DESIGN OF A TRANSACTION PROCESSING SYSTEM FOR THE GRADUATE STUDENT RECORD DATA BASE

By

DAVID K. SCHOTTEL
B.A., PARK COLLEGE, 1974

A MASTER'S REPORT

Submitted in Partial Fulfillment of the

Requirements for the Degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

- 1982 -

Approved by:

Major Professor

ACKNOWLEDGEMENTS

I wish to greatfully acknoweledge and sincerely thank the following people, without whom this report would never have been completed:

My Major Professor and mentor, Dr Elizabeth A. Unger, a truely remarkable woman.

Dr Thomas L. Gallagher, whose assistance in the computing center proved invaluable.

My wife and daughter, whose love, patience and understanding helped me to persevere.

My Lord, without whom nothing is possible.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter I

INTRODUCTION

1.1 BACKGROUND

It has been recognized for some time that automation of the administrative tracking of Computer Science Graduate Candidates would enhance the responsiveness of the department to the needs of the student. This tracking process begins with an initial student inquiry, progresses through an approved program of study and research project and terminates upon successful completion of all requirements for graduation. Automating this tracking function will facilitate a more rapid response to student information needed for managing the many aspects of his academic career. As a result of this automation effort the ease of creating and maintaining student files as well as producing desired reports and letters should be greatly enhanced, reducing the department administrative workload.

1.2 DEFINITIONS

1. EDITOR

The editor of this transaction processing system is a suitability edit that verifies whether the data in a transaction meets the criteria established prior to the processing of the transaction.

2. MENU,

A menu is a list of items that presents choices to users of a system. A collection of menus used to present the functions of, or information about, a particular application program or software system is a menu-driven interface. (6)

3. SCREEN FIELD CODE

A screen field code is a unique identifier assigned to each field of each screen skeletal format. (12)

4. SCREEN FORMAT IDENTIFIER

A screen format identifier is a unique identifier assigned to each screen skeletal format managed by the screen formatter program. (12)

5. SCREEN FORMATTER PROGRAM

The screen formatter program is a component of the transaction processing system that drives the user terminal. In the context of this report it receives the desired user function through the

processor program, converts it to a screen format itentifier, retrieves the necessary screen formats for the function, and passes them to the processor program for display.

6. TRANSACTION

A transaction is a logical unit of information that is processed in its entirety before another unit is processed.

7. TRANSACTION PROCESSING SYSTEM

A transaction processing system is a system in which the mode of user computer processing lies between batch processing (no user control during execution) and interactive processing (complete user control during execution). Transaction processing limits the user to a certain set of resources and functions as batch processing does but still provides the on-line capability of interactive processing.

8. USER FRIENDLY

A system is said to be user friendly if it is easy to learn, easy to remember and easy to use. User friendliness was a key consideration beginning with the initial design and analysis of this project all the way through implementation.

1.3 OBJECTIVE

The objective of this report is to outline the design of a transaction processing system which will interact with the user via a series of menus, and interface with both the data base management system, INGRES, which is a relational based model, and the output processing system. The design philosophy and analysis approach will be presented as a basis for system definition. Problems encountered during the design will be discussed with recommendations for future enhancements outlined.

1.4 ENVIRONMENT

The Graduate Student Record System (GRASTURS), includes two distinct but identically formatted data bases as a key part of its environment. An active data base includes all students who are currently enrolled as graduate students in the computer science program. Once students have graduated, or have discontinued the program, they are purged to an inactive data base. The structure used in the development of these data bases was the relational model, with all relations in third normal form as verified by Bernstein's Algorithm Number Two. The data base resides on the Computer Science Department's Perkin Elmer, model 3220, minicomputer. The operating environment is the Bell Laboratories' UNIX operating system and version 6.3 of the INGRES Data Base Management System. Data base queries will

be formulated using INGRES data manipulation language, QUEL while all the application programs, and interfaces, will be written in Pascal. The transaction processor was designed to allow for a controlled number of input stations with various other stations capable of read only access and selected output generation. Multiple input stations were considered during the design and will be included in a future expanded system implementation.

1.5 METHODS

The design effort began with an analysis of the input and output documents used by both the Graduate School and the Computer Science Department to establish and maintain graduate student records. Initial menus were then developed based on user requirements and desired functions. With user friendliness a key ingredient throughout the design phase of the project, formatted screens were designed. User advice was solicited at every opportunity. Data fields within the screens were verified against the relations of the data base to insure that the data were being stored, and that their sequence coincided with the domain sequence of the relations. A suitability editor was designed with edit criteria established for each data field that appears on a formatted screen. Finally the transaction processor program was analyzed in terms of controlling the functions and procedures to facilitate menu storage, screen

formatting, data base interaction, input editing and output processor interface. This program acts as the driver, and ties all the components into a cohesive operational system.

1.6 PROBLEMS ENCOUNTERED

Perhaps the greatest problem encountered was visualizing how formatted screen data was to be entered into the established relational data base by using a Pascal program to interface with the INGRES data manipulation language, QUEL. Reversing this process and getting the same data out of the data base posed the other side of this problem. The author's lack of experience in using both the relational model and the INGRES data base management system contributed greatly to this problem. The unavailability of intelligent terminals for use by the system added some additional problems. This meant that protection for fields on the formatted screens had to be written into the software as opposed to utilizing a built in hardware feature of an intelligent terminal. Further, it meant that system (UNIX) files had to be built for data entered on the various screens instead of using storage within the terminal. Finally, keeping user friendliness constantly in the fore throughout the design process, sometimes at the expense of simplicity and ease of future programming, was a constant concern.

1.7 REPORT OUTLINE

The remaining chapters of this report detail the design and functions of the transaction processing system. In chapter two, system interfaces with the user, the data base, and the output processor will be explained as well as system security and transaction editing. Chapter three outlines the system functions available to the user. In chapter four the implementation is analyzed, discussed and critiqued. Chapter five summarizes the report and uses this as a basis for specifying future system enhancements.

Chapter II

GENERAL DESIGN

2.1 INTRODUCTION

In designing the transaction processing system, a modular approach was adopted. After a good definition of the problem, the first area to be considered was the security of the system and all the implications of access control and data integrity. Specifics were outlined regarding the types of users and their required system access, with a logon procedure developed to support these requirements. System functions for the user to add, maintain and use data within the data base, and how the user interfaced with these functions, was next considered. Analysis of the data elements required to be maintained on graduate students and how best to have the user input these data elements, via formatted screens, was the concern here. As user friendliness was the overriding consideration in designing this system, a great deal of analysis effort was devoted to screen layouts, prompts and aids, and user instructions. In considering the next logical area, both the INGRES data base management system and the relations with their domains were analyzed to insure that the transfer of data from screen to data base and back again could be accomplished. Related to

this problem was that of interfacing with the output processor so that formatted screens collected the necessary data to provide the required outputs. Finally, edit criteria were established for data entered into the system to insure a degree of accuracy for the data base.

Designing the system in terms of these logical entities allowed the author greater flexibility in adapting to change without having to redesign the entire system. Further, as bottlenecks arose in one area, there was very little difficulty in shifting effort to another to maximize time utilization and design effort. In the sections which follow, specifics regarding the design of each of the logical entities will be presented.

2.2 SECURITY

In looking at the security requirements for both the system and the data base, the following factors were considered:

1. Who are the primary users?
2. What level of access is required for each?
3. How is security to be provided?
4. Is a transaction audit trail required?

In answering these questions, two types of users were defined as requiring access to the system. The first type, in the majority, were the professors and departmental administrators who require student information for academic

counseling, rewiew and analysis. It was determined that the requirements for these users involved viewing student information and outputting status reports, with very little need to change any of the data. Based on these requirements, users of this type were provided a read only access capability along with an ability to produce summary type outputs.

The second type of user was one requiring total access to the system and the data base. These users include the data base administrator and selected administrative personnel. Total access to system information was required as it is their responsibility to create and maintain the data base as well as use it to produce departmental letters and student information reports.

Having identified the users and the extent of their access, logon procedures were developed and an access facility built which provided for the degree of security necessary to meet the specified requirements. A layered approach (See figure 2.1) was adopted. In this approach a user first logs onto the UNIX system by entering his user identification and password. If he is a valid system user, as verified by the UNIX users file, the logon will be accepted. The next layer of security is invoked when the user attempts to enter the Graduate Student Record System (GRASTURS). The user enters the system name, GRASTURS, which invokes the system transaction processing program. This

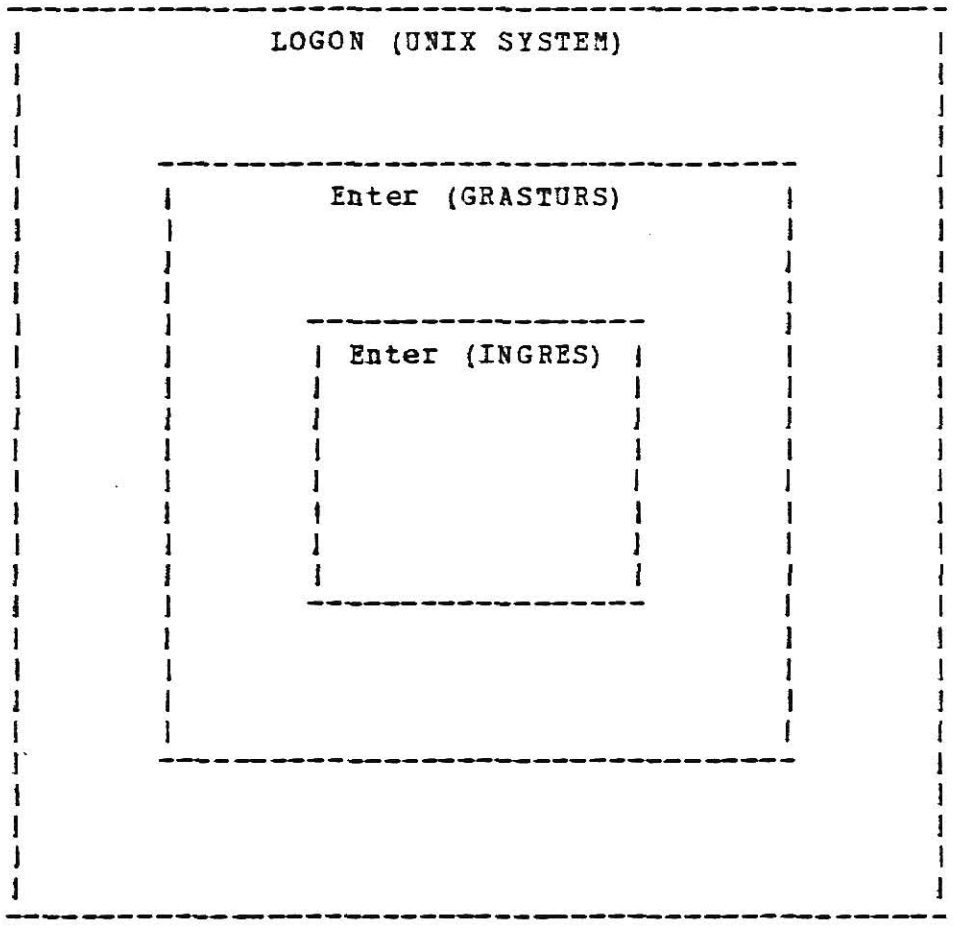


FIGURE 2.1: SYSTEM SECURITY

program maintains a second users file containing the passwords, user identifications, and level of access (read only, read-write) of all personnel authorized to access GRASTURS. Should an individual not be authorized access, a system message will be returned so stating. Once a valid user has entered the Graduate Student Record System, a second program, the data base interface program, logs the user onto the INGRES data base management system. Here the third level of security is implemented, in that the data base administrator, through the use of the INGRES "Permit" command, specifies the level of access at the time of data base creation. This provides an extra degree of security in that, even if an unauthorized user were to breach the application program level of security, he would still have to breach the data base management system level of security. It should be cautioned that both the GRASTURS users file and the "Permit" commands granting access must stay in synchronization if system security and data base integrity are to be insured.

