

**University Institute of Information Technology,**

**PMAS-Arid Agriculture University,**

**Rawalpindi Pakistan**

**Image based Malware Classification**

***By***

**Shahid Akhter 17-ARID-2038**

**Hamza Khan 17-ARID-1996**

**Shuja Sultan xx-ARID-2042**

***Supervisor*Mr. Zeeshan Javed**

***Bachelor of Science in Information Technology***

***(2017-2021)***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others**.

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software documentation and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Shahid Akhter Hamza Khan Shuja Sultan

--------------------------- --------------------------- ---------------------------

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (IT) “Image based malware classification” was developed by “Shahid Akhter**, 17-ARID-2038”**, “Hamza Khan**, 17-ARID-1996”** and “Shuja Sultan**, 17-ARID-2042”** under the supervision of “Mr Zeeshan Javed” and that in their opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Information Technology.

---------------------------------------

**Supervisor**

---------------------------------------

**External Examiner (If any)**

---------------------------------------

**Administrator UIIT**

**Executive Summary**

In public places, there is often a need for monitoring people and different activities going on, which can be referred later for many reasons including security. Appointing humans for this task involves many problems such as increased employee hiring, accuracy problem, trust, no proof for later use, and also the fact that a human can remember things till a certain time limit. Talking about the current security system, they use dumb still cameras with a continuous recording facility ir-respective of the fact that any event may happen or not. Moreover they are usually pointing at a specific user defined locations so more than one cameras are required to cover the entire region.

To prevent all these problems from prevailing, the CSCS is developed. It is a surveillance system, which provides solution to many of these problems. It is a stand-alone application which doesn’t require any computer to operate. It monitors different situations using a camera which is able to rotate intelligently based on sensor messages and captures the scene in the form of video or photos later reference as well.

**C**ustomizable **S**urveillance **C**ontrol **S**ystem **(CSCS)** is a surveillance system that can be assigned a sensor type as in our case a heat sensor is used, it works accordingly, rotates the camera upon event detection and perform user defined actions like capturing video and stores them, for the future use.

It is an embedded system consisting of Linux fox kit with embedded a running server application also a camera, USB storage device and a sensor node base station is attached with fox kit. LAN communication is used by user to download the videos and to operate the system manually.

**Acknowledgement**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Dr. Kashif Sattar” and our Co-Supervisor “Dr. Tariq Ali” for personal supervision, advice, valuable guidance and completion of this project. We are deeply indebted to them for encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

Shahid Akhter Hamza Khan Shuja Sultan

--------------------------- --------------------------- ---------------------------

**Abbreviations**

|  |  |
| --- | --- |
| **SRS** | Software Requirement Specification |
| **PC** | Personal Computer |
|  |  |
|  |  |
|  |  |

