

E-INFO SYSTEM (IT/IS department)

By

Izzar Norwani Mohamad Isa

**Dissertation submitted in partial fulfillment of
the requirements for the
Bachelor of Technology (Hons)
(Information System)**

JUNE 10, 2004

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

t
QA
76.9
.I98
2004

1. Human - computer interaction
2. System design
3. IT/IS - - theory

CERTIFICATION OF APPROVAL

E-INFO SYSTEM (IT/IS DEPARTMENT)

**Research on “HUMAN COMPUTER INTERACTION and GRAPHICAL USER
INTERFACE”**

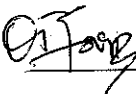
BY,

**IZZAR NORWANI MOHAMAD ISA
1931**

*Dissertation submitted in partial fulfillment of the requirements for the
Bachelor of Technology (Hons)
(Information System)*

10 JUNE 2004

Approved by,




(Mrs. Amy Foong)

**Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan**

CERTIFICATION OF ORIGINALITY

This is hereby that I am responsible for the work submitted in this project, that the original work is my own except a s specified in the references and acknowledgements and the original work contained herein have not undertaken or done by un specified sources or persons



.....
(IZZAR NORWANI MOHAMAD ISA)

ABSTRACT

This project is divided into two terms, first the research on PHP application and second system development on the decision support system on web development. Research web application will be based on the problem statement and objective of the project while the web decision support system is the support idea for the project. The project will require a hybrid model for SDLC methodology.

Reviews on the system will be made according to the SDLC and the objective of the projects. Artificial Intelligent module is used for the web system in determine the best DSS solution for the business. Research will be more on the implementation of the DSS in the web development focus on the cost, availability and the architecture of the DSS. Advantages of this system and several criteria in the system will be part of this project.

TABLE OF CONTENTS

ABSTRACT	4
INTRODUCTION.....	6
LITERATURE REVIEW.....	9
METHODOLOGIES.....	26
FINDINGS / SOLUTION.....	29
CONCLUSION.....	31
REFERENCES.....	32

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND OF STUDY

IT and IS department is a one of the largest academic department in UTP. The department of IT and IS. They present the higher education of UTP student by giving all knowledge and expertise on information technologies and system. But still IT and IS department did not have a portal that can gather all student and lecturer in one roof. Currently, we can see how students find difficulty to find lecturer on the IT/IS department, sometimes lecturer is not in so by developing this portal, student and lecturer just can interact by using chat system and instant messenger that provided by this portal

A student web portal would allow IT/IS students to access online campus services, websites, and course information from one convenient location, using the internet. Students would able to share information among them, view the internal news channels aimed at particular groups of students, and external information such as sports, campus event and entertainment. There also have a page that give information about lecturers in IT/IS department

In developing the student IT /S department portal, student need to come out with the strong reason why should IT/IS department to have a student web portal; interview IT/IS students, staff, and faculty to assess their level of interest in a portal and ensure that those who develop the portal understand the features these stakeholders feel a portal should have; investigate best practices at other Universities that have already deployed student web portals; and suggest management models for the portal project

1.2 PROBLEM STATEMENT

1.2.1 Problem Identification

The current way for the IT/IS department in UTP uses a big number of papers and phone calls. This will cost a lot of money while the UTP management might want to reduce the daily cost.

Mostly people who browsing on UTP websites do not have enough information about what is offered and surrounding in UTP itself. The information in UTP websites just a normal and only little information that can be gathered there. There is no information about lecturers in IT/IS department and staff that are working on that department.

Here on my project, I designing a web sites that can be considered the most E-Info system that can resolve this problem. All information about IT/IS department will be updates and people outside UTP can know what is the real environment in IT/IS department itself

1.2.2 Significant of the Project

Many of students believe that develop IT/IS department portal would help the campus meet a variety of goals shared by students, faculty, IT/IS staff and senior administrators. These include:

- Building harmony community by interact relationship between lecturer and students.
- Studies the powerful HCI elements to eb used
- improving the quality of student life, thereby improving retention
- showcasing what the University offers in IT and IS course

Today, IT/IS did not have its own department portal rather than information just can gathered from UTP websites. This student portal can be maintain by the IT/IS student or lecturer himself. There is will be student involvement in maintaining this portal. This portal function is a place where IT/IS students can share the information, news, and also as a place where all students can communicate with each other.

1.3 OBJECTIVES

The author must accomplish the objective by the end of this project. They are as following:

- To maintain an environment where students and lecturers can interact each other, knowing about each other by using this system
- To develop a user-friendly interface by applying the HCI elements, which will be both functional and easy to use for the users of the system
- To develop a system that is accessible to the users; regardless of location, experience, or the type of computer technology used

1.3 SCOPE OF PROJECT

The system will giving out information about IT/IS department. This system also will scope on HCI elements on web development. HCI (human-computer interaction) is the study of how people interact with computers and to what extent computers are or are not developed for successful interaction with human beings

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The objective of this project is to develop ITIS department portal E-Info system. The author focuses on the each other academic department in other universities. The review centers on what type of services are being offered via Web pages, and how it is presented. In the design of E-Info System pages, the concept of human-computer interaction will be studied and addressed.

2.2 Definition of E-Info system

The official definition according to the www.thefreedictionary.com is:

“A service that provides information and assistance to the users of a computer network service that provides information and assistance to the users of a computer network.”

[1]

E-Info System offers problem solving for general computing issues, hardware, software, and network support for the users.

2.3 Reviewed Web

The web pages were review as to know how other university helpdesks worked, services provided to their students, and other functions/links contained in the web pages;

2.3.1 Alberta University Helpdesk

2.3.1.1 Description

This is a very detailed web page that offers a multitude of information to the users. These pages are intended for University of Alberta staff, students and faculty who require technical support from CNS. [2]

2.3.1.2 Services

The page is provided: Frequently Asked Questions (FAQs), Tutorials, Help on various programs and utilities, Graphics, Netscape, Modems, News and various search engines. An available time is listed for contacting the Helpdesk with questions.

2.3.1.3 Findings

In “Electronic Email Help” link, there are lists of solved problem related to email together with the detail in fixing the problems. While in the Search Engine” link, it offers 6 different search engines in order to gather intended information.

