Achievement

The student will declare his/her research contribution and, as appropriate, those of colleagues or other contributors to the contents of the thesis.

# Chapter 1

# Introduction

Jupyter Notebook can be generated from codified knowledge to improve their reusability, usability, reliability, and modifiability. In many software engineering domains, like scientific computing, knowledge is duplicate across software artifacts, which sometimes make it difficult and tedious to write and maintain them manually.

Generative programming is a technique to address this problem. It allows programmers to write the code or document at a higher abstract level, and the generator produces the desired outputs. With generative programming, we can do things more efficiently since automation increases the efficiency. Drasil is an application of generative programming, and it is the framework we used in this research. We are tyring to generate Jupyter Notebook in Drasil to extend the flexibility of Drasil and to acheive the reusability, usability, and maintainability of software artifacts.

## 1.1 Background

### 1.1.1 Drasil

Drasil is a framework that can generate software artifacts, including Software Requirement Specification (SRS), code (C++, C#, Java, and Python), README, and Makefile, from a stable knowledge base. The main goals are to reduce knowledge duplication and improve traceability. Drasil captures the knowledge through our hand-made case studies; it allows users to provide recipes of their scientific problems and generate code and documentation. Each piece of information is encoded in Drasil once and that information can be used wherever it is needed. Drasil is capable of generating SRS in document languages HTML and LaTeX. We are working on increasing our automated database and improving how we encode the recipes.

### 1.1.2 Jupyter Notebook

Jupyter Notebook is an interactive open-source web application for creating and sharing documents that contain text, executable code, equations, graphics, and visualizations. There are several advantages of Jupyter Notebook: sharable, all-in-one, and executable code and equations. First of all, the notebook is easy to share because it can be converted into other formats such as HTML, Markdown, and PDF. Another benefit is that it combines all aspects of data in one single document, making the document easier to visualize, maintain and modify. In addition, Juypyter Notebook provides an environment of executable code and equations. Programmers don't have to write the entire program before executing it, the notebook can execute a specific portion of the code. Jupyter Notebook is especially useful in data science based on its uses and the benefits mentioned above.

## 1.2 Problem Statement

Both Jupyter Notebook and Drasil focus on scientific computing, therefore, we are interested in generating Jupyter Notebook in Drasil. Following are the three main problems we are trying to solve:

1. We need to generate document format that is readable by the notebook. Jupyter Notebook is a simple JSON document with .ipynb file extension. Drasil can only write in HTML and LaTeX before; we will need Drasil to be able to generate documents in JSON format so the documents can be read and written in Jupyter Notebook.

2. Not only being able to generate SRS in Drasil, we are trying to expand the application of Drasil and increase the automated database by generating a simple physics lesson plan. We need to capture the elements of chapters, found the family of lesson plans, and classified the knowledge in order to build a general structure in Drasil so the lesson plan will be able to generalize to other lessons.

3. Jupyter Notebook is an interactive application for creating documents that combine text and executable code. However, Drasil doesn't support interactive receipes right now. We are looking for the possibility to generate a document with a mix of text and code, increasing the ability of Drasil and the potential to solve more scientific problems.

By generating Jupyter Notebook in Drasil, we can extend the benefits of Jupyter Notebook, expand the flexibility and the kind of knowledge we can manipulate. In addition, we will be able to improve the reusability, usability, reliability, and modifiability of the generated software artifacts.

## 1.3 Thesis Outline

Thesis outline here.

# Chapter 2

# Your Chapter Title

This is a sample chapter

If you need to use quotes, type it “like this”.

## 2.1 Referencing

These are some sample references to GAMYGDALA (Popescu et al., 2014) from the `references.bib` file and state effects of cognition (Hudlicka, 2002) from the `references_another.bib` file. These references are not in the same .bib file.

## 2.2 Figures

This is a single image figure (Figure 2.1):

This is a multi-image figure with a top (Figure 2.2a) and bottom (Figure 2.2b) aligned subfigures:

Figure 2.1: This is a single figure environment

## 2.3 Tables

Here is a sample table (Table 2.1):

| | | |
|---|---|---|
| A | $\longleftrightarrow$ | B |
| C | $\longleftrightarrow$ | D |

Table 2.1: A sample table

### 2.3.1 Long Tables

A sample long table is shown in Appendix B.

## 2.4 Equations

Here is a sample equation (Equation 2.4.1):

$$y = mx + b \tag{2.4.1}$$

(a) Figure 1

(b) Figure 2

Figure 2.2: A Multi-Figure Environment

# Chapter 3

# Conclusion

Every thesis also needs a concluding chapter

# Appendix A

# Your Appendix

Your appendix goes here.

# Appendix B

# Long Tables

This appendix demonstrates the use of a long table that spans multiple pages.

| **Col A** | **Col B** | **Col C** | **Col D** |
|---|---|---|---|
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |

*Continued on the next page*

*Continued from previous page*

| **Col A** | **Col B** | **Col C** | **Col D** |
|---|---|---|---|
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |
| A | B | C | D |

# Bibliography

Eva Hudlicka. 2002. This time with feeling: Integrated model of trait and state effects on cognition and behavior. *Applied Artificial Intelligence* 16, 7-8 (2002), 611–641.

Adrian Popescu, Joost Broekens, and Maarten van Someren. 2014. GAMYGDALA: An emotion engine for games. *IEEE Transactions on Affective Computing* 5, 1 (2014), 32–44.