Analysis indicated a transaction audit trail was desirable but beyond the scope of this report. It is envisioned, however, that this audit trail would record and maintain, for some specified period of time, the date, time, user ID, and screen field code of all data items that were changed during a session. It would also record and maintain the date, time, and user ID of all unauthorized attempts at

accessing the data base. These transaction audit trails could then be reviewed periodically by the data base administrator to help determine if an unauthorized user had attempted entering, or in fact had entered the system.

It should be emphasized that GRASTURS has only a minimum level of security; there is a great deal of room for improvement. As time and resources permit a more detailed and extensive security subsystem should be develcped to encompass both system security and data base security needs.

2.3 USER INTERFACE

The user interface is where human and machine meet. As Schofield (10) states, the user interface consists of "all messages that can pass between the user and the machine and the conditions under which they can cccur". As previously mentioned, one of the goals of this project was to design this interface so that it would be as user friendly as possible. Unfortunately the attributes of user friendliness cannot be as accurately defined as the interface itself. This is true because we all have different opinions as to what exactly makes an interface friendly. For this report, the friendliness of an interface will be the relative ease by which an individual can use the interface without referring to a users' manual or some other non-machine aid.

A menu-driven approach was adopted as the design philosophy tc maximize the interface friendliness. That is,

all operations are invoked through the prompting that is provided by menus. As a result, the operator is required to recall very little to perform an operation when the options at any particular stage are presented on the screen. Aderet (11) indicates, regarding menu-driven systems, that "The greater the degree of prompting, the less the long-term memory requirements; hence it may be predicted that menu-driven systems should be of benefit inasmuch as they are likely to ease cognitive overload for new operators, decrease learning time, reduce error and increase product acceptability".

The transaction processor program (See figure 2.2) is the key program in the transaction processing system that interfaces with the user. It is responsible for logging the user onto the system and verifying the level of system access. It responds to the user by displaying menus, prompts, outputs, and error messages in order to allow him to complete his processing as easily and quickly as possible. It controls the screens displayed on the terminal by passing to the screen formatter program the function selected by the user. The formatter program converts this function to a screen format identifier, which is used to locate the desired screen. The formatter program then passes the screen back to the processor program for display on the terminal.

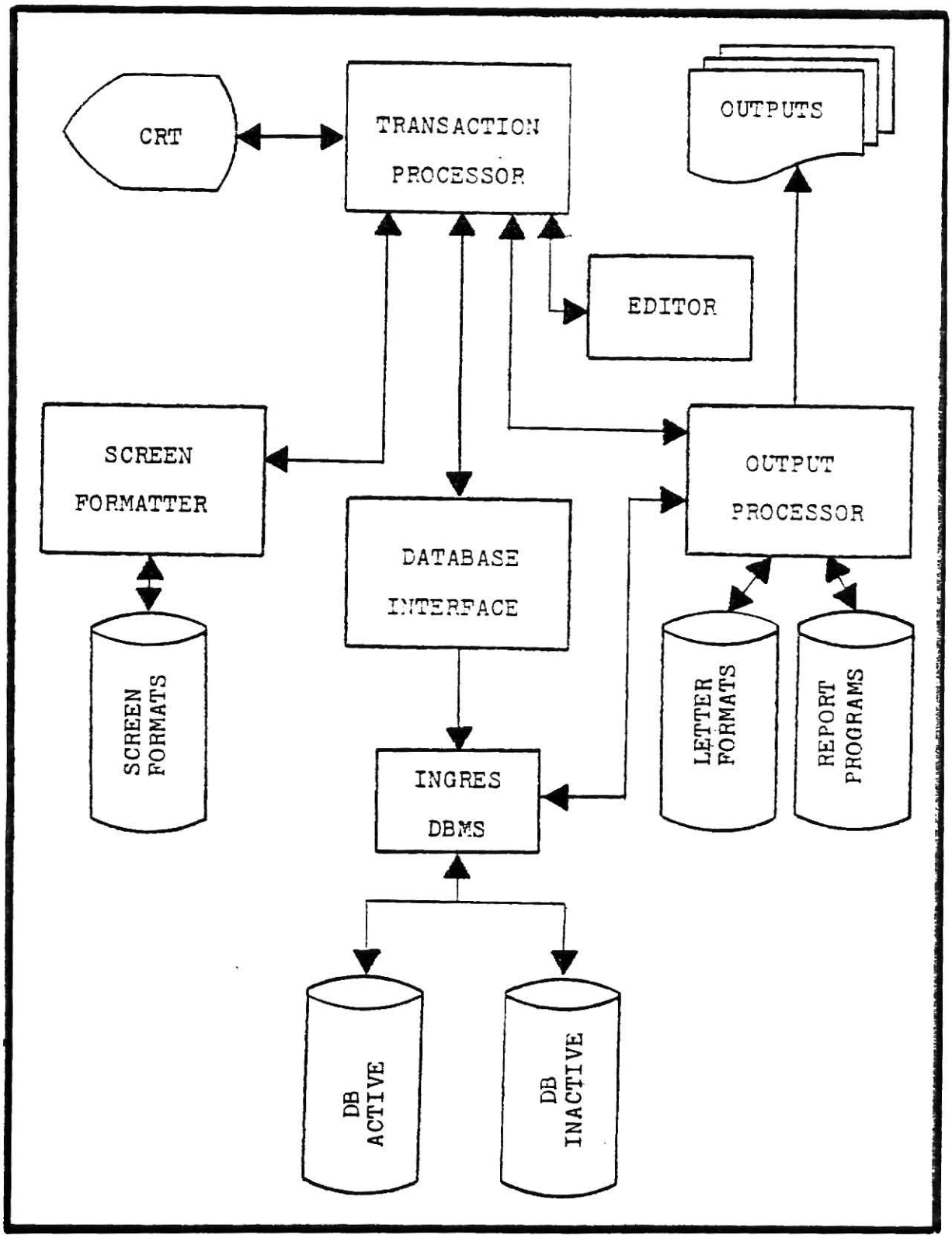


FIGURE 2.2: SYSTEM DIAGRAM

In designing the screen formats for the various system functions several factors were taken into consideration. The first, and probably the most important, being the experience of the read-write users. The experience of the read only users was only marginally considered as their access, and therefore the functions they can perform, was limited. The read-write users' responsibility for data base maintenance made their experience factor paramount. Analysis indicated that the experience of these users was at the mid-novice level. With this in mind, prompts and aids were developed to explain and walk the user through a session. It should be pointed out, however, that the screens were developed so that as a user gained experience on the system these same prompts would not slow him down with trivial questions and unecessary instructions.

Instructions on the system in general, and the specific functions within the system, were provided as a separate function which the user could ask for. This eliminated the need for lengthy "how-to" instructions on each screen. The level of prompts used on all screens was the fill-in-the-blank prompt. In some instances a command and a list of its parameters are presented, where the user selects the parameter he desires and enters it in the space provided. In others, a format is provided into which the user enters the specific data requested. In all cases an underlined area was provided for a required data entry eliminating confusion as

to where data was to be entered, and in many cases, how data was to be entered.

Another factor considered was the ease of reading a screen full of data within a format. In this case, a screen would tend to be very busy making it difficult for users to discern data from format. This problem can be overcome by varying the light intensity of the input data. By making it of higher intensity than the screen format it would become much easier to read and use.

Still another factor considered was error handling. In all cases error messages were designed to be as descriptive as possible. They are highlighted and are displayed at the same location on every screen. This should make it easier for the user to recognize that an error has occured and to determine what that error is. Error handling was made as user friendly as possible in that the system does not crash or lock the keyboard, or ring bells when an error condition is encountered. Error recovery procedures are described in the system explanation function and in most cases all that is required is a change to the entry in question for the system to continue processing. If, however, the user continues to get deeper into trouble with each attempted correction, an escape mechanism has been provided on every screen. This mechanism will allow him to terminate either the function he is presently in or the system altogether, without damaging the data base.

Finally, in considering screen design, the location of data items and their layout on the screens was analyzed. Users were consulted to determine which source documents would be entering the system first. It was the data items from these documents that were placed together on the first screen. This approach prevailed throughout the design. Data items which logically occurred together or came from the same source document were kept together. Further, data that could be displayed in tabular form was placed in tables to make it easier for the user to read and use.

Associated with each data item location on a screen is a screen field code. The use of this code facilitates the positioning of the cursor at the next data item location to be filled with data. This is accomplished by depressing the carriage return. This code also allows the user to access certain frequently changed data fields directly, without having to scroll through a series of screens to locate them.

To summarize, it's important to note that the design of a user friendly interface is still an art and not an exact science, requiring the consideration of many factors. In this project as many of these factors as possible were incorporated into the interface. Any future implementation should not consider these same factors as being all inclusive, but should readily make changes to them as the rquirements of the user so dictate.

2.4 DATA BASE AND OUTPUT PROCESSOR INTERFACE

Having looked at how the user interfaces with the system, we turn now to the various processes within the system and how they interact and interface with both the data base and the output processor. To facilitate a clear understanding of this process these interfaces will be described at the functional level.

The transaction processor program not only interfaces directly with the user but acts as the driver for the entire Graduate Student Record System. Depending on the action requested by the user, this program transfers control to the appropriate program for processing. Results are then passed back through the processor program for display on the terminal. In the case of output processing, this program acts as the interface between the transaction processing system and the output processing system and transfers control to the output processor whenever the user selects the output function or desires a printed copy of the current screen. The user selected output function, keys the processor program to transfer control to the output processor program for the processing of a departmental letter or report. The requested output is then passed back through the processor program and is displayed on the screen. Should the user desire a printed copy of the output, or any screen currently being displayed, a PRINT action is selected which tells the processor program to call the

output processor for printing of the output. Once the output has been routed to the printing device, control is again passed back to the transaction processor program. The key design issue here is that all requests for printed output as well as any department letters or student information reports are transfered to the output processor program for processing.

The data base interface program, as its name implies, is the transaction processing system's interface with the INGRES data base management system. It is responsible for logging the user onto the INGRES system, formatting the input data and generating the INGRES commands to add, change, or delete data from either the active or inactive data base. This is where the bulk of the work of the system is performed and is probably the most complex program in the system. When data is to be entered into the data base the processor program transfers an edited screen of data to the interface program. The interface program then builds a UNIX system file for each relation on the screen. Semicolons separate the data items on the screen which correspond exactly to the domains within the relations of the data base. The special character "#" is used to signify the end of a file which corresponds to the end of a relation. When the UNIX files are built all special characters are removed before data is entered into the data base. Once all the screen data has been moved into UNIX files a series of

INGRES "Copy From" commands are generated by the interface program to append the data to the appropriate data base. The "Copy" command is somewhat simplified by having the sequence of the data items on the screen map exactly to the domains of the data base.