2.3.2 Bates College

2.3.2.1 Description

This home page explains an overview of the structure of the information on the page. There are also user policy, e-mail to the helpdesk, and requests to borrow laptops.

2.3.2.2 Services

This web page provides Supported Software Installation, Tips, Techniques, and FAQs link. It also offers information on various platforms including:

- DOS
- MacIntosh
- OS/2
- UNIX
- Windows

2.3.2.3 Findings

The web page is very helpful in providing the “Virus Warning” link as the users can be alert of the new virus that will affect their PCs.

Besides, the “Tutorial” link provides basic instructions about how to get started with a specific software application or technology. [3]

The other information that is provided relates specifically to the Internet. Internet Guide Reference, News, Technical and Developer's Resources for the Web, and the Usenet FAQ archive. But still there are a few resources available.

Other than that, the phone number and the email of the helpdesk are provided to assist the student in order to log their problems. This will help the author to develop an efficient helpdesk system which will provide the multiple ways to solve the users' problems.

2.5 DEFINITION OF HUMAN-COMPUTER INTERACTION

HCI (human-computer interaction) is the study of how people interact with computers and to what extent computers are or are not developed for successful interaction with human beings. [8]

A significant number of major corporations and academic institutions now study HCI. Historically and with some exceptions, computer system developers have not paid much attention to computer ease-of-use. At the same time, they might argue that computers are extremely complex products to design and make and that the demand for the services that computers can provide has always out driven the demand for ease-of-use.

There is currently no agreed upon definition of the range of topics which form the area of human-computer interaction. Yet we need a characterization of the field if we are to derive and develop educational materials for it. Therefore we offer a working definition that at least permits us to get down to the practical work of deciding what is to be taught:

Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.

From a computer science perspective, the focus is on interaction and specifically on interaction between one or more humans and one or more computational machines. The classical situation that comes to mind is a person using an interactive graphics program on a workstation. But it is clear that varying what is meant by interaction, human, and machine leads to a rich space of possible topics, some of which, while we might not wish to exclude them as part of human-computer interaction, we would, nevertheless, wish to identify as peripheral to its focus. Other topics we would wish to identify as more central.

Take the notion of machine. Instead of workstations, computers may be in the form of embedded computational machines, such as parts of spacecraft cockpits or microwave ovens. Because the techniques for designing these interfaces bear so much relationship to the techniques for designing workstations interfaces, they can be profitably treated together. But if we weaken the computational and interaction aspects more and treat the design of machines that are mechanical and passive, such as the design of a hammer, we are clearly on the margins, and generally the relationships between humans and hammers would not be considered part of human-computer interaction. Such relationships clearly would be part of general human factors, which studies the human aspects of all designed devices, but not the mechanisms of these devices. Human-computer

interaction, by contrast, studies both the mechanism side and the human side, but of a narrower class of devices.

Or consider what is meant by the notion human. If we allow the human to be a group of humans or an organization, we may consider interfaces for distributed systems, computer-aided communications between humans, or the nature of the work being cooperatively performed by means of the system. These are all generally regarded as important topics central within the sphere of human-computer interaction studies. If we go further down this path to consider job design from the point of view of the nature of the work and the nature of human satisfaction, then computers will only occasionally occur (when they are useful for these ends or when they interfere with these ends) and human-computer interaction is only one supporting area among others.

There are other disciplinary points of view that would place the focus of HCI differently than does computer science, just as the focus for a definition of the databases area would be different from a computer science vs. a business perspective. HCI in the large is an interdisciplinary area. It is emerging as a specialty concern within several disciplines, each with different emphases: computer science (application design and engineering of human interfaces), psychology (the application of theories of cognitive processes and the empirical analysis of user behavior), sociology and anthropology (interactions between technology, work, and organization), and industrial design (interactive products). In this report, we have adopted, as an ACM committee, an appropriate computer science point of view, although we have tried at the same time to consider human-computer interaction broadly enough that other disciplines could use our analysis and shift the focus appropriately. From a computer science perspective, other disciplines serve as supporting disciplines, much as physics serves as a supporting discipline for civil engineering, or as mechanical engineering serves as a supporting discipline for robotics. A lesson learned repeatedly by engineering disciplines is that

design problems have a context, and that the overly narrow optimization of one part of a design can be rendered invalid by the broader context of the problem. Even from a direct computer science perspective, therefore, it is advantageous to frame the problem of human-computer interaction broadly enough so as to help students (and practitioners) avoid the classic pitfall of design divorced from the context of the problem.

To give a further rough characterization of human-computer interaction as a field, we list some of its special concerns: Human-computer interaction is concerned with the joint performance of tasks by humans and machines; the structure of communication between human and machine; human capabilities to use machines (including the learnability of interfaces); algorithms and programming of the interface itself; engineering concerns that arise in designing and building interfaces; the process of specification, design, and implementation of interfaces; and design trade-offs. Human-computer interaction thus has science, engineering, and design aspects.

Regardless of the definition chosen, HCI is clearly to be included as a part of computer science and is as much a part of computer science as it is a part of any other discipline. If, for example, one adopts Newell, Perlis, and Simon's (1967) classic definition of computer science as "the study of computers and the major phenomena that surround them," then the interaction of people and computers and the uses of computers are certainly parts of those phenomena. If, on the other hand, we take the recent ACM (Denning, et al., 1988) report's definition as "the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application," then those algorithmic processes clearly include interaction with users just as they include interaction with other computers over networks. The algorithms of computer graphics, for example, are just those algorithms that give certain experiences to the perceptual apparatus of the human. The design of many modern computer applications inescapably requires the design of some

component of the system that interacts with a user. Moreover, this component typically represents more than half a system's lines of code. It is intrinsically necessary to understand how to decide on the functionality a system will have, how to bring this out to the user, how to build the system, how to test the design.

Because human-computer interaction studies a human and a machine in communication, it draws from supporting knowledge on both the machine and the human side. On the machine side, techniques in computer graphics, operating systems, programming languages, and development environments are relevant. On the human side, communication theory, graphic and industrial design disciplines, linguistics, social sciences, cognitive psychology, and human performance are relevant. And, of course, engineering and design methods are relevant.