**Table of Contents**

**Introduction 1**

[1.1 Brief](#_Toc268523777) 2

[1.2 Relevance to Course Modules](#_Toc268523779) 2

[1.3 Project Background](#_Toc268523780) 3

[1.4 Literature Review](#_Toc268523780) 3

[1.5 Methodology and Software Life Cycle](#_Toc268523782) 5

**Problem Definition** 6

[2.1 Purpose](#_Toc268523795) 7

[2.2 Product Functions](#_Toc268523796) 7

[2.3 Proposed Architecture](#_Toc268523804) 7

[2.4 Project Deliverables](#_Toc268523806) 8

[2.5 Operating Environment](#_Toc268523807) 8

[2.6 Assumptions and Dependencies](#_Toc268523808) 8

**Requirement Analysis** 9

[3.1 Functional Requirments](#_Toc268523823) 10

[3.2 Non – Functional Requirments](#_Toc268523825) 11

[3.2.1 Usability](#_Toc268523787) 11

[3.2.2 Reliability](#_Toc268523787) 11

[3.2.3 Performance](#_Toc268523787) 11

[3.2.4 Supportability](#_Toc268523787) 11

[3.2.5 Design Constraints](#_Toc268523787) 11

[3.2.6 Licensing Requirements](#_Toc268523787) 11

[3.3 Use case Model](#_Toc268523823) 12

[3.3.1 Use Case Diagarm](#_Toc268523787) 12

[3.3.2 Actors Discription](#_Toc268523787) 16

[3.3.3 Use Case Discription](#_Toc268523787) 17

**The Design** 21

[4.1 UML Structural Diagrams](#_Toc268523830) 22

[4.1.1 Component Diagram](#_Toc268523787) 22

[4.1.2 System Component Diagram](#_Toc268523787) 24

[4.1.3 Package Diagram](#_Toc268523787) 25

[4.1.4 Deployment Diagram](#_Toc268523787) 26

[4.2 UML Behavioral Diagrams](#_Toc268523830) 27

[4.2.1 Activity Diagrams](#_Toc268523787) 27

[4.2.2 State Machine Diagrams](#_Toc268523787) 29

[4.3 UML Interaction Diagrams](#_Toc268523830) 30

[4.3.1 Sequence Diagrams](#_Toc268523787) 30

[4.4 Node Structure](#_Toc268523830) 31

[4.5 Communication Design Protocol](#_Toc268523830) 32

**Implementation** 33

[5.1 Communication Protocol Implementation](#_Toc268523830) 34

[5.2 PC Application Implementation](#_Toc268523830) 36

[5.3 Embedded Application Implementation](#_Toc268523830) 38

[5.4 Wireless Sensor Application Implementation](#_Toc268523830) 51

**Testing and Evaluation** 52

[6.1 Verification](#_Toc268523830) 53

[6.1.1 Functional Testing](#_Toc268523787) 53

[6.1.2 Static Testing](#_Toc268523787) 57

[6.2 Validation](#_Toc268523830) 57

[6.3 Usability Testing](#_Toc268523830) 57

[6.4 Unit Testing](#_Toc268523830) 57

* 1. [Integration Testing](#_Toc268523830) 57

6.6 [System Testing](#_Toc268523830) 57

**GUI** 58

**Future Work** 70

**References** 72

**List of Figures**

Fig 1.1 Block Diagram 8

Fig 2.1 Use Case Diagram 9

# Chapter 1: Introduction

# Brief

The most challenging part of internet security is discovering malware variants. The exponential increase in malware attacks has become one of the major threats to Internet security. A recent threat report from Symantec indicated that their 123 million sensors record thousands of malicious threat events per seconds on daily basis. The presence of malware in the internet of things (IoT) and mobile devices increased. According to the latest threat report from Kaspersky Lab in 2019, remove the number of users that encountered Android malware more than tripled to 1.7 mil- lion globally. Connectivity between an IoT device and a personal computer is established through a cloud service. The complex IoT hardware and software environments, provides more opportunities for potential adversaries to attacks. Malware is separated into various classes by their functionalities i.e., Viruses, Worms, Trojans, and Backdoors. These classes further divide into families based of the type of variants. Malware writers deploy many obfuscation methods such as dead-code insertion, subroutine reordering, and code transposition, to create variants of an existing malware family in order to evade detection.

# Relevance to Course Modules

This project cover our different course modules like software engineering, Artificial intelligence etc. In this project we will use different concepts like machine learning, image processing, static and dynamic visualization, so we will learn all these concepts.

# Project Background

Our project work for classifying the malware images into its different groups/families and also detect malicious content by using the method of images classification. We want optimal performance from the system.

# Literature Review

In law enforcement agencies throughout the world, there are growing digital forensic backlogs of un imaged, unprocessed, and unanalyzed digital devices stored in evidence lockers. The sheer volume of cases requiring digital forensic processing extends far beyond digitally executed crimes such as phishing, online sharing of illicit content, online credit card fraud, etc. In this model deep learning is used for classification of malware, the final model has 98:8% accuracy based on the validation data, and it requires raw binary of the file for analysis.

In this system different pretained models like ResNet50, VGG16 etc. are used for the classification of malwares by using images. This model use Static analysis and visualization analysis for the detection and classification of malwares. Using image processing technique, Nataraj et al. (2011) visualized malware binaries into grayscale. Through a machine learning approach such as GIST they extracted features from malware grayscale. Accuracy reached 97.18% on a dataset containing 9458 malware samples related to 25 different malware families. it takes 1.18 s on average to identify new malware samples but if the number of images increase its average time will increase.

This Deep learning model use both supervised and unsupervised learning model for the training purpose while in these learning models machine learning is used and deep learning is also used with grayscale images and all these come forward from binary files, in the end the classification is performed. But it achieved about 98.6% accuracy for SVM based malware detection but it consumes much time.