If the user wants to view the data on a student, the reverse of this process is executed. The processor program passes the name of the student along with the desired screen format to the interface program. The interface program then generates a series of INGRES "Retrieve All" commands which move the data from the appropriate data base into UNIX system files. The interface program then moves the data from the UNIX files into the formatted screen and passes it to the processor program for display. Should the name not be found in the requested data base, an error message to that effect is generated and passed back to the processor program for display.

An important consideration in the design of this program is its compatibility with the INGRES data manipulation language, QUEL. A thorough understanding of the language and all its facilities is necessary in implementing the interface program if the system is to function properly. Further, the building of UNIX system files is a key design issue which facilitates the mapping of an entire relation into the data base using the "Copy" command, and out of the data base using the "Retrieve All" command. Had an

alternative approach been taken, that of appending data to the data base using the "Append" command and retrieving data from the data base using only the "Retrieve" command, this process would have been far more complex. For example, to append data to an existing relation , each of the domains and their value would have to be specified. i.e.

Append To Donation(Name="Frank",Amount=5,Ext=204)

An "Append" command of this nature would have to be generated for each relation in order to enter data into the data base. The problem is compounded when one realizes that one of the relations in the data base has thirty-one domains. Using the "Retrieve" command to access existing data also proves to be somewhat complex. i.e.

Retrieve (P.Amount,P.Ext) Where P.Name="Frank"

Again, a "Retrieve" command similar to this would have to be generated for each relation in the data base, with all domains specified.

Compare the complexity of these two commands to the simplicity of the "Copy" command using UNIX files for adding data to the data base:

Copy Donation () From "the UNIX file"

and the "Retrieve All" command for retrieving data from the data base:

Retrieve (All) where P.Name="Frank".

There would still have to be a "Copy" and "Retrieve All" command for each relation in the data base, but the commands

are no where near as complicated. The approach taken simplifies the programming requirements and should, when implemented, reduce processing time.

2.5 EDITOR FUNCTION

The final program to consider in the transaction processing system is the edit program, which is invoked any time data is to be saved in the data base or an output letter or report is to be printed. The edit is a suitability edit that verifies whether the data entered meets the criteria established for reasonableness. The design focus was one of simplicity and data format (syntax) as opposed to data content (semantics). The edit program maintains a table of screen field codes with corresponding edit criteria. If a data value is entered which does not meet the suitability criteria for it, an error message and the screen field code of the data item is passed to the processor program. The processor program then positions the cursor at the screen field code location which corresponds to the data item and displays the error message, "ERROR AT CURSOR". The user has the option to correct the data item and reinitiate either the printed output function or the SAVE action once again. This new data value is again edited and the process continued until a screen of data meets all the edit criteria. Once it does, the edit program indicates this result to the user and control is passed back to the

processor program for further processing of the screen by either the output processor or the interface program. Tables 1-7 outline the edit criteria for each data field and list the screen field codes and field lengths. Though not listed as part of the edit criteria, all data items, with the exception of the name field, may also be blank.

As was discussed earlier, the key design issue for this program was one of simplicity while at the same time providing some assurance of data integrity. The edit program is an area that can be expanded to provide a more stringent suitability edit or even a compatability edit should experience indicate the integrity level of the data base is inadequate.

TABLE 1

EDIT CRITERIA I

DATA ITEM	LENGTH	CODE	CRITERIA
Name	35	1	A
last	15		A
first	10		A
middle	10		A
SSAN	9	2	A/N
Current Addr	47	3	A/N
street	25		A/N
city	15		A
state	2		A
zip	5		N
Permanent Address	47	4	A/N
street	25		A/N
city	15		A
state	2		A
zip	5		N
local Telephone Num	7	5	N
Sex	1	6	M, F
Date KS Residency Began	5	7	A/N
month	3		A
year	2		N
Citizenship	10	8	A
Previous Name	15	9	A
Grad Sch Application Date	7	10	A/N
day	2		N
month	3		A
year	2		N

TABLE 2

EDIT CRITERIA II

DATA ITEM	LENGTH	CODE	CRITERIA
GRE Scores:			
official	1	11	Y,N
verbal	3	12	N
quantitative	3	13	N
analytical	3	14	N
Advanced GRE Score:			
official	1	15	Y,N
field	9	16	A
score	3	17	N
TOEFL:			
official	1	18	Y,N
score	3	19	N
date	5	20	A/N
month	3		A
year	2		N
Foreign Student Requirements			
finance	1	21	Y,N
medical	1	22	Y,N
Letters Of Recommendation	1	23	1,2,3
Curriculum	5	24	A
Advisor	15	25	A
Program Entry Date	5	26	A/N
month	3		A
year	2		N

TABLE 3

EDIT CRITERIA III

DATA ITEM	LENGTH	CODE	CRITERIA
Expected Grad Date	5	27	A/N
month	3		A
year	2		N
Proq Of Study Approved	5	28	A/N
month	3		A
year	2		N
CMPSC Info Sheet Date	7	29	A/N
day	2		N
month	3		A
year	2		N
CMPSC Application Date	7	30	A/N
day	2		N
month	3		A
year	2		N
Processing Fees Paid	1	31	Y,N
Admission Request:			
Fall	1	32	A
Spring	1	33	A
Summer	1	34	A
year	2	35	N
Type Admission Request:			
MS	1	36	A
PHD	1	37	A
SPEC	1	38	A

TABLE 4

EDIT CRITERIA IV

DATA ITEM	LENGTH	CODE	CRITERIA
Student Status (Can be replicated four times)			
date	4	39	A/N
regular	1	40	A
provisicnal	1	41	A
special	1	42	A
probation	1	43	A
Universities Attended (Can be replicated four times)			
school	25	59	A
dates (from-to)	10	60	A/N
month	3		A
year	2		N
month	3		A
year	2		N
major	5	61	A
GPA	3	62	N
scale	4	63	N
degree	5	64	A
degree date	5	65	A/N
month	3		A
year	2		N

TABLE 5

EDIT CRITERIA V

DATA ITEM	LENGTH	CODE	CRITERIA
TA Qualified To Teach:			
CMPSC 200	1	87	A
CMPSC 201	1	88	A
CMPSC 202	1	89	A
CMPSC 203	1	90	A
CMPSC 204	1	91	A
CMPSC 205	1	92	A
CMPSC 206	1	93	A
CMPSC 300	1	94	A
CMPSC 305	1	95	A
CMPSC 306	1	96	A
CMPSC 405	1	97	A
CMPSC 410	1	98	A
CMPSC 420	1	99	A
CMPSC 430	1	100	A
CMPSC 450	1	101	A
CMPSC 560	1	102	A
CMPSC 580	1	103	A

TABLE 6

EDIT CRITERIA VI

DATA ITEM	LENGTH	CODE	CRITERIA
Research Option:			
thesis	1	104	A
report	1	105	A
paper	1	106	A
Proposal Submit Date	5	107	A/N
month	3		A
year	2		N
Proposal Approved	1	108	Y,N
Title	50	109	A/N
Orals Date	5	110	A/N
month	3		A
year	2		N
Orals Result:			
pass	1	111	A
fail	1	112	A
conditional pass	1	113	A
Advisory Committee:			
major professor	15	114	A
member	15	115	A
member	15	116	A

TABLE 7

EDIT CRITERIA VII

DATA ITEM	LENGTH	CODE	CRITERIA
Graduate Courses (Can be replicated 16 times)			
course number	9	117	A/N
course name	20	118	A
credit hours	1	119	N
grade	1	120	A
semester/year	4	121	A/N
deficiency	1	122	A
repeat	1	123	A
major	1	124	A
support	1	125	A
Current GPA	3	261	N

Chapter III

FUNCTIONS

3.1 INTRODUCTION

The implementation of the functions of data base load, maintenance and use centers around a general screen menu of options. These options allow the user to select prompts or aids to generate transactions, to store transactions in the data base and to obtain desired retrieval results. With these functions constituting the prime user-system interface, it was important that user friendliness become the overriding design philosophy in development. Screens were designed for ease of function selection; data entry, modification and deletion; and output generation. Initial screens of each function give a short explanation as to what the function does, with some instructions provided on expected user response. A single screen transaction must be completely processed independently of any other screen transaction and a single transaction cannot be processed against multiple records of the same type. The specifics of the functions and how the user interacts with each one are detailed in the sections which follow.

3.2 GENERAL MENU

Once a user has properly logged onto the system the first formatted screen that appears is the General Menu. (See figure 3.1)

The General Menu gives a short explanation regarding the system and then lists the following six system functions:

1. System Explanation
2. Add A Student
3. Change/View Student Information
4. Purge A Student To Inactive Data Base
5. Delete A Student
6. Outputs

The user selects the function he desires and types the item number into the space provided. This will then display the first formatted screen of the selected function. If the user has been granted read only access he may only select the System Explanation, Change/View Student Information, or Outputs functions. If an invalid item number is selected, an error message to that effect will be displayed. Recovery is accomplished by entering a valid item number or exiting the system.