2.6 HISTORY OF HUMAN COMPUTER INTERACTION

Human-computer interaction arose as a field from intertwined roots in computer graphics, operating systems, human factors, ergonomics, industrial engineering, cognitive psychology, and the systems part of computer science. Computer graphics was born from the use of CRT and pen devices very early in the history of computers. This led to the development of several human-computer interaction techniques. Many techniques date from Sutherland's Sketchpad Ph.D. thesis (1963) that essentially marked the beginning of computer graphics as a discipline. Work in computer graphics has continued to develop algorithms and hardware that allow the display and manipulation of ever more realistic-looking objects (e.g., CAD/CAM machine parts or medical images of body parts). Computer graphics has a natural interest in HCI as "interactive graphics" (e.g., how to manipulate solid models in a CAD/CAM system).

A related set of developments were attempts to pursue "man-machine symbiosis" (Licklider, 1960), the "augmentation of human intellect" (Engelbart, 1963), and the

"Dynabook" (Kay and Goldberg, 1977). Out of this line of development came a number of important building blocks for human-computer interaction. Some of these building blocks include the mouse, bitmapped displays, personal computers, windows, the desktop metaphor, and point-and-click editors (see Baecker & Buxton, 1987, Chapter 1).

Work on operating systems, meanwhile, developed techniques for interfacing input/output devices, for tuning system response time to human interaction times, for multiprocessing, and for supporting windowing environments and animation. This strand of development has currently given rise to "user interface management systems" and "user interface toolkits".

Human factors, as a discipline, derive from the problems of designing equipment operable by humans during World War II (Sanders & McCormick, 1987). Many problems faced by those working on human factors had strong sensory-motor features (e.g., the design of flight displays and controls). The problem of the human operation of computers was a natural extension of classical human factors concerns, except that the new problems had substantial cognitive, communication, and interaction aspects not previously developed in human factors, forcing a growth of human factors in these directions. Ergonomics is similar to human factors, but it arose from studies of work. As with human factors, the concerns of ergonomics tended to be at the sensory-motor level, but with an additional physiological flavor and an emphasis on stress. Human interaction with computers was also a natural topic for ergonomics, but again, a cognitive extension to the field was necessary resulting in the current "cognitive ergonomics" and "cognitive engineering." Because of their roots, ergonomic studies of computers emphasize the relationship to the work setting and the effects of stress factors, such as the routinization of work, sitting posture, or the vision design of CRT displays.

Industrial engineering arose out of attempts to raise industrial productivity starting in the early years of this century. The early emphasis in industrial engineering was in the

design of efficient manual methods for work (e.g., a two-handed method for the laying of bricks), the design of specialized tools to increase productivity and reduce fatigue (e.g., brick pallets at waist height so bricklayers didn't have to bend over), and, to a lesser extent, the design of the social environment (e.g., the invention of the suggestion box). Interaction with computers is a natural topic for the scope of industrial engineering in the context of how the use of computers fit into the larger design of work methods.

Cognitive psychology derives from attempts to study sensation experimentally at the end of the 19th century. In the 1950's, an infusion of ideas from communications engineering, linguistics, and computer engineering led to an experimentally-oriented discipline concerned with human information processing and performance. Cognitive psychologists have concentrated on the learning of systems, the transfer of that learning, the mental representation of systems by humans, and human performance on such systems.

Finally, the growth of discretionary computing and the mass personal computer and workstation computer markets have meant that sales of computers are more directly tied to the quality of their interfaces than in the past. The result has been the gradual evolution of standardized interface architecture from hardware support of mice to shared window systems to "application management layers." Along with these changes, researchers and designers have begun to develop specification techniques for user interfaces and testing techniques for the practical production of interfaces.

2.1 DEFINITION OF GRAPHICAL USER INTERFACE

Graphical user interface is a user interface based on graphics (icons and pictures and menus) instead of text; uses a mouse as well as a keyboard as an input device. [9]

Graphical user interface (GUI) is an operating system or application interface that includes graphical (versus text-based) elements such as windows, pull-down menus, buttons, scroll bars, iconic images, wizards, the mouse, etc. Sound, voice, motion video,

and virtual reality interfaces may become part of the GUI for many applications in the future. A system's graphical user interface along with its input devices is sometimes referred to as its "look-and-feel."

2.6.1 PRINCIPLE OF GOOD GUI

Successful GUIs share many common characteristics. One of them is speed. The developer can give a GUI the appearance of speed in several ways. First, avoid repainting the screen unless it is absolutely necessary. Secondly, have all field validations occur on a whole-screen basis instead of on a field-by-field basis. Also, depending upon the skills of the user, it may be possible to design features into a GUI that give the power user the capability to enter each field of each data record rapidly. Such features include mnemonics, accelerator keys, and toolbar buttons with meaningful icons, all of which would allow the speed user to control the GUI and rate of data entry. [6]

Joel Spolsky in his book explain about focusing on the logic of good user interfaces and pushing to develop a good user model is bound to resonate and get programmers to think about making their interfaces logical from the user's perspective, rather than the perspective of the inner architecture, which the user could typically care less about. [7]

Controls are the visual elements that let the user interact with the application. GUI designers are faced with an unending array of controls to choose from. Each new control brings with it expected behaviors and characteristics. Choosing the appropriate control for each user task will result in higher productivity, lower error rates, and higher overall user satisfaction.

2.6.2 TEN PRINCIPLES FOR GOOD GUI DESIGN

- 1) The user must be able to anticipate a widget's behavior from its visual properties. Widgets in this context refer to visual controls such as buttons, menus, check boxes, scroll bars, and rulers. So let's call this the Principle of Consistency at the Widget Level. This principle stresses that every widget in your application should behave consistently. If one button responds to a single mouse click then every button should respond to a single click. In software development environments such as Delphi you can create your own widgets. If your application requires a new widget that behaves differently from a common or closely related widget, anticipate the confusion and give your new widget a distinctive appearance. Use metaphor affordances whenever possible to make your widget's appearance tell the user how that widget behaves.