Hybrid deep learning model (IMCFN) which combines the visualization and fine-tuned CNN architecture for malware detection and classification that are computationally cost-effective IMCFN algorithm is mainly divided into two parts: malware image generation and CNN fine-tuning via backpropagation technique. During the fine-tuning, they utilized data augmentation technique to improve the performance. The experimental investigation analysis was carried out by implementing the various programs in Python Programming Language. The experiment was run in NIVIDIA Ti-1080 12GB GPU for training and Intel Core i7-4790 processor with 8 GB main memory for classification.

# Analysis from Literature Review (in the context of your project)

We mainly use Image processing in our model with malmig dataset, in order to get an enhanced image or to extract binary data information from it and we will use combined dynamic and static visualization with malmig dataset to add up more accuracy. We can also create our own dataset for model training purpose. We will use combined approach and then compare the results in such a way we can get comparatively more accuracy in less time.

# Methodology and Software Lifecycle for This Project

In this project AGILE methodology will be used because this is suitable for such kind of models.

In software development, agile practices approach discovering requirements and developing solutions through the collaborative effort of self-organizing and cross-functional teams and their customer/end user.

We describe our deep learning based approach for malware classification in detail, including the data set that will be used for experiments, data preprocessing, deep learning architectures, and experimental design. We will use agile method and deliver the project in increments.

SDLC is also used for the development of the whole project from fusibility to delivery of the project.

* + 1. Rationale behind Selected Methodology

We selected the agile method for the development of project because in each and every phase of development we requires the comparison and consultancies with the client to deliver the better product.

**Chapter 2: Problem Definition**

This chapter discusses the precise problem to be solved. It should extend to include the outcome.

# Problem Statement

In All previous shared models of Malware classification accuracy is compromised and they clams high accuracy apart from that they take much time but gives less accuracy, several models acquire high cost and their datasets are fabricated. Like in IMCFN the experiment was run in NIVIDIA Ti-1080 12GB GPU which is costly product. Several models are only for technical persons, non-technical person may have no knowledge that how to use trained model or how to use model for taking better results. As we know numbers of devices and users on internet increasing day by day and threat to data is very high because cyber security will be compromised. With the growth of technology, the number of malware are also increasing day by day. Malware now are designed with mutation characteristic which causes an enormous growth in number of the variation of malware. Not only that, with the help of automated malware generated tools, with these growths in new malware, traditional signature based malware detection are proven to be ineffective against the vast variation of malware. On the other hand, machine learning methods for malware detection are proved effective against new malwares. At the same time, machine learning methods for malware detection have a high false positive rate for detecting malware.

# Deliverables and Development Requirements

Our project plan is to ensure development of a code using python which can detect malware or malicious and classify the malware by using its images. The malware detection and classification will be performed using machine learning. Which can detect the malicious code. if the code is clean the user will be notified about it.

Developmental requirements for the project is given as follow.

The main steps performed through this framework are sketched as follows:

A set of features is computed for every binary file in the training or test datasets (presented in Section II), based on many possible ways of analyzing a malware.

A machine learning system based firstly on one-sided perceptron’s, and then on feature mapped one-sided perceptron’s and a kernelized one-sided perceptron’s, combined with feature selection based on the F1 and F2 scores, is trained on a medium-size dataset consisting of clean and malware files. Cross-validation is then performed in order to choose the right values for parameters. Finally, tests are performed on another, non-related dataset.

In the end we will analyze different aspects involved in the scale-up of our framework to identifying malware files on very large training datasets.

# Current System (if applicable to your project)

A brief description of an existing system. Figure 2.1 is the sample figure, please follow the same style of numbering and caption for the whole report.



**Figure 2.1: Sample picture**

The following table (Table 2.1) is sample table; please follow the same style of numbering and caption for the whole report.

**Table ‎2.1: Sample Table**

|  |  |  |
| --- | --- | --- |
| **Header 1** | **Header 2** | **Header 3** |
| Text | Text | Text |
|  |  |  |

The following list style is the sample to consistently follow in the whole report.

* List items 1
* List items 2

# Chapter 3: Requirement Analysis

Software Requirements Specification (SRS) report should be included in this chapter.

