Project Report on

<Your project title>

<Student Name>
<NCC ID>

Computing Project
Level 5 Diploma in Computing
Softwarica College of IT & E-Commerce
Kathmandu, Nepal

<Report Submission date>

Statement and Confirmation of Own Work

Each time you submit an assignment you must attach this statement as the cover page for both the hard copy and the electronic version. If the statement is missing your work will not be marked.

Student Declaration

I have read and understood NCC Education's Policy on Academic Dishonesty and Plagiarism.

I can confirm the following details:

Programme/Qualification Name: NCC Education Level 5 Diploma in Computing

Student ID/Registration number:

Name:

Centre Name: Softwarica College of IT & E-Commerce

Module Name: Computing Project

Module Leader: Achyut Timsina

Number of words:

I confirm that this is my own work and that I have not plagiarised any part of it. I have also noted the assessment criteria and pass mark for assignments.

Due Date:

Student Signature:

Submitted Date:

Abstract

Here goes your abstract.

Abstract is read at first before reading your report. It should try to summarize your product or idea and lure the potential reader to read through your report to find details of how you have actually build the product. It should be understandable to the non-technical people, or people from other fields as well.

Some tips for writing *abstract*:

- Context/background of the project
- Aim of the project
- The solution that emerged
- Conclusions
- Main recommendations
- Avoid jargon, technical terms (if possible)
- Use layman's terms
- Be generic to specific

Contents

List of Tables

List of Figures

Acknowledgment (Optional)

Here goes your acknowledgment. It is optional.

Chapter 1

Introduction

This chapter should provide:

- Brief, relevant background to the system
- Justification for your project (Why it matters?)
- If any external agency is involved you should explain what the agency is, why the project is required and any other background.
- The system developed (Provide overview of your project, most probably with a rich picture.)
- The solution that emerged (For example: The final solution was developed using PHP and MySQL and implemented on the third party web server.)
- List of main features that your project has
- A short overview of the remaining chapters
- Rich picture and/or overview diagram(s) of your application

1.1 Aims

State two or three major goals/aims of your project. for example:

- To build desktop application for managing student information of ABC college
- To automate the student information managing process in ABC college

1.2 Objectives

Objectives are the set of activities that you perform to achieve the stated aims. Your objectives should be SMART.

Some of the objectives will be technical(for instance, to solve some technical problem) and some will be personal (for instance, to learn a new programming language), and some academic (for instance, to learn different approaches to research activities).

State the objectives that you will have to perform to achieve the stated aim. For example:

- To gather requirements from the stakeholders
- To design ...
- To develop ...
- To test ...
- To document ...
- etc.

1.3 Development methods

Describe which development method you choose and why?

Refer to NCC lecture notes for available types of software development methods. Choose either object-oriented waterfall or agile (iterative) development methodology.

List the reasons for selecting appropriate software development lifecycle model in your project.

Chapter 2

Analysis specification

Why we need to perform analysis? –Give adequate reasons for conducting analysis

Provide guide to the reader of your analysis document (Which section/chapter is where in your document, and what it consists of?) For example: The requirements engineering process is described in Section 2.2. The Use cases and architectures are provided in Section 2.1, and 2.3.

This chapter should have following sections, along with its own introduction and conclusion:

- Use cases
- Requirements
- Architecture

2.1 Use cases

This section should provide:

- The use cases that support the requirements of your project
- Use cases should have a use case diagram
- Each use case should have a scenario description
- For scenario description, you can use a suitable style
  - Paragraph style
  - Single column or sequence of interaction steps

  - More formal styles with pre-conditions, post-conditions, entry criteria, exit criteria, etc.

- You can break down your use case diagram based on major components or modules of your system.
- You should use various relations between use cases whenever applicable. Most frequent relationships are:
  - extends
  - includes
  - uses

One thing you need to have in your mind while developing use cases is "Developing use case is fundamentally a scenario writing activity, not just drawing use case diagrams"(Larman 2012). Hence a typical use case

2.2 Requirements

How you have collected requirements for this project? Describe in detail your requirements capture, analysis, and specification method(s).

2.2.1 Functional requirements

You should at least have **10** functional requirements.

ID: R1

Title: User signup

Description: A new user should be able to register through the web portal. The user must provide user-name, password and e-mail address.

Rational: To acquire users credentials for login process

ID: R2

Title: User login

Description: Existing user should be able to login to the system using existing user name and password. Appropriate message to the user should be provided whether one has entered correct credentials.