- 2) The user must be able to anticipate the behavior of your program using knowledge gained from other programs. This is the Principle of Consistency at the Platform Level. Consistency is important not only to visual elements like widgets but to abstractions such as mouse gestures, accelerator keys, placement of menus, and icons and toolbar glyphs. There are plenty of decisions regarding GUIs that are arbitrary and platform-specific. Obtain a good GUI application design guide for your target platform, and follow it. If you feel compelled to improve upon conventions, you will more than likely undo your "improvements" after users complain about them. If you are doing cross-platform development, maintain consistency with the host platform. Maintaining consistency with the host platform trumps achieving consistency of your application across platforms. Your users will change applications on the same platform far more frequently than they will run your application on different platforms.

3) View every user warning and error dialog that your program generates as an opportunity to improve your interface. Good GUI interfaces rarely need or use warnings and error dialogs. Exceptions include those that signal hardware problems such as a disk failure or lost modem connection, or warnings that ask the user's permission to perform an irreversible and potentially erroneous step. Otherwise, error dialogs in GUI interfaces represent interface design flaws. Prevent, don't complain about, user errors. The most common flaws arise from improperly formatted user input and inappropriate task sequencing. Design your program interface to help your users enter appropriate data. If your program requires formatted data (dates, currency, alphanumeric only, or a particular numeric range) use bounded input widgets that appropriately limit the user's input choices. If a certain program step cannot be legitimately performed until your user completes other steps, disable the dependent step until all its dependencies are satisfied. Most GUI environments dim disabled widgets to signal that the widget cannot be selected. Use disabled widgets to limit user actions to those that are valid.

4) Provide adequate user feedback. Like the Consistency Principle, the Principle of User Feedback applies to widgets and to program activity. Widgets often provide visual feedback; click a button, and it briefly suggests it has been depressed. Click a check box and its new appearance alerts the user it has been selected or deselected. If you create your own widgets, be sure to provide users with visual feedback for each affordance. User feedback at the program level requires that users know whether a step is in progress or completed. Change the cursor (the Mac wristwatch or the Window hourglass) to indicate brief delays, and use progress bars to indicate progress of longer tasks. Design every screen in your application so a novice user can easily tell what steps, especially critical steps, have been performed.

- 5) Create a safe environment for exploration. Humans are born explorers. Great interfaces invite and reward exploration, and offer the novice both the thrill of discovery and the satisfaction of unassisted accomplishment. Some interfaces encourage users to explore unfamiliar features, others do not. By allowing users to undo or redo, you encourage them to explore your application without fear of corrupting the database. Undo/Redo capabilities also eliminate the need for dialogs requesting permission to perform a seemingly erroneous function. A good interface makes a user feel competent, while poor interfaces leaves the same user feeling incompetent.

- 6) Strive to make your application self-evident. Good applications have comprehensive manuals and online help materials explaining program features and how to use them to solve real world problems. Great applications are those whose novice users rarely need to refer to the manuals or online help. The difference between good and great is often the degree to which the application and its interface are self-evident. From your choice of labels and widget captions to the arrangement of widgets on the screen, every interface design decision you make needs to be tested by users. Your goal is to create an interface that needs no explanation. A pharmacy management system interface need not be self-evident to a newspaper editor, but it should be to every pharmacist.

- 7) Use sound, color, animation and multimedia clips sparingly. Sound, color, animation and multimedia presentations are appropriate for education or entertainment, but

effective use in other applications is difficult. Most platforms have written conventions that describe the appropriate use of sound, color and animation. Follow them, and remember never to use color or sound as the sole means of communicating with the user (many users are colorblind or hearing-impaired). Remember that these components must pass the same usability tests as all other application features: include them only if they improve your users' ability to accomplish tasks. Returning to our example, tax programs may use multimedia clips to extend the utility of the software as part of their online educational strategy, for interested users seeking added tax information. They are a software asset because these clips are under user control and independent to the task of completing the tax forms.

- 8) Help users customize and preserve their preferred work environment. If your application will be installed and operated by a single user, preserving the work environment may be as simple as creating a few registry entries such as the window's latest size and position. However, applications designed for multiple users or installed on different computers must address additional issues. Most common among these are video issues -- both hardware-specific involving the display (screen size, video resolution and color depth) and user-specific such as poor visual accommodation and acuity or colorblindness. Keep in mind that regardless of programming, the characteristics of the user's display will affect your application's appearance; your full-screen interface may look fine on a 14-inch VGA display, but will those 8-point Times-Roman labels and captions be legible on a 17-inch display at a resolution of 1152x864? One popular solution to any hardware irregularities or user preferences is to permit the user to tailor the basic interface. Common user-tailored details include fonts, toolbar location and appearance, menu entries, and (especially important for users with impairments) color and sound. It is helpful to give users a way to choose among several predefined schemes, and always include a

way to return to the default color or sound scheme. If multiple users take turns using your application at a single workstation, consider recording preferences as user-specific profiles rather than as a single description of the application's appearance the last time it was run.

- 9) Avoid modal behaviors. Programs using modal behavior force the user to perform tasks in a specific order or otherwise modify the user's expected responses. Modal behaviors generally feel uncomfortable to the user because they restrict more intuitive or natural responses, but if consciously and thoughtfully applied they can be used to advantage. For example, "Wizard" type tools simplify complex tasks by modal behavior. Warnings and error messages are also typically modal, forcing users to first address a critical issue before returning to the task. Modality in this latter context is necessary but interrupts the user's concentration and goal-oriented behavior, and so is another reason to avoid unnecessary warning and error messages (see Principles 3 and 5 above). The best modal behaviors are subtle but not hidden, and come forth naturally as a consequence of the metaphor.

In a typical painting program, for example, selecting a widget generally alters the subsequent function of the program and therefore results in modal behavior. Pick the brush widget, and you are ready to paint. Pick the letter stencil widget, and you type some text. Pick a shape widget, and you then draw that shape. This rarely causes confusion because the modal behavior is based on a real world analogy; we already know that by selecting a drawing instrument we are limiting the color, texture and line thickness that will appear on our paper. Good interfaces reveal the palette selection at a glance, and change the cursor to provide additional visual feedback once a selection is made.