# Use Cases

Malware (short for “malicious software”) refers to any software that can be harmful to the host machines. And can cause potential destruction and data loss in their wake, many viruses include functionality that allows them to replicate and spread to other devices. Our aim to detect this malicious code and warn the owner about it. So that the system remain protected.

|  |  |
| --- | --- |
| **Use Case ID:** | 3.1.1 |
| **Use Case Name:** | Imaged Malware Classification |
| **Actors:** | Users who will perform classification |
| **Description:** | [Provide a brief description of the reason for and outcome of this use case.] |
| **Trigger:** | 1. User will submit malware file. 2. User load data in Cuckoo sandbox and received report in textual form. 3. User upload the textual report file. 4. System will analyze both files and generate results |
| **Preconditions:** | 1. Dataset of malware files must be available. 2. Textual report file must be available. 3. Cuckoo must be installed to evaluate the malware. |
| **Post conditions:** | User Action.  User will load Image dataset, report file in the system.  System Response.  1. Firstly the malware file should be detected on the client machine.  2. Evaluate the behavior of malware file.  3. Classify the malware on the basis of behavior.  4. Final result is generated, |
| **Normal Flow:** | 1. User will upload image dataset. 2. User will upload Text dataset. 3. The cuckoo sandbox starts evaluation. 4. It stores the malware file in database. 5. Then it will run the malware file on the malware machine. 6. It will detect the malware. 7. After the behavior evaluation. It will classify based on data. 8. It will present the final result. |

# Use case Diagram

# 

# Functional Requirements

Functional requirements represent the technical specification of the system. The following table shows what the system tries to accomplish.

b

|  |  |  |
| --- | --- | --- |
| **Priority** | **Requirement** | **Description** |
| High | Visualize the detected malware (Graphical User Interface). | The system must have a Graphical User Interface so that the user can run the malware scanning on his file system. Without the GUI the user will be unable to tell if a malware is detected. |
| High | Dataset Train | Images from malware files are obtained. These files are called dataset and will be used to train the machine learning algo. |
| High | Dataset Validate | Images are ran through validation process after the training of machine learning algo. |
| Low | Have network or web based protection | Since we web protection is out of scope for this project. We are not considering it. |

# Non-Functional Requirements

In contrast with functional requirements, non-functional requirements are used to describe what the system should be, rather than what the system should do. They focus on the quality of the system. That being said, the following non-functional requirements were taken into consideration in this project:

|  |  |  |
| --- | --- | --- |
| **Priority** | **Requirement** | **Description** |
| High | Performance | Performance is a fundamental nonfunctional requirement since the machine will be trained using dataset and may have to run for a long period of time. |
| High | Scalability | This non-functional requirement is just as important as the performance one. Without scalability with the growing number of malware in the database the tool will be useless. |
| High | Reliability | Mean time between failures of the system must be minimal. |
| High | Testability | The system structure should aim to be capable of being tested thoroughly with automated tests. This will improve Reliability. |

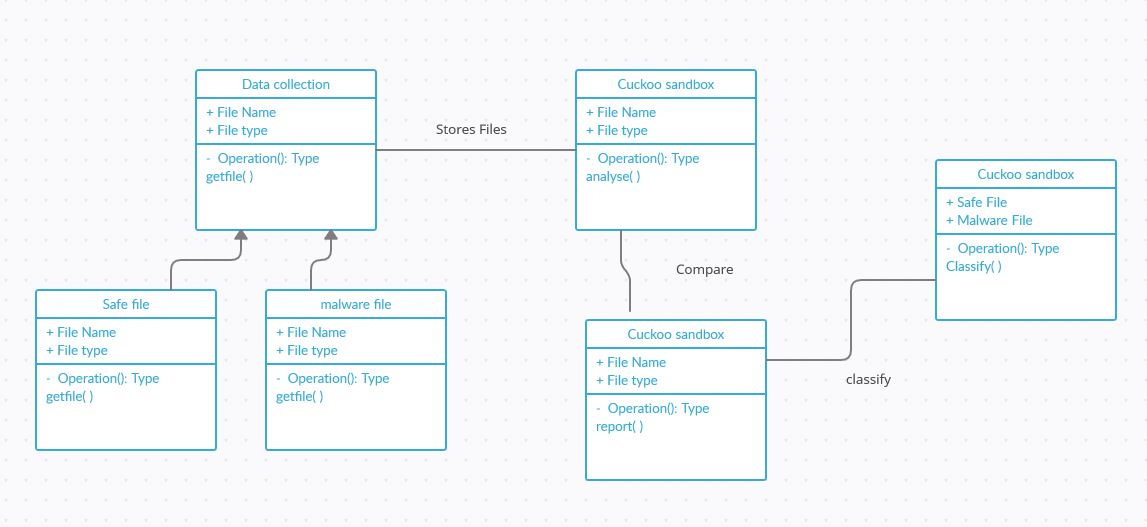
# Chapter 4: Design and Architecture

This chapter will discuss the design and architecture of your system.