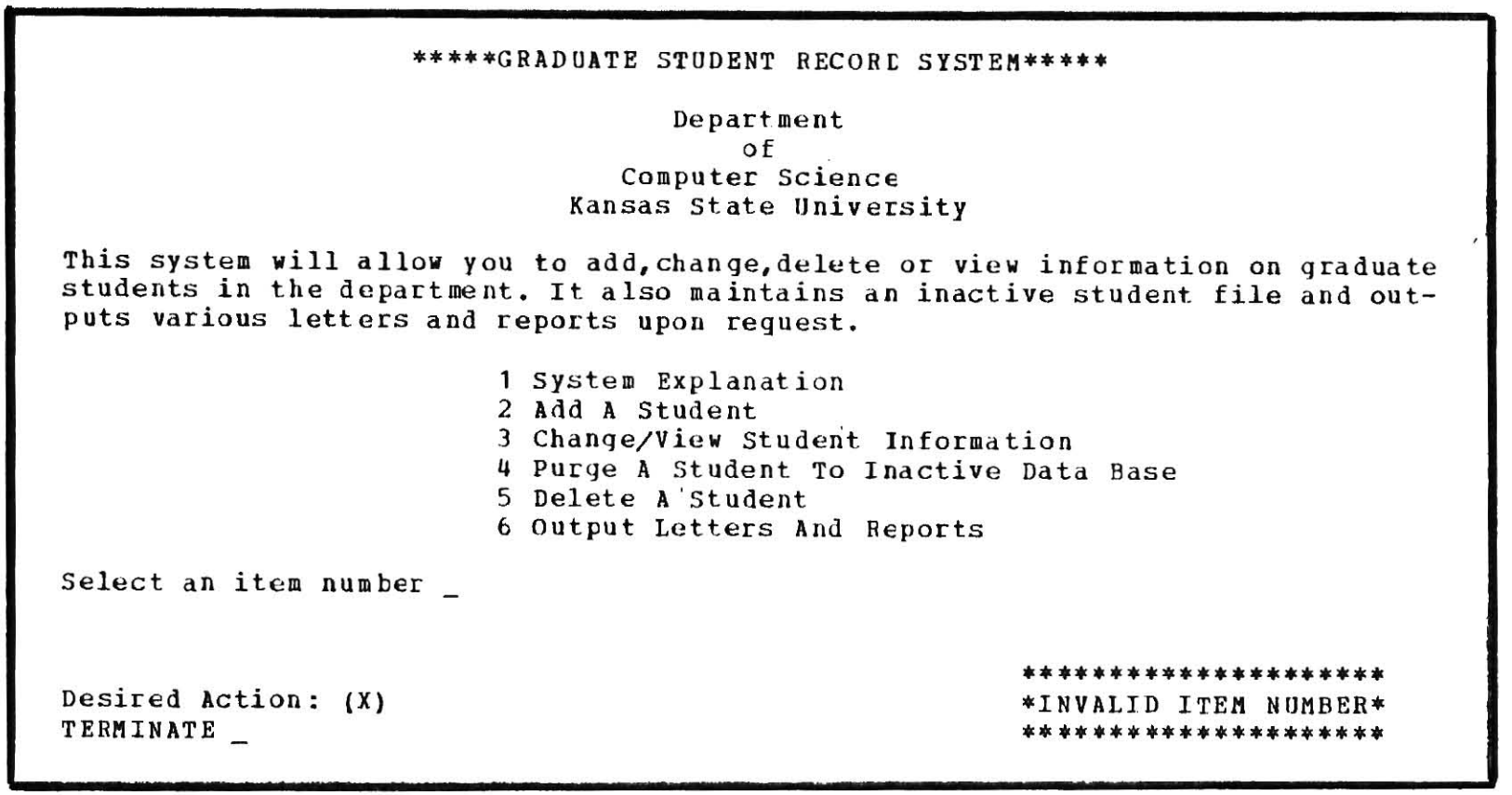


FIGURE 3.1: GENERAL MENU

3.3 SYSTEM EXPLANATION

The first item on the General Menu is the System Explanation function. It is envisioned that the new user, after logging on the system, will make this his first selection. This function will display a series of screens of textual information describing the system in general and each function specifically. (See figures 3.2-3.5) Though not intended to replace a user's manual, much of the information included in such a manual has been incorporated into this function. To scroll to the next screen, the VIEW action in the DESIRED ACTION portion of the screen is required, and it may be selected by placing an "X" in the space provided. The screens will continue to scroll as long as the VIEW action is selected. When the user wants to terminate this function the QUIT action should be selected. This action will return him to the General Menu for further processing. The TERMINATE action will terminate the function and allow the user to exit the system.