So if your application absolutely requires modal behavior, bind that behavior to a strong metaphor and give your user visual feedback so the mode is natural and obvious.

- 10) Design your interface so that your users can accomplish their tasks while being minimally aware of the interface itself. We could call this the Principle of Transparency. Interface transparency occurs when the user's attention is drawn away from the interface and naturally directed at the task itself. It is the product of several factors, including a screen layout that puts both tools and information where the user expected them to be; icons and labels that are meaningful to the user; and metaphors (including gestures) that are easy for users to recognize, learn, remember, and perform. Choosing good metaphors and adhering to the above principles are an important start, but the most direct way to insure a transparent interface is to perform user testing throughout the program's creation.

Table 1: Guideline for Using Controls

Control	Number Of Choices In Domain Shown	Type Of Controls
Menu Bar	Maximum 10 items	Static action
Pull-Down Menu	Maximum 12 items	Static action
Cascading Menu	Maximum 5 items, 1 cascade deep	Static action
Pop-up Menu	Maximum 10 items	Static action
Push-button	1 for each button, maximum of 6 per dialog box	Static action
Check Box	1 for each box, maximum of 10 to 12 per group	Static set/select value
Radio Button	1 for each button, maximum of 6 per group box	Static set/select value
List Box	50 in list, display 8 to 10 rows	Dynamic set/select value
Drop-down List Box	Display 1 selection in control at a time, up to 20 in a drop-down box	Dynamic set/select single value
Combination List Box	Display 1 selection in control at a time in standard format up to 20 in a drop-down box	Dynamic set/select single value; add value to list
Spin Button	Maximum 10 values	Static set/select value
Slider	Dependent on data displayed	Static set/select value in range

CHAPTER 3: METHODOLOGY

3.1 PROCEDURE IDENTIFICATION

For this project, the author decided to apply 3 major phase in Waterfall Model. This model provides sequential approach to software development which consists of planning, analysis and design.



Fig1: SDLC

Table 1: Activities involved in the project

Phase	Activity	Description
1. Planning	<ul style="list-style-type: none">- Proposal submission- Project scheduling- Problem analysis	<ul style="list-style-type: none">- to get the approval from the FYP Committee- to make sure that the project flow the timeline given- to analyze the problem statement as well as its appropriate solutions
2. Analysis	<ul style="list-style-type: none">- Initial investigation- Data gathering- Requirement analysis	<ul style="list-style-type: none">- to get the overall services in IT/IS department- to gather information about the project- to analyze the system requirement

3. Design	<ul style="list-style-type: none"> - Design flow - Design data structure for database - Design the application interface 	<ul style="list-style-type: none"> - to design a new flow of the system based on the current service - to develop data structure in order to create database of the system - to develop user interface that is available and accessible to the user
4. Implementation	<ul style="list-style-type: none"> - Developed application for system administrator - developed application for student - 	<ul style="list-style-type: none"> - the author developed functions for system administrator - the author developed log problems for a student
5. Delivery	<ul style="list-style-type: none"> - user manual - oral presentation - report submission 	<ul style="list-style-type: none"> - the author prepared step by step instruction for the user to use the application - the author presented to the examiners and supervisors - the author documented all the task done, findings and discussions

3.2 TOOL

3.2.1 Software

3.2.1.1 Oracle

Oracle is one of the most powerful databases. It provides functions to the users such as an Internet ready platform for building web based applications and a comprehensive suite of Internet enabled business applications.

3.2.1.2 PHP

PHP is used towards web based client-server development.

3.2.1.3 Adobe Photoshop 6.0

This software is used for image editing and for logo design.

3.2.2 Hardware

- PCs (both hostel and office).
- Servers (within the college).
- Printers.
- Floppy Disks

CHAPTER 4: EXPECTED RESULT AND DISCUSSION

4.1 INTRODUCTION

The author had discovered many findings through this period of time. Started from analyzing the user requirement until designing, the author kept communicates with the user. Here are some expected results to be discussed.

4.2 PROJECT WORKS

4.2.1 Research on Human-Computer Interaction

The author had defined the definition of HCI in the literature review. In this case, the author had focused on the principle of good graphical user interface (GUI). There were many factors contributed to the successful GUI. Important characteristic of good GUIs is speed, or more specifically, responsiveness. Appearance of speed for GUI can be achieved by having all field validations occur on a whole-screen basis instead of on a field-by-field basis. Basically, the successful GUI depends upon the skills of the user, it may be possible to design features into a GUI that give the power user the capability to enter each field of each data record rapidly. Table 1 gives the guideline of design controls.

4.2.2 Data Flow Diagram

Through the analysis phase, the author has design the system data flow diagram (DFD). This process is very helpful in integrating the database with the system. The author has breakdown the DFD until Level 3. Refer to Appendix 1-5.

4.2.3 Interview

In order to gather information for developing the system, the author has gone through the interview with the respective representative from IT and Media department. Basically, the questions are very helpful in gaining the user requirement and achieving the expected result at the end of this project.

CHAPTER 5: CONCLUSION

IT and IS department of University of Technology PETRONAS (UTP) has no specific system to track, monitor and handled that information required. IT and IS department has decided to develop a system that will help to solve the problem occurred.

The students and staff of UTP just need to log their problem to this online system and the solution of their problem will be displayed in the system. The students and staffs also may review the previous problems with their solution. This will increase the efficiency of the services provided by ITIS department

This system will consider the HCI element in implementing the user interface. The examples of HCI areas to be discovered are usability, accessibility and visibility. Although the author believes that there are many constraints in order to finish this project, the author will face them as a challenge.

For future recommendation, the author suggested that there will be a pager service in order for the technician to keep alert of new problem. However, it must be addressable via an email address. Other matter that can be suggested is to have the notification system that will keep track the information flow between the management and the student itself. Since the author did not have enough time to discover this service, hope that other student can continue implementing this project.