# System Architecture

Explain and justify the choice of system architecture for your project.

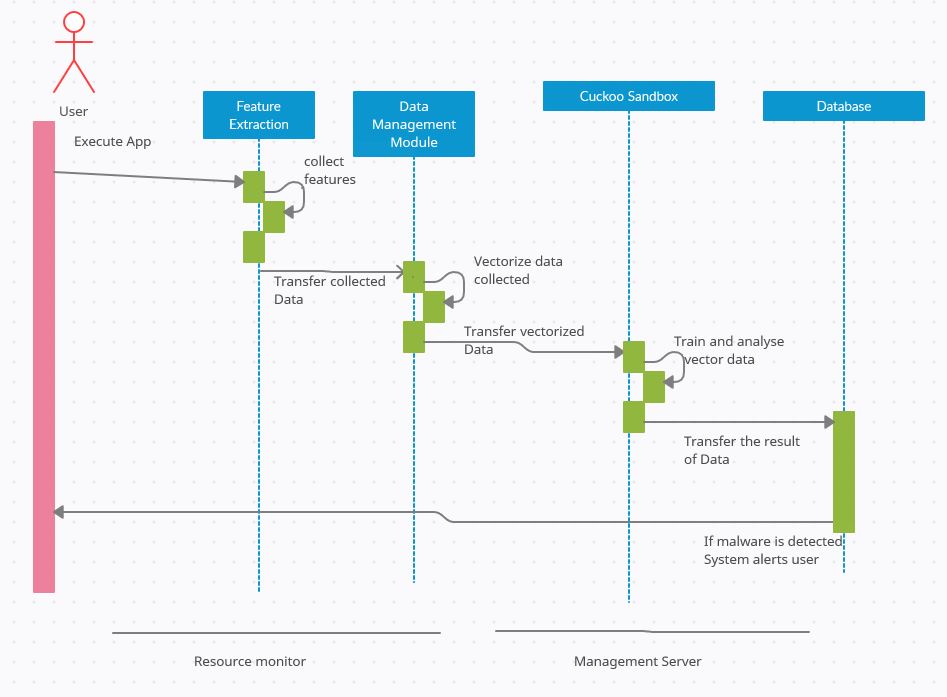
**Class Diagram:**



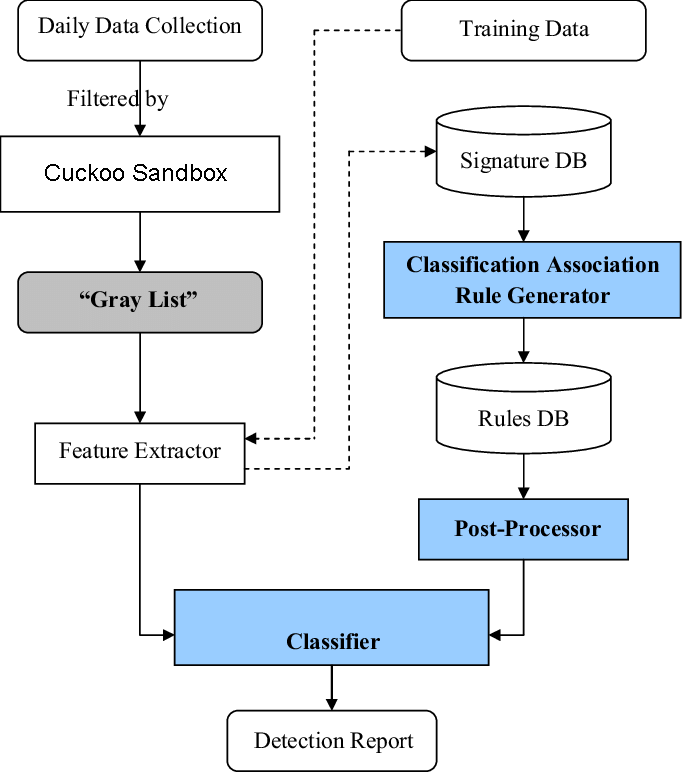
# System Design

As system design varies from system to system, therefore you are required to explore which design pattern is suitable for your system. For guidelines an IEEE Recommended Practice for Software Design Descriptions (section 5 and 6) is provided with this template.