Rational: To maintain user security and privacy

2.2.2 Non-functional requirements

This section should provide quality or non-functional requirements for your system. You should at least have 5 testable quality requirements. The non-functional requirements are:

Performance It should specify both the static and the dynamic performance requirements placed on the software or on human interaction with software as a whole. For example: the number of simultaneous users to be supported; amount and type of information to be handled, etc. All the requirements should be stated n measurable terms. For example: *95% of the transactions should be processed in less than 1s.*

Reliability This should specify the factors required to establish the required reliability of the software system at time of delivery. For example: *The system produces reliable results 99% of the time.*

Availability This should specify the factors required to guarantee a defined availability level for the entire system such as checkpoint, recovery, and restart. For example: *The system should be 99% available.*

Security This should specify the factors that protect the software from accidental or malicious access, use, modification, destruction, or disclosure. Specific requirements in this area could include the need to:

- Utilize certain cryptographic techniques;
- Keep specific log or history data sets;
- Restrict communications between some areas of the program;
- Check data integrity for critical variables;

Example non-functional requirement:

ID: Q1

Title: Communication security

Description: The message should be encrypted for log-in communications, so others cannot get user-name and password from those messages.So all of the login messages should be encrypted.

Rational: To maintain user secrecy and privacy

2.2.3 Prioritization

Use MoSCoW prioritization technique to prioritize your all requirements. Provide brief description of how you will perform MoSCow prioritization technique with relevant references.

Provide final result of prioritization in here.

For example (Just an example, but you are free to use your custom way to present the prioritization process):

Requirement	**MoSCoW**
R1. User Authentication	Should
...	...
...	...

Table 2.1: Requirements prioritization using MoSCoW

2.3 Architecture

2.3.1 System architecture

This section should have:

- How you are going make your project easy to change and maintain?
- How you have separated different aspects (persistence, user interface, etc.) of your system?
- You should draw a system architecture diagram and explain it.
- There are various ways you can build your system architecture, but two of the most widely used are Layered architecture and the MVC. Choose one and describe! I suggest you to use MVC.

2.3.2 Initial class diagram

This section should have high level class diagram of your application. You can come up with your initial class diagram using Natural Language Analysis method. The classes you identified in this section are at domain level. It should just have your classes identified from the requirements specification and use cases of your project. It should not consist of classes pertaining to databases, user interfaces, or any other implementation induced classes.

Your class diagram should depict:

- Identified classes from Natural language analysis of your requirements specification
- Relevant attributes and methods of each class
- Relationships between those classes. You should at least identify **inheritance** and **composition** relationship between your classes.
- Finally, you should describe why and how you have come up with the class diagram.

2.4 Conclusion

Provide conclusion for your project analysis specification. It should summarize the activities you did in analyzing the system.

Chapter 3

Design specification

Why we need to perform design? –Give adequate reasons for conducting design

Describe UML and why do we use it for designing OO projects like yours?

Visit the site `www.uml-diagrams.org` for further references on UML.

Which tool you have used in drawing your UMLs? Describe them. (I recommend you to use Visual Paradigm [1])

Provide guide to the reader of your analysis document (Which section/chapter is where in your document, and what it consists of?). For example: Section 3.1 describes structural model of the system. The behavior models are depicted in Section 3.2.

3.1 Structural model

This section shows the static structure of the system and its parts on different abstraction levels and how they are related to each other.

This section should provide detailed application level class diagram for your system. This includes:

- Classes
- Relationships between classes
- Attributes and methods
- Appropriate description of the drawn class diagrams and identified relationships

[1] `www.visual-paradigm.com`

This section should elaborate on the initial class diagram you have drawn in previous analysis document. This class diagram should reflect your actual code structure. It includes not just the domain level classes but also implementation level classes. For example classes for database connections, UI, design patterns, architectural classes. In this section you should also discuss why you have come up with the classes and the relationships. Diagrams without proper description will be poorly evaluated.

Put your ER diagrams here if you have used any RDBMS (Relational database management system). Describe your ER diagram. Put your data dictionary in Appendix of your report.

3.2 Behavior model

Behavior model show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time. This section models how a system should respond to users and evolve over time.

Why behavior model is needed in designing computing projects? What are various ways of modeling behavior of a software project? (sequence diagrams, activity diagrams, collaboration diagrams, etc.)

Describe various behavior model in brief, and discuss suitability of one or two behavior model in your project.

3.2.1 Sequence diagram

Sequence diagrams show the interchange of message between lifelines (objects). It shows the collaboration of objects based on a time sequence. It shows how objects interact with others in a particular *scenario of a use case*. Sequence diagrams shows:

- the order in which methods are invoked in a system.
- the scope, or lifeline, of objects.
- the operation at a higher level of abstraction than an activity diagram.