REFERENCES

- [1] What is E-Info? Meaning of E-Info. What does E-Info?
<http://www.thefreedictionary.com/E-Info>

- [2] Computing and Network Services Helpdesk
<http://www.ualberta.ca/HELP/>

- [3] Bates College: ILS: Learn
<http://abacus.bates.edu/ils/learn/index.html>

- [4] Gary B. Shelly, Thomas J. Cashman and Harry J. Rosenblatt, System Analysis and Design Fourth Edition, Shelly Cashman Series, 2001, Course Technology

- [5] Ovitz Taylor Gates Consulting, Helpdesk Ticket System, 2001
<http://www.helpdesksurvival.com/HelpdeskTicketSystem.html>

- [6] James Hobart, Principle of Good GUI Design, 2001
http://axp16.iie.org.mx/Monitor/v01n03/ar_ihc2.htm

- [7] Joel Spolsky, User Interface Design for Programmers, 2001, Apress

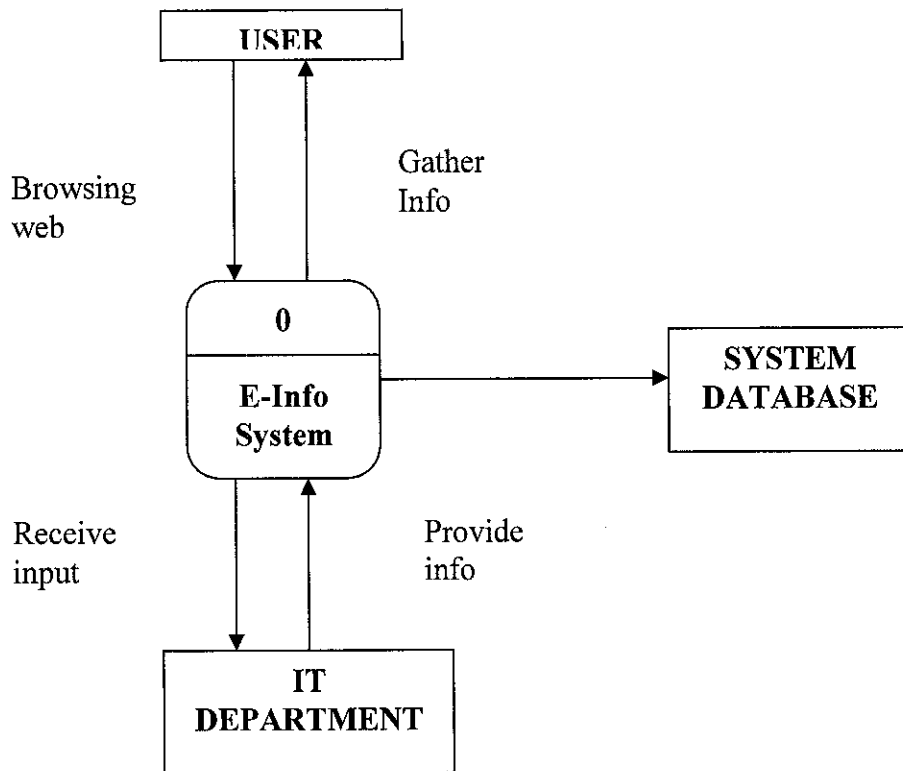
- [8] human-computer interaction - a whatis definition - see also HCI, 2000-2004, TechTarget, Inc
http://whatis.techtarget.com/definition/0,,sid9_gci213992,00.html

- [9] GUI definition of GUI_ What is GUI Meaning of GUI_ What does GUI mean GUI synonyms, GUI antonyms, 2004, Interapple, Inc
<http://www.thefreedictionary.com/GUI>