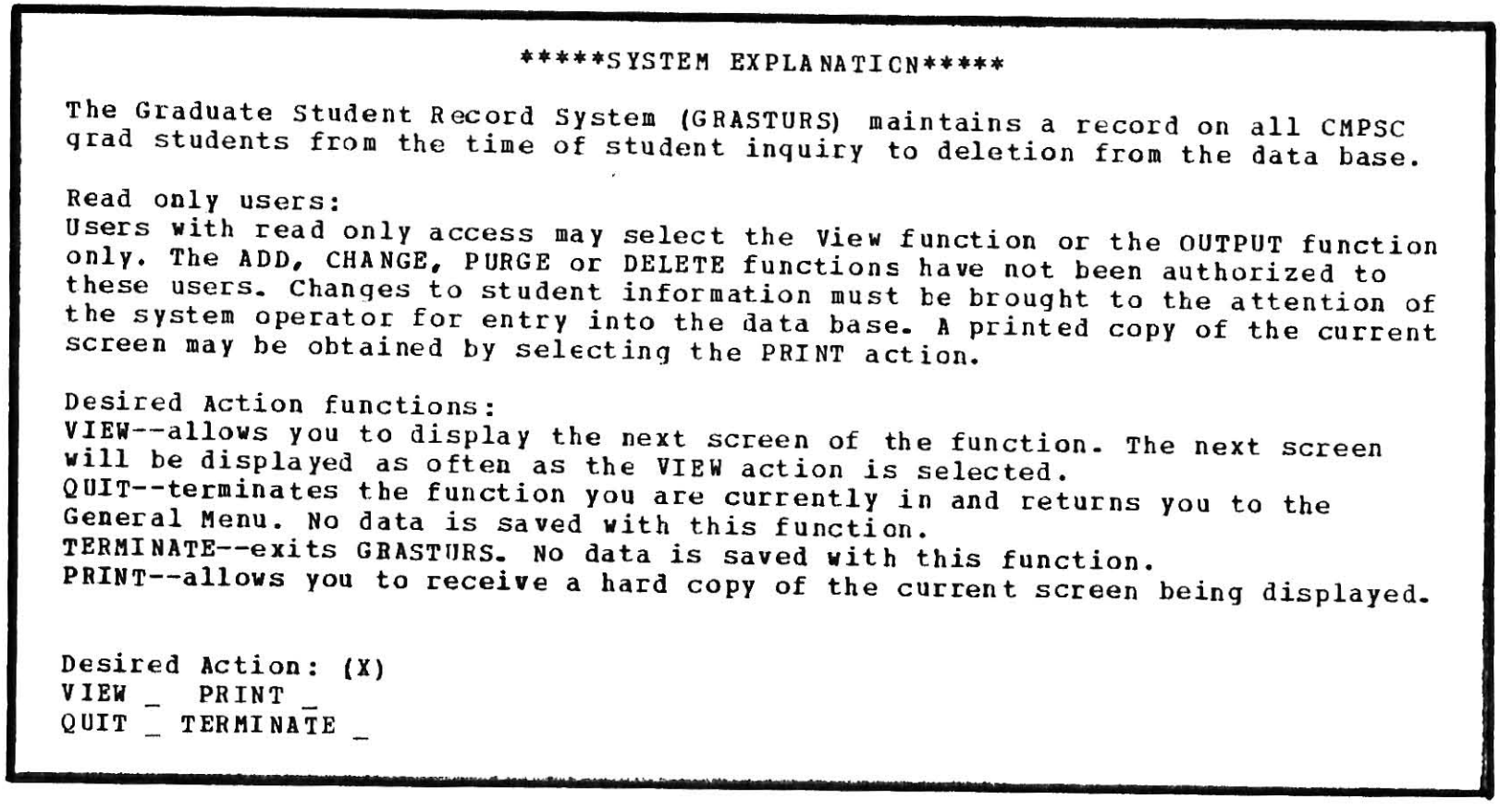


FIGURE 3.2: SYSTEM EXPLANATION

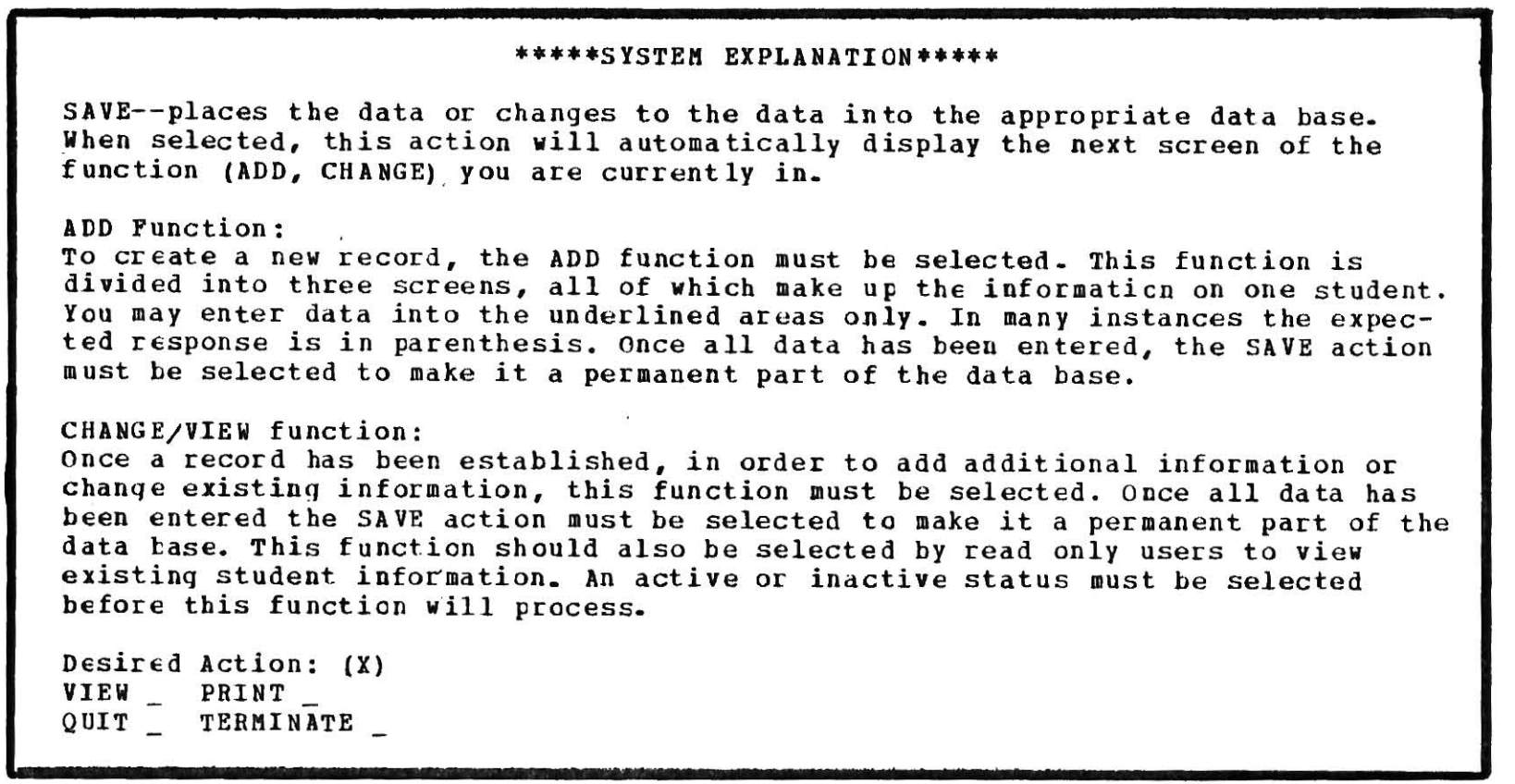


FIGURE 3.3: SYSTEM EXPLANATION

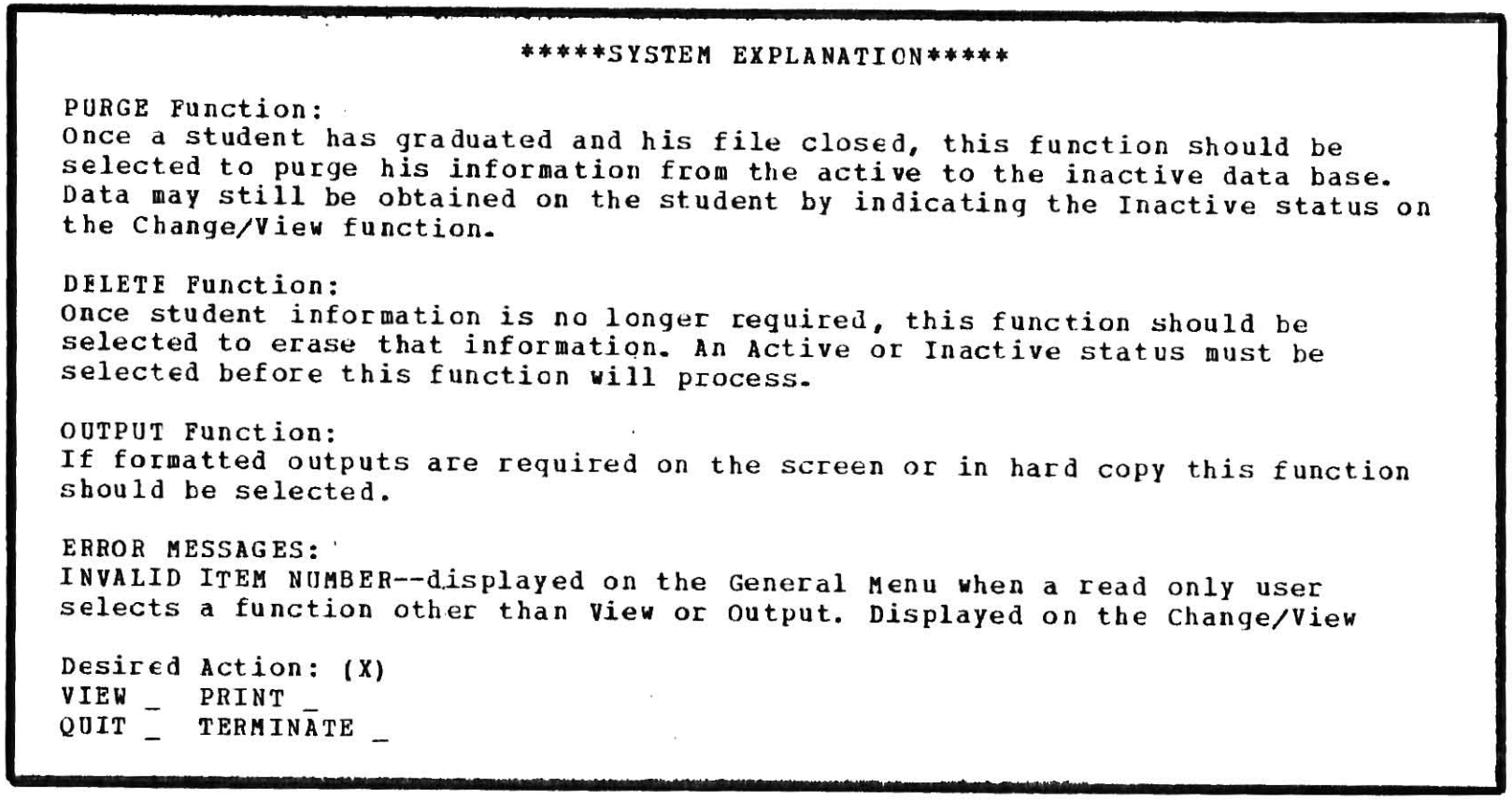


FIGURE 3.4: SYSTEM EXPLANATION

*****SYSTEM EXPLANATION*****

screen when an item number is selected that is not displayed. To recover, enter the appropriate item number.

ERROR AT CURSOR--displayed after a SAVE action on the Add and Change screens when invalid data has been entered into the data fields. The cursor will be positioned at the data field needing correction. To recover, correct the data and select the SAVE action again.

NAME NOT FOUND IN ACTIVE/INACTIVE D.B.--displayed when the incorrect status (Active/Inactive) was selected, or the name as the data base knows it was not entered. To recover, check the spelling of the name and the status and re-enter the request.

Desired Action: (X)
VIEW _ PRINT _
QUIT _ TERMINATE _

FIGURE 3.5: SYSTEM EXPLANATION

3.4 DATA BASE ADD TRANSACTIONS

The data base load is accomplished through the Add function which must be selected initially to create a new record for each student. The function is divided into three formatted screens which make up all the information on a student required for the data base. (See figures 3.6-3.8) Protected fields on the screens provide an outline for entering data with expected data formats often times outlined within parenthesis. The user responds by entering data into the underlined areas only, and then hitting the carriage return which automatically spaces to the next data field. Once the user has finished entering data, the SAVE action in the DESIRED ACTION portion of the screen must be selected. This initiates the data load process. If the data passes the suitability edit, it is mapped into the data base and the next screen of the function is displayed. If not, an "ERROR AT CURSOR" message is displayed and the cursor is positioned at the data field in question. The data field must then be corrected and the SAVE action selected again. Several other options are available in the DESIRED ACTION portion of the screen. First, if the user desires a printed copy of the screen, and its data, the PRINT action may be selected. This action will print, without destroying the data, as many copies as the number of times the action is selected. Second, as discussed in the System Explanation

section, if the user desires to terminate the function, the QUIT action may be selected. This will return him to the General Menu without saving any data to the data base. Finally, to terminate the function and exit the system the user may select the TERMINATE action. It is important to reiterate that the only way data is entered into the data base is through the SAVE action, and then only after it has passed the edit. The general format and procedures for the remaining two screens of the Add function are the same as the first, with the error process and desired actions identical. The SAVE action on the final screen, however, will return the user to the General Menu for further processing.