**Sequence Diagram:**

****

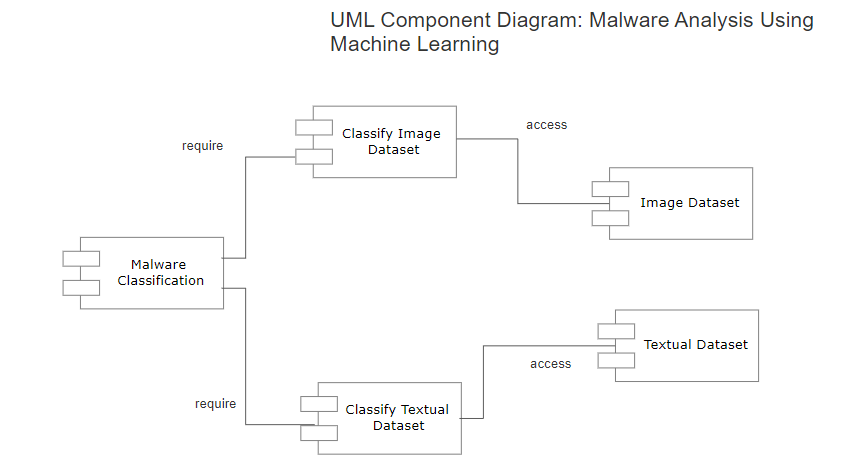
**Data FLOW:**



# Chapter 5: Implementation

This chapter will discuss implementation details supported by UML diagrams (if applicable). You will not put your source code here. Any of the following sections may be included based on your project.

# Component Diagram



# Network and Protocol Choice

It goes here.

# Choice of Object Middleware

RMI vs. CORBA vs. DCOM etc.

# User Interface

Our user interface designs focuses on ease of use. Up to 3 roles can interact with user interface.

1. **Data Trainer:** This user can use Upload Tab to upload image and textual Datasets.
2. **Data Analyzer:** This user can use Analyze Tab to run the datasets through sandbox. Sandbox will analyze the data with the help of machine learning.
3. **Data Classifier:** This user can use Classify Tab to classify the analyzed file as Malicious or Benigh files with the help sandbox using machine learning.

# Chapter 6: Testing and Evaluation

This chapter may include the following sections.

# Verification

Verification section.

# Validation

Validation section.

# Usability Testing

Usability testing section.

# Module / Unit Testing

Unit testing.

# Integration Testing

Integration testing.

# System Testing

System testing.

# Acceptance Testing

Acceptance testing.

# Stress Testing

Stress testing.

# Hardware Configuration for Testing

Hardware configuration.

# Evaluation

Evaluation section.

# Deployment

Evaluation section.

# Maintenance

Evaluation section.

# Chapter 7: Conclusion and Future Work

This chapter concludes the project and highlights future work.

# Conclusion

Conclusion section.

# Future Work

Future work section.

# References

References to any book, journal paper or website should properly be acknowledged. Please consistently follow the style. The following are few examples of different resources i.e. journal article, book, and website.

1 Lyda M.S. Lau, Jayne Curson, Richard Drew, Peter Dew and Christine Leigh, (1999), Use Of VSP Resource Rooms to Support Group Work in a Learning Environment, ACM 99, pp-2. (Journal paper example)

2 Hideyuki Nakanishi, Chikara Yoshida, Toshikazu Nishmora and TuruIshada, (1996), FreeWalk: Supporting Casual Meetings in a Network, pp 308-314 (paper on web) http://www.acm.org/pubs/articles/proceedings/cscw/240080/p308-nakanishi.pdf

3 Ali Behforooz& Frederick J.Hudson, (1996), Software Engineering Fundamentals, Oxford University Press. Chapter 8, pp255-235. (book reference example)

4 Page Author, Page Title, http://www.bt.com/bttj/archive.htm, Last date accessed. (web site)