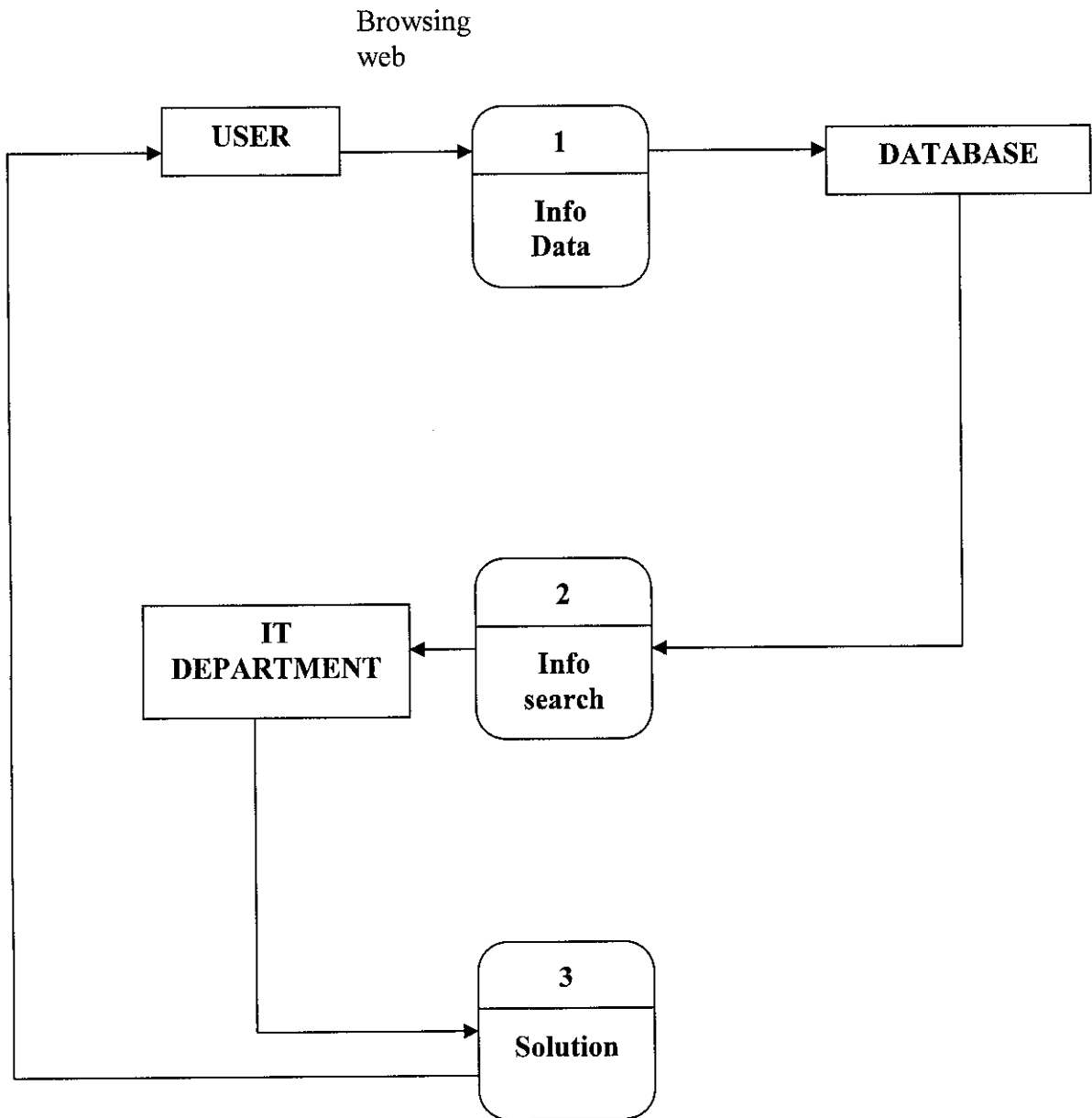
APPENDICES

**Appendix 1:
Context Diagram**

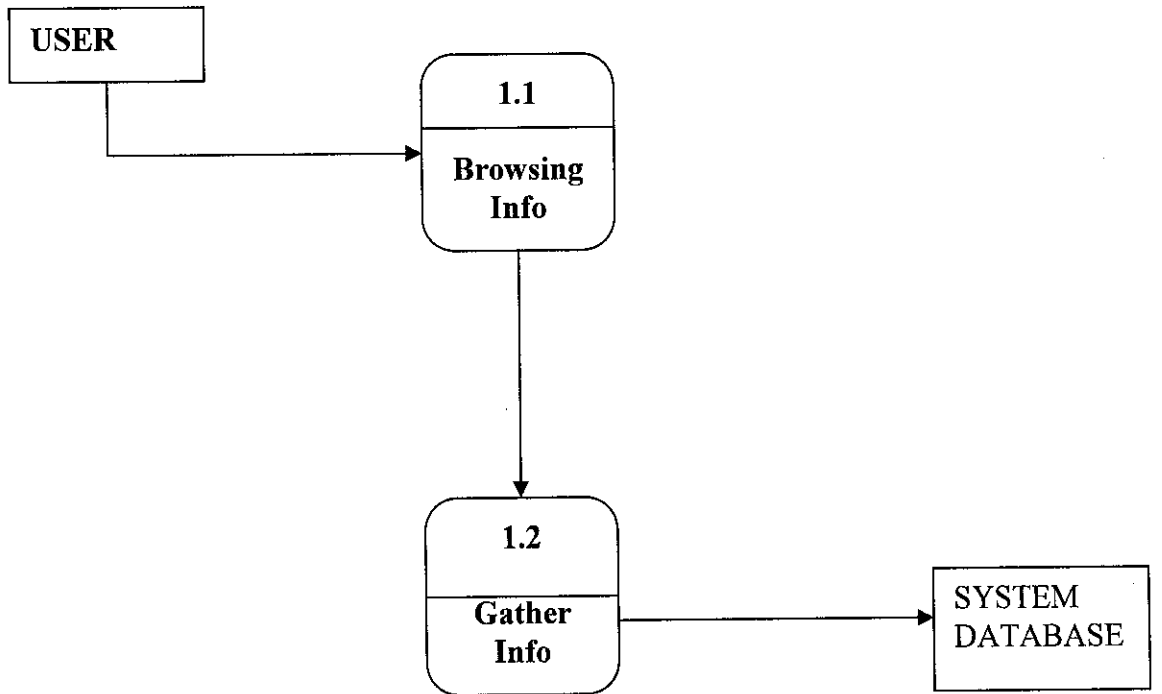
Appendix 2: E-Info System Context Diagram



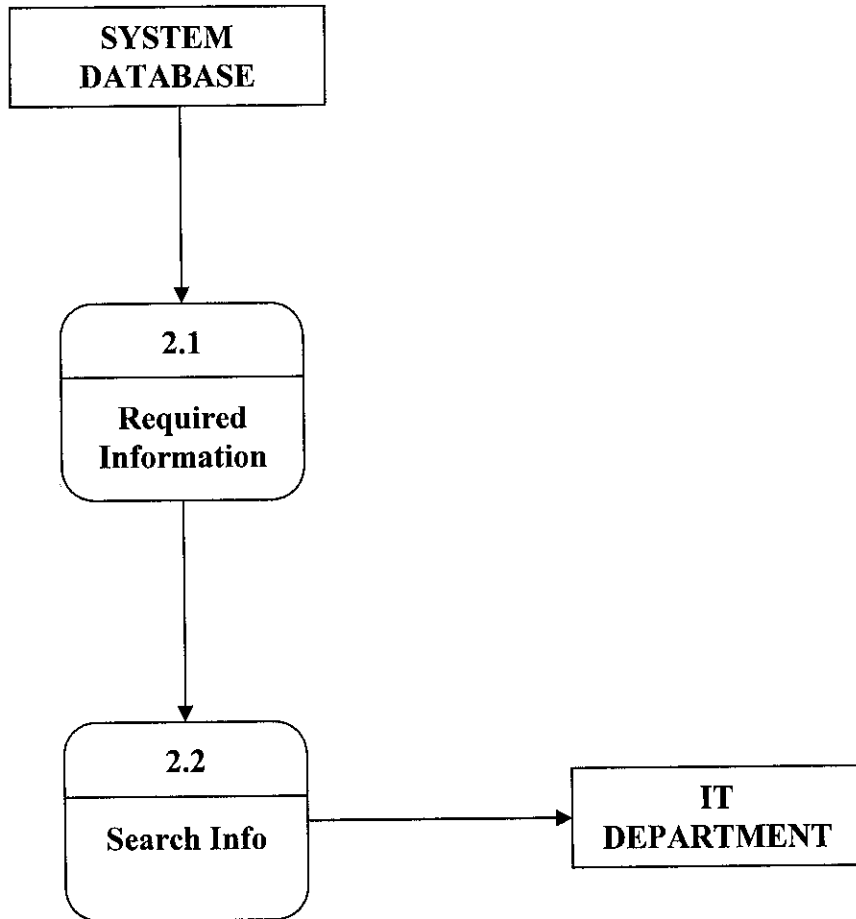
Appendix 3: DFD LEVEL 0



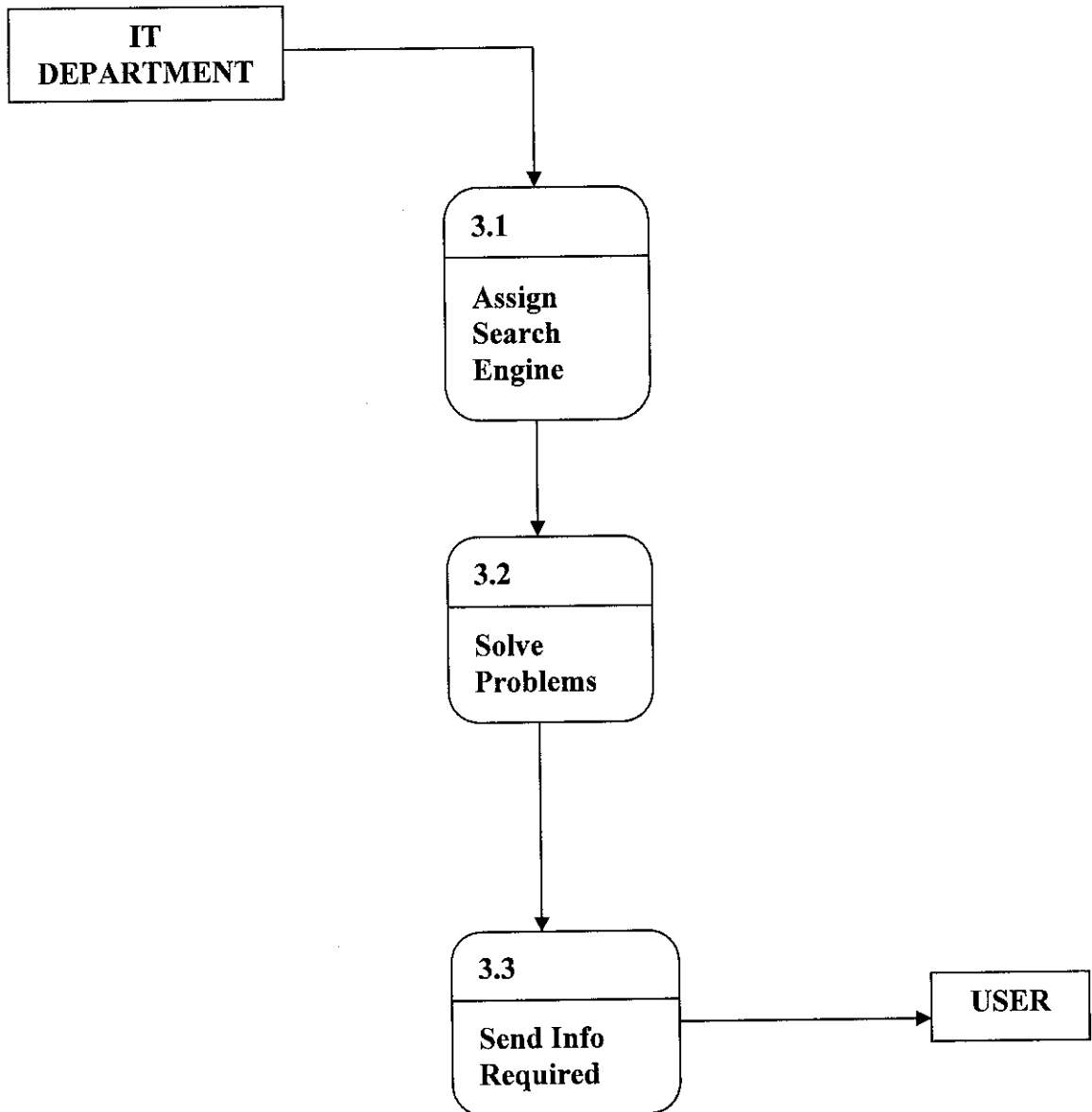
Appendix 4: DFD LEVEL 1



Appendix 5: DFD LEVEL 3



Appendix 6: DFD LEVEL 3



E-INFO SYSTEM

ID	Task Name	Duration	Start	December	January	February	March	April	May	June	July	
1	PLANNING	74 days	Mon 1/19/04	[Task bar spanning Dec 19 to May 19]								
2	Project title approval	1 day	Mon 1/19/04									
3	Project background of study and prc	2 days	Tue 1/20/04									
4	Data and information gathering	4 days	Thu 1/22/04									
5	Project preliminary report submissio	1 day	Thu 1/29/04									
6	Logbook submission	66 days	Thu 1/29/04									
21	Project timeline planning	2 days	Wed 2/4/04									
22	ANALYSIS	13 days	Wed 2/4/04									
23	User requirement analysis	3 days	Wed 2/4/04									
24	Interview and Questionaire	3 days	Wed 2/4/04									
25	System workflow analysis	5 days	Mon 2/9/04									
26	Design UML and DFD	5 days	Mon 2/9/04									
27	User interface proposal	5 days	Mon 2/16/04									
28	DESIGN	21 days	Mon 2/23/04									
29	Interface prototype development	7 days	Mon 2/23/04									
30	Database structure development	7 days	Wed 3/3/04									
31	System interface and database stru	7 days	Fri 3/12/04									
32	SUBMISSION	54 days	Tue 3/23/04									
33	System testing	5 days	Tue 3/23/04									
34	Integration testing	5 days	Tue 3/23/04									
35	Post testing questionnaire	3 days	Thu 3/25/04									
36	Supervisor final draft submission	4 days	Tue 3/30/04									
37	Final draft submission	1 day	Fri 4/9/04									
38	Oral presentation	7 days	Thu 4/22/04									
39	Project dissertation submission	0 days	Fri 6/4/04									

Project: E-INFO System

Task Split

Milestone Summary

External Tasks External Milestone