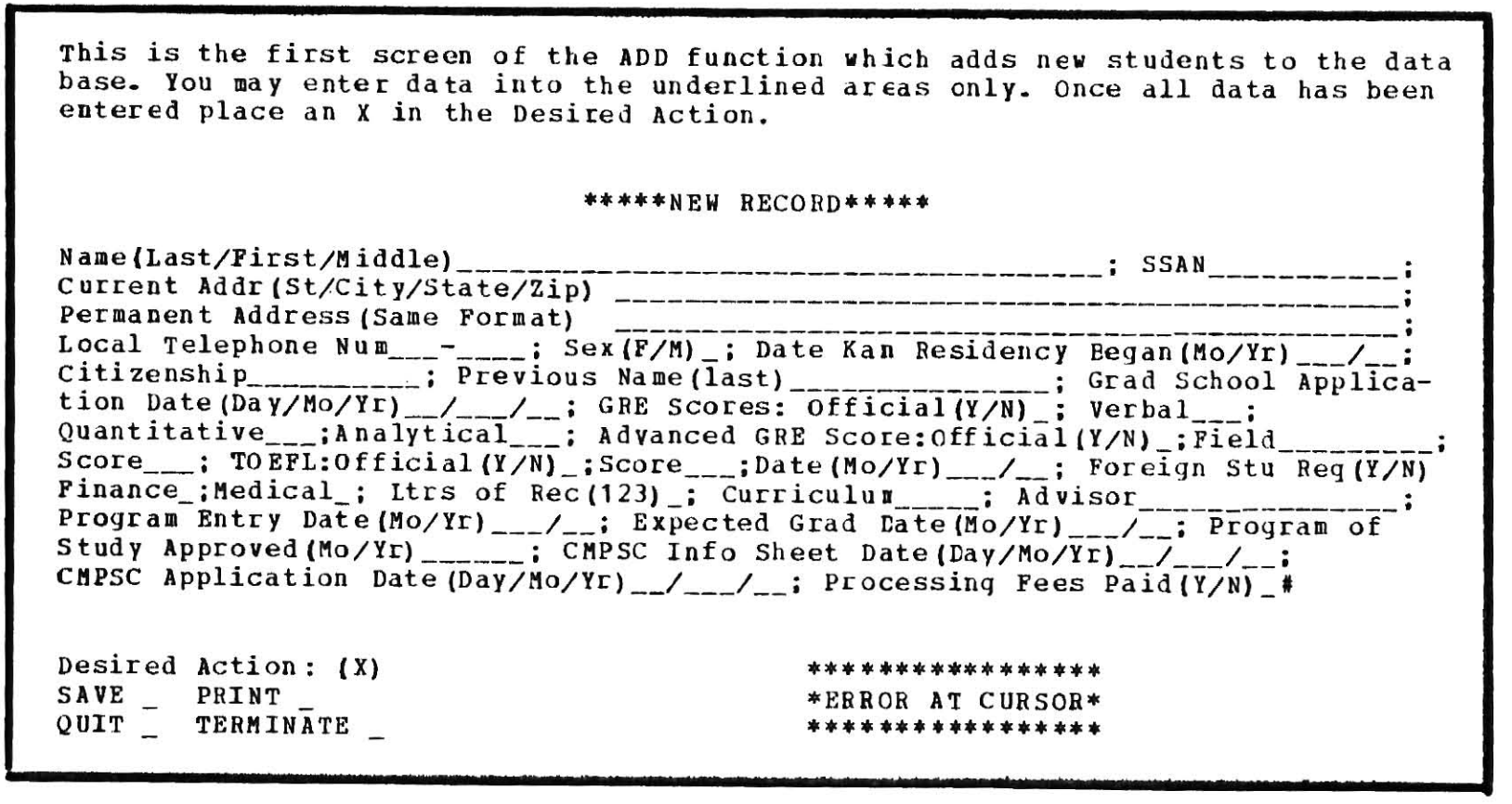


FIGURE 3.6: ADD FUNCTION

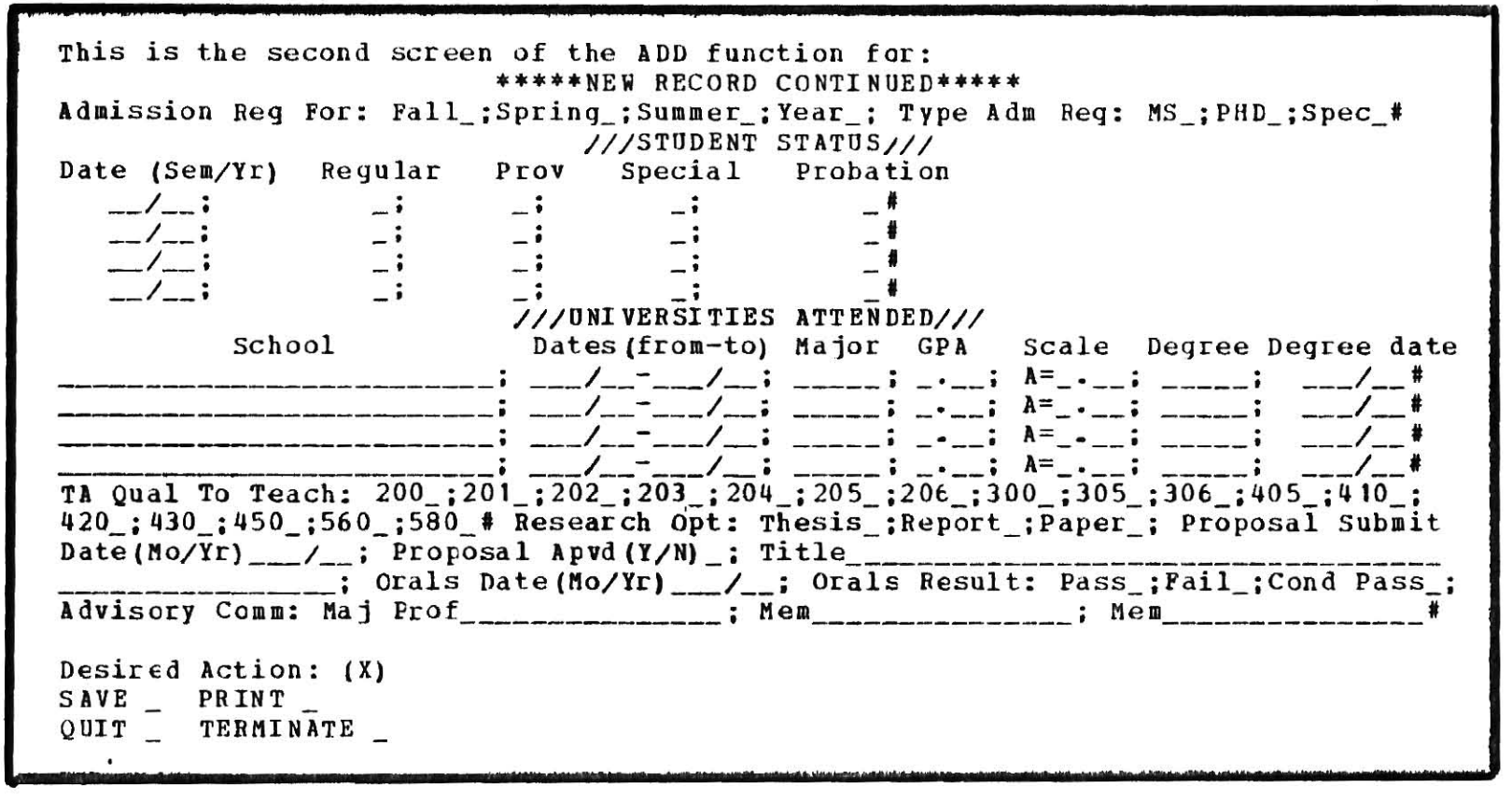


FIGURE 3.7: ADD FUNCTION

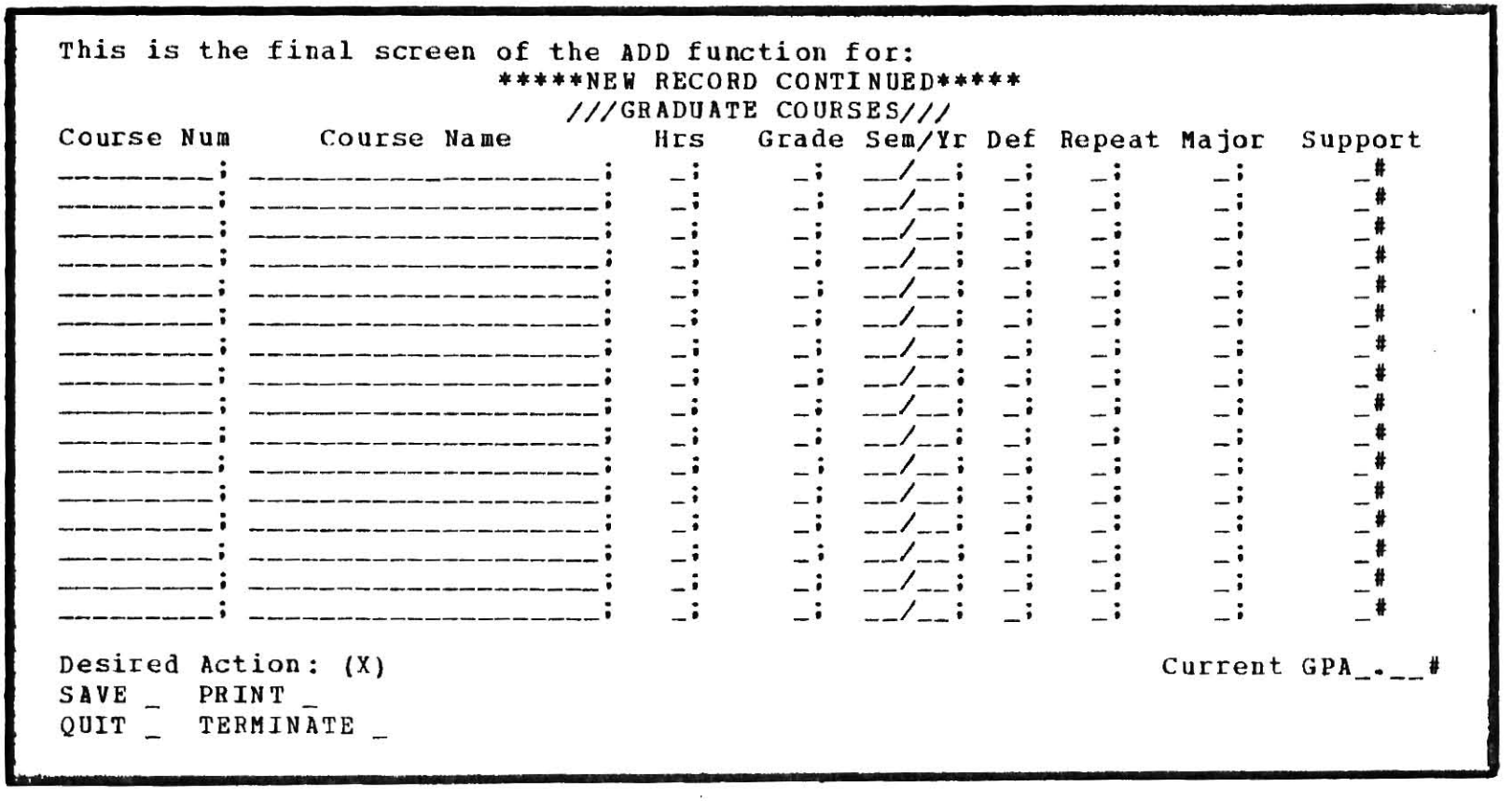


FIGURE 3.8: ADD FUNCTION

3.5 DATA BASE MAINTENANCE TRANSACTIONS

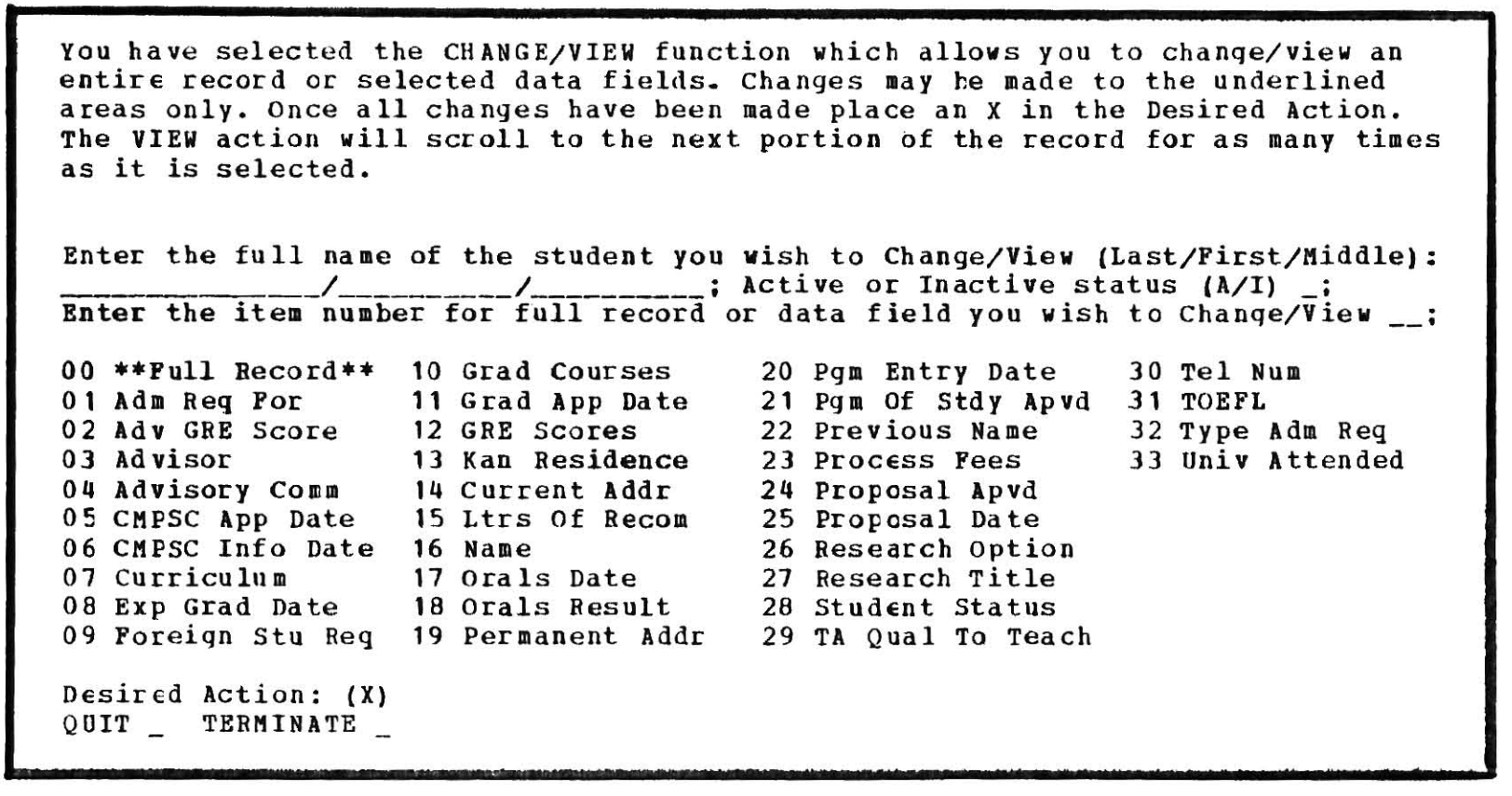
Having loaded the data base with student information, the maintenance of this data, to insure data integrity, now becomes the concern. For the purposes of this system maintenance is concerned with the changing of data items in a student record or the elimination of a record from one or both of the data bases. Data base maintenance is performed through the functions of "Change/View Student Information", "Purge A Student To Inactive Data Base", and "Delete A Student". Once the user selects any of these three functions, the first screen is displayed with a request for the student's full name. The student's name is the only key the user is expected to know to access all the information in the system. In the case of the change and delete functions, a student status (active or inactive) must also be entered to let the system know which data base to access.

The Change/View function should be selected whenever a student record, whether complete or not, has already been established through the Add function. In looking at this function two options are available to the user for retrieving information. In both cases all three screens of the student record may be accessed, the difference, however, being the position of the cursor at a specified data field. In the first option, entering the Full Record item number (i.e.00) in the space provided will display the first screen

of the student record with the cursor positioned at the first data item. (See figures 3.9-3.10) Changes to the data may then be made by moving the cursor to the desired data field making changes in the field's contents, and then selecting the SAVE action. If the changed data passes the edit it will become a permanent part of the data base. If it does not pass the edit an error message will be displayed and the recovery process described for the Add function should be followed. The SAVE action will cause the display of the second screen of the student record. If the user does not want to change information on any screen the VIEW action may be selected to bring up the next screen. The VIEW action will continue to scroll to the next screen of the record as many times as it is selected. This action was designed so that the read only user could view all information maintained on a student.

The second option for retrieving information is for the user to enter the item number of the specific field he desires to change. This will bring forth the screen on which the data item resides and position the cursor at that data item. Changes can then be made, and the SAVE action selected in order to make them a permanent part of the data base. This option is designed to save the user time in locating specific data items within the three screen record.

Should the student name not be found in the data base specified, an error message to that effect will be



```
You have selected the CHANGE/VIEW function which allows you to change/view an
entire record or selected data fields. Changes may be made to the underlined
areas only. Once all changes have been made place an X in the Desired Action.
The VIEW action will scroll to the next portion of the record for as many times
as it is selected.


Enter the full name of the student you wish to Change/View (Last/First/Middle):
_____________/_________/_________; Active or Inactive status (A/I) _;
Enter the item number for full record or data field you wish to Change/View __;

00 **Full Record**  10 Grad Courses     20 Pgm Entry Date    30 Tel Num
01 Adm Req For      11 Grad App Date    21 Pgm Of Stdy Apvd  31 TOEFL
02 Adv GRE Score    12 GRE Scores       22 Previous Name     32 Type Adm Req
03 Advisor          13 Kan Residence    23 Process Fees      33 Univ Attended
04 Advisory Comm    14 Current Addr     24 Proposal Apvd
05 CMPSC App Date   15 Ltrs Of Recom    25 Proposal Date
06 CMPSC Info Date  16 Name             26 Research Option
07 Curriculum       17 Orals Date       27 Research Title
08 Exp Grad Date    18 Orals Result     28 Student Status
09 Foreign Stu Req  19 Permanent Addr   29 TA Qual To Teach

Desired Action: (X)
QUIT _  TERMINATE _
```

FIGURE 3.9: CHANGE/VIEW FUNCTION

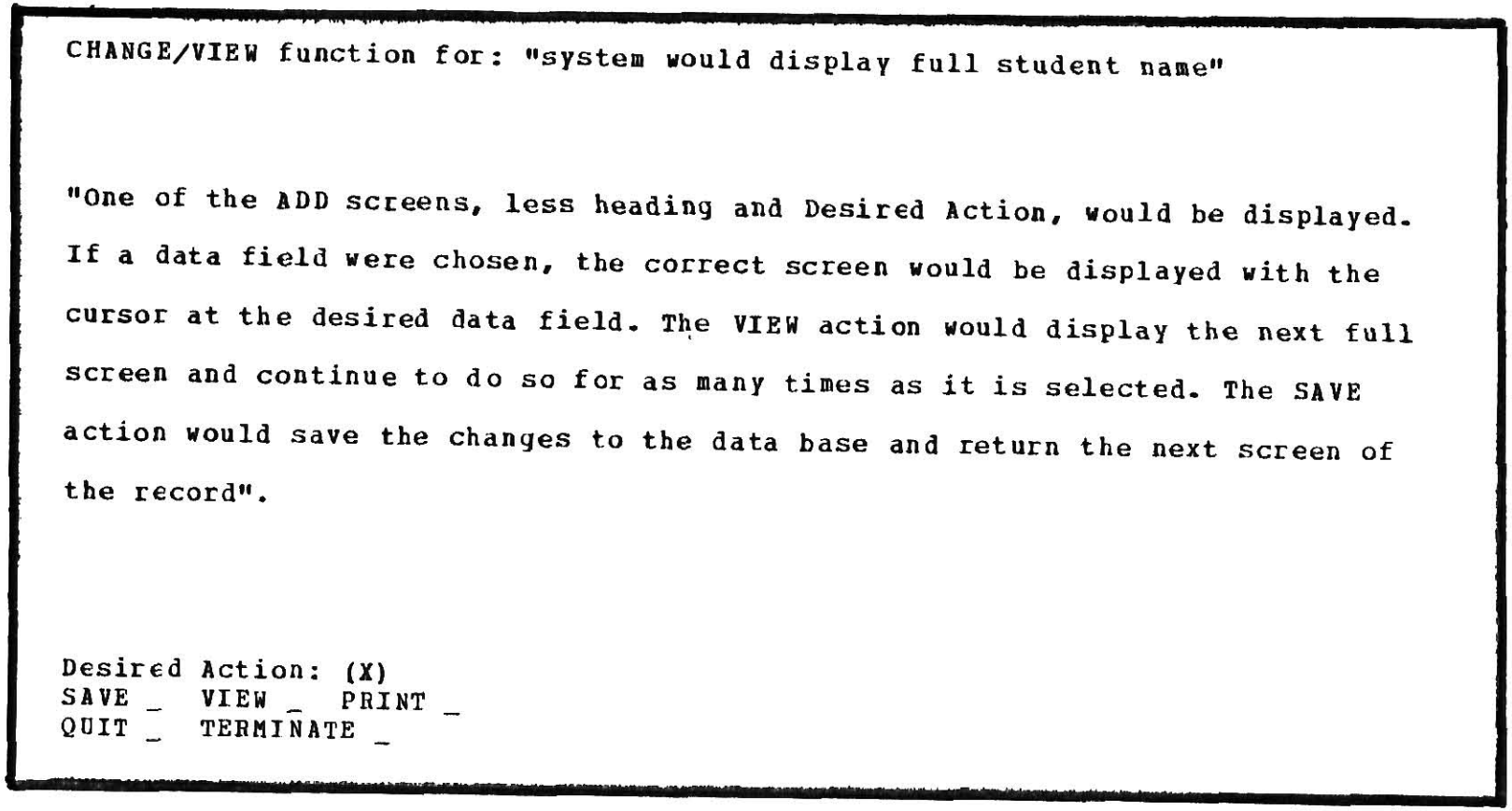


FIGURE 3.10: CHANGE/VIEW FUNCTION

displayed. To recover, both the spelling of the student's name and his status should be checked. To assist in this process, a listing may be obtained from the Output function which alphabetically lists students in the data base. If a read only user attempts to save data to the data base by selecting the Save action, an error message will be returned indicating "SAVE NOT ACCEPTED". To recover, an appropriate action should be selected. (ie Print, Quit, Terminate, View) Finally, the Change/View function facilitates printing of screens, terminating the function and exiting the system through the various DESIRED ACTIONS as described previously.

The second portion of the maintenance aspect for the system is the elimination of data. The functions supporting this aspect are "Purge A Student To Inactive Data Base" and "Delete A Student".(See figures 3.11-3.12) Both functions are similar in that one eliminates data from the active data base while the other eliminates it from the system. After selection of either function from the General Menu, the initial screen appears requesting the full name of the student. In the case of the delete function a student status must also be selected. The system responds by displaying the full student name and Social Security Number and asks if this is the student to be purged or deleted. The user must either enter yes or no (Y/N) before the process can be completed. The purge function moves the student information from the active data base to the inactive, while

the delete function eliminates the data from the desired data base. This two step process provides some extra assurance to the user that the appropriate student has been eliminated. If the name is not found in the data base, an error message is displayed with the same recovery procedures necessary as were discussed in the Change/View section. The only actions provided by the DESIRED ACTIONS portion of the screen, for either of these functions, are the QUIT and TERMINATE actions which were discussed previously.

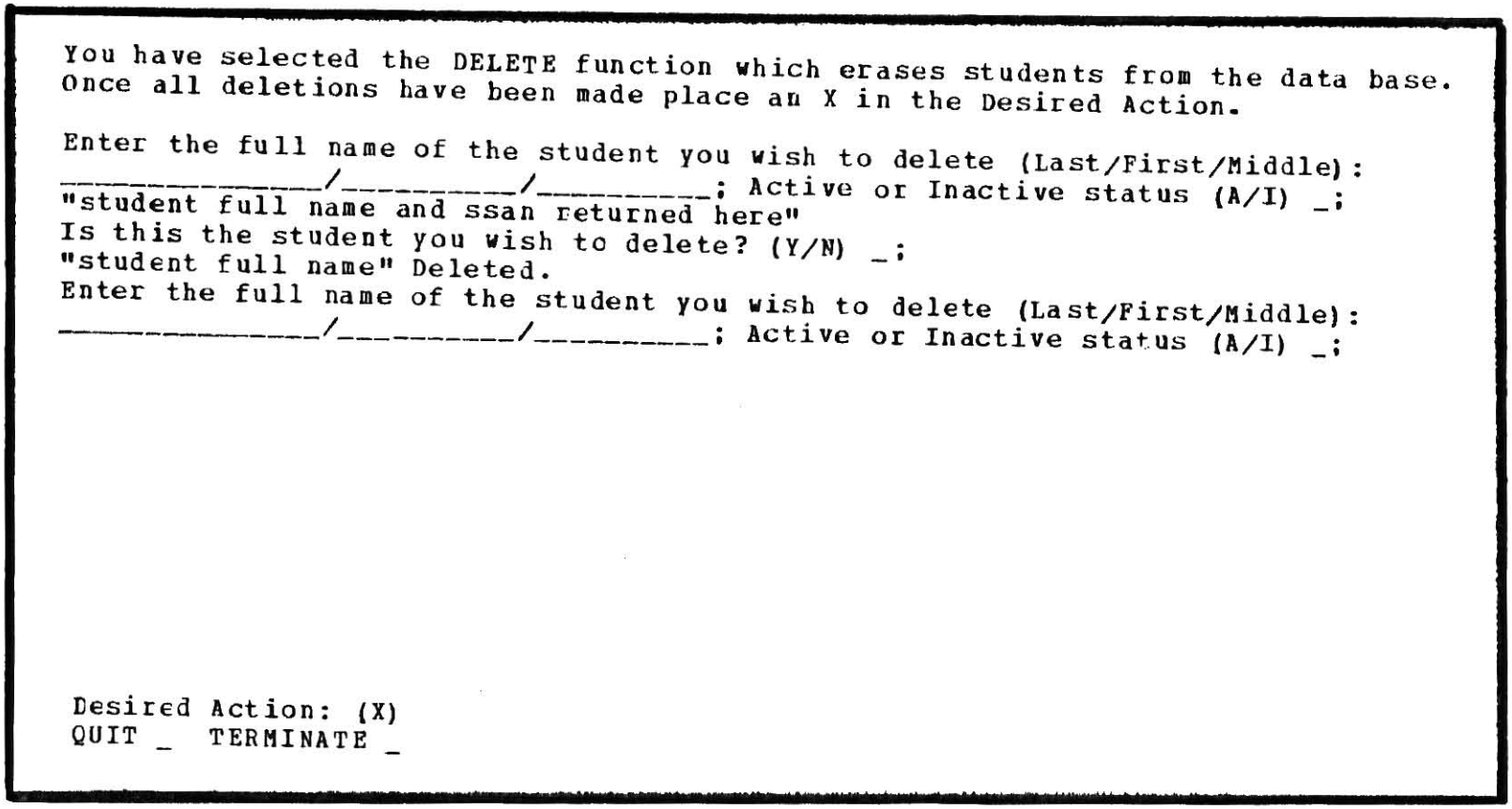


FIGURE 3.11: DELETE FUNCTION

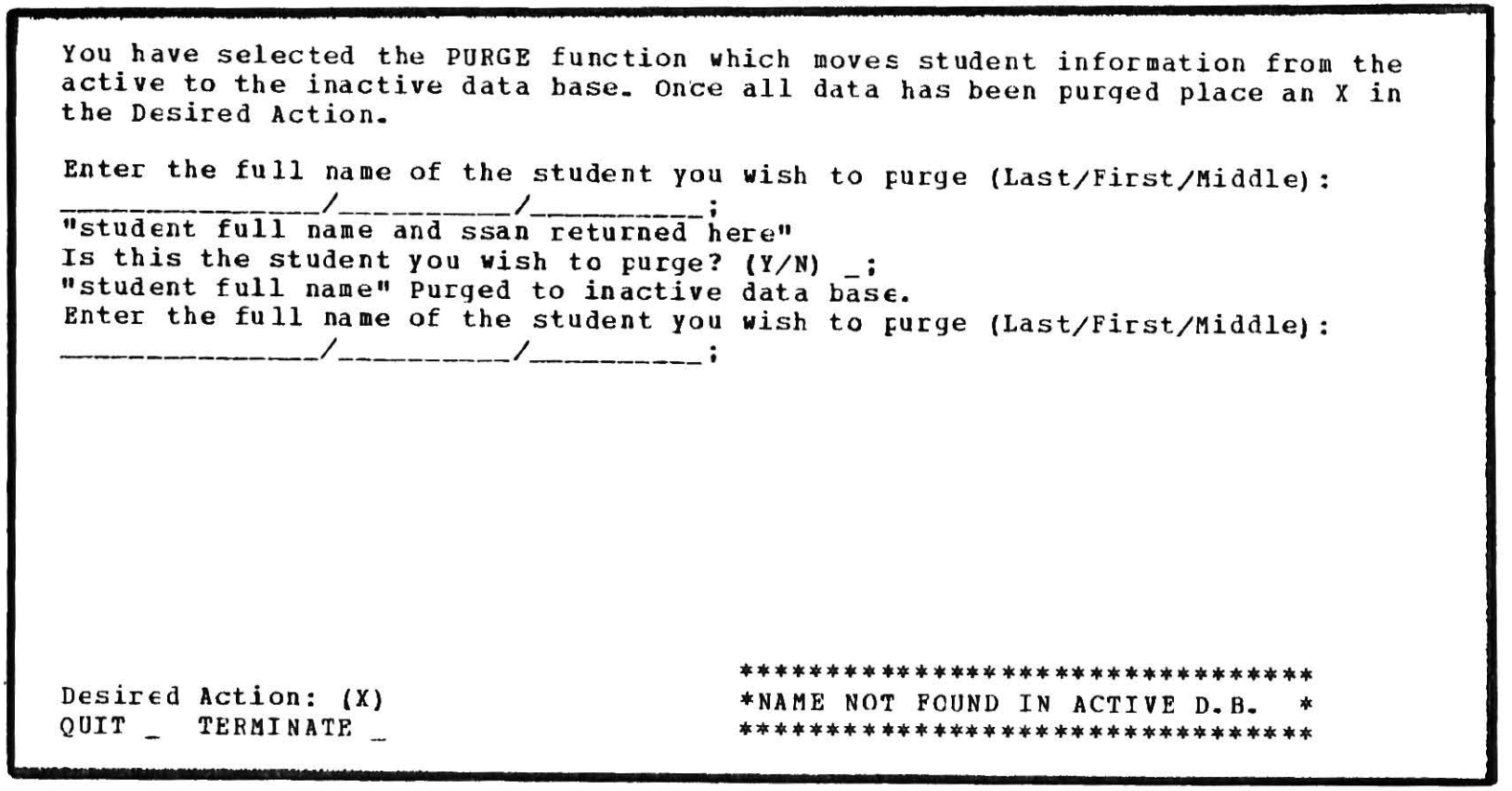


FIGURE 3.12: PURGE FUNCTION

3.6 DATA BASE UTILIZATION TRANSACTIONS

In the area of system utilization two types of users have been defined. The first type, the read-write users, are responsible for loading and maintaining the data as well as producing the various outputs required by the Computer Science Department. These users have full access to the system and are the key element in data base integrity and total system functioning. The second type of user is the read only user who may view student data and produce certain outputs but cannot, in any way, change the data. All the functions available to both types of users have been discussed in previous sections of this chapter. We now consider the final function, the "Outputs" function.

Once selected from the General Menu, the first screen of this function will be returned asking whether Department Letters or Student Information Reports are required. (See figure 3.13) Selecting either one of these will display a screen which provides a list of that type of output. (See figures 3.14-3.15) The specific output may then be selected and brought to the screen. In each instance a letter or report will be displayed with spaces provided for data entry. In the case of Department Letters, spaces will be provided for such items as name, address and date.

For Student Information Reports data are required to build QUEL queries for processing by the system output

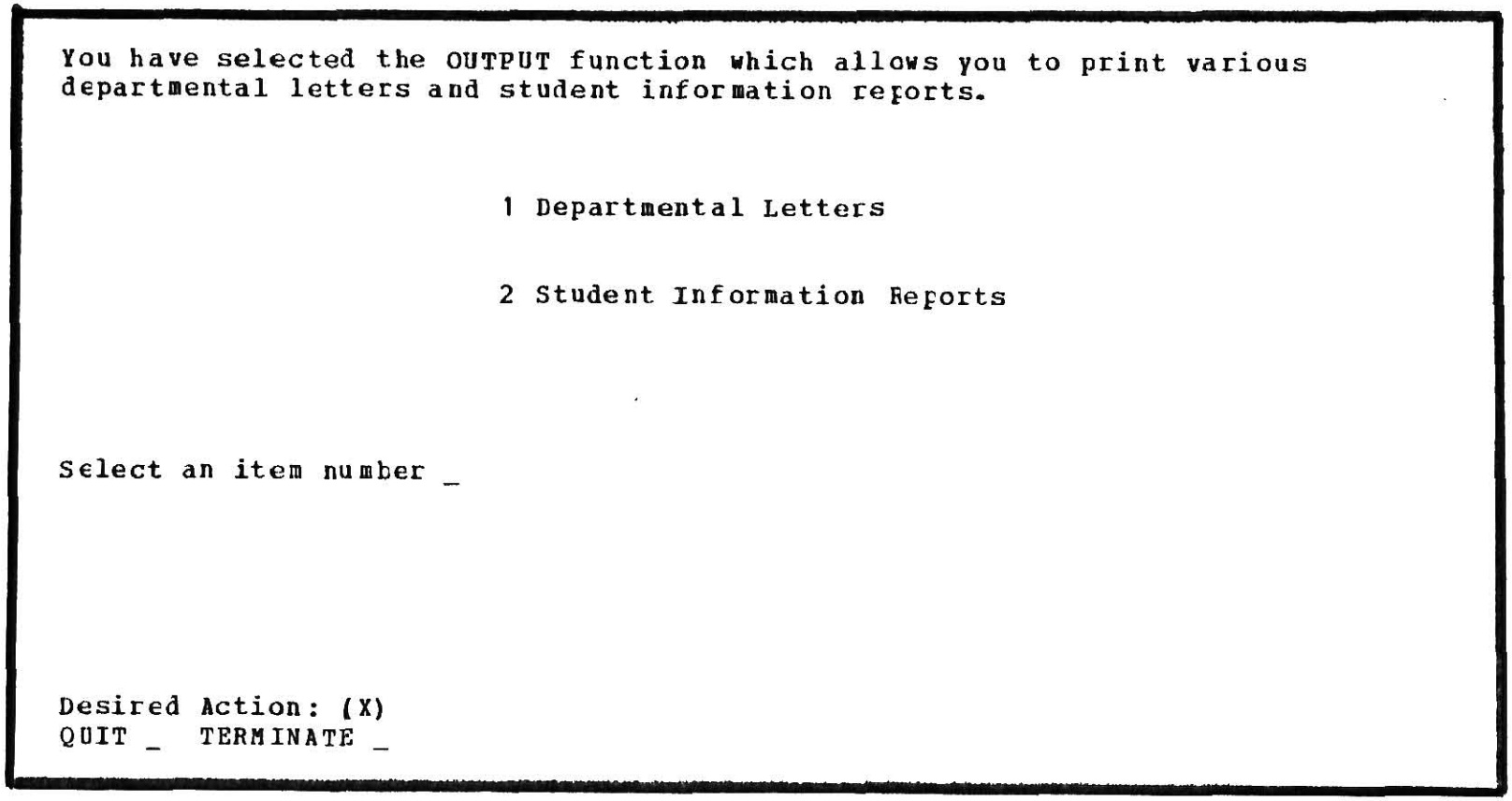


FIGURE 3.13: OUTPUT FUNCTION

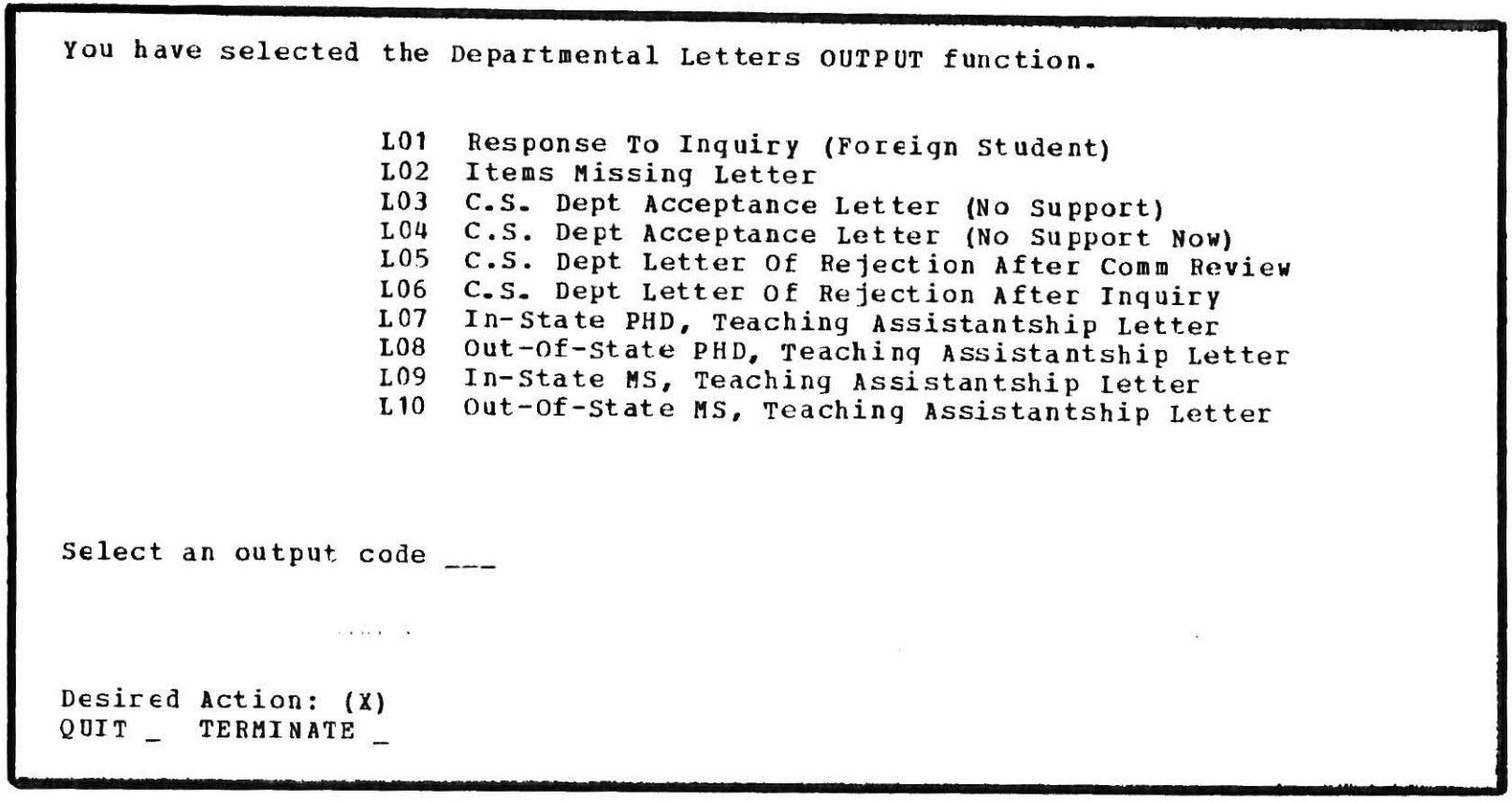


FIGURE 3.14: OUTPUT FUNCTION