You should develop a number of sequence diagrams for each major use case in your project. Each sequence diagrams should be properly labeled and described.

3.2.2 Activity diagram

Have your activity diagrams with relevant description in this section.

Activity diagrams show sequence and conditions for coordinating lower-level behaviors, rather than which classifier own those behaviors. These are commonly called control flow and object flow models. The behaviors coordinated by these models can be initiated because other behaviors finish executing, because objects and data become available, or because events occur external to the flow.

Activity diagrams are known as work flow diagrams. They are much like flow-charts, but more structured. Activity diagrams are used to describe the full process behind and internal process or a user request. They describe the logic of the operations that are shown on class diagrams.

Activity diagrams are mostly used for two purposes:

- Outlining the high level activity in a system
- Formally representing algorithms (each activity becomes a line of code)

3.3 Conclusion

Provide conclusion for your project design specification. It should summarize the activities you did in designing the system.

Chapter 4

Implementation

This chapter should discuss:

- Choice of programming language
- System cut-over from the development architecture to the implementation architecture
- Data migration from the development architecture and/or existing systems to the implementation architecture
- Training (how, why and when particular groups of users will use the user guides?)

4.1 Programming language(s)

Which programming language you have used to develop your application? Why you have chosen that particular language? Given detailed description of the particular programming language, including historical background, current development, recent features, language usage context, etc. (Research and write).

4.2 Development environment

You should write about:

- Standard libraries
- IDEs

- Frameworks (testing frameworks, etc.)
- Your development platform
- Other CASE / design tools you have used

4.3 Deployment strategy

You should describe:

- System migration from development environment to production environment
- Data migration from development environment from development environment to production environment
- Can be shown using deployment diagram

4.4 User training

This involves the development of user guide that should be placed in the appendix. The user guide should consist of screen dumps of the system along with supportive narrative that explains how to use a particular screen. There should be a short section in the report that explains how, why and when particular groups of users will be trained.

4.5 Conclusion

Summarize the things you have done in this chapter.

Chapter 5

Other project issues

You should provide the reflection on your project works till now.

5.1 Project management

You should discuss what are deviations between proposed and actual project schedule. The identified deviations with an appropriate reason for such deviation should be explained in this section in detail here.

5.2 Risk management

You should compare the proposed risks with actual risks and find out deviations and/or similarities between them. In case of deviations, you should provide appropriate reason(s) for such deviation in your risk plan.

5.3 Configuration management

This section should provide overview of your configuration management approach and whether it worked or not (discuss).

5.4 Testing

This section should include:

- Reasons for doing testing in computing projects
- Types of testing available

- Your choice of testing methods

5.4.1 Unit testing

Describe unit testing (should answer why, what , how unit testing is done?) Describe your choice of testing framework (if you have used any) For example PHPUnit or JUnit

5.4.2 Integration testing

Describe integration testing (should answer why, what, how integration testing is done?)

List a sample integration test case suitable for your project as shown in the student's guide for the computing project.

You should put your actual test script in the appendix of this report.

Chapter 6

Conclusion

The conclusion is the last section of your report. It is generally read at last and most probably your readers have already read all previous sections of your report. So it is not just enough to restate what you have already written in your report, but should discuss what should your readers do next about the product.

This chapter is a reflective evaluation of your project in terms of:

- Whether or not you have achieved your aims and objectives
- What problems occurred and how you overcame them
- Things that you may do differently in any further projects that you undertake and your reasons for doing so
- State the commercialization potential or practical applications of your outcome

Chapter 7

Future works

Your suggested further extensions to your project can be discussed in this section.

You should use Harvard style of referencing. (Dawson 2005, Weaver 2004). Try to minimize the web references. Maximize the use of books, journal, scientific papers, articles into your report. As you already know that the web references are very volatile and can change pretty frequently, whereas the sources that are more permanent or could be made accessible for a foreseeable future are the sources that I suggest to use.

References

Dawson, C.W. (2005). *Projects in Computing and Information Systems: A Student's Guide*. Addison-Wesley. ISBN: 9780321263551.

Larman, Craig (2012). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3/e*. Pearson Education India.

Weaver, P.L. (2004). *Success in your project: a guide to student system development projects*. Prentice Hall. ISBN: 9780273678090. URL: `http://books.google.com.np/books?id=LKZQAAAAMAAJ`.

Appendices

Appendix A

Test scripts

Appendix B

User guide

Appendix C

Source code (Optional)

You need to handover the source code into a repository in bitbucket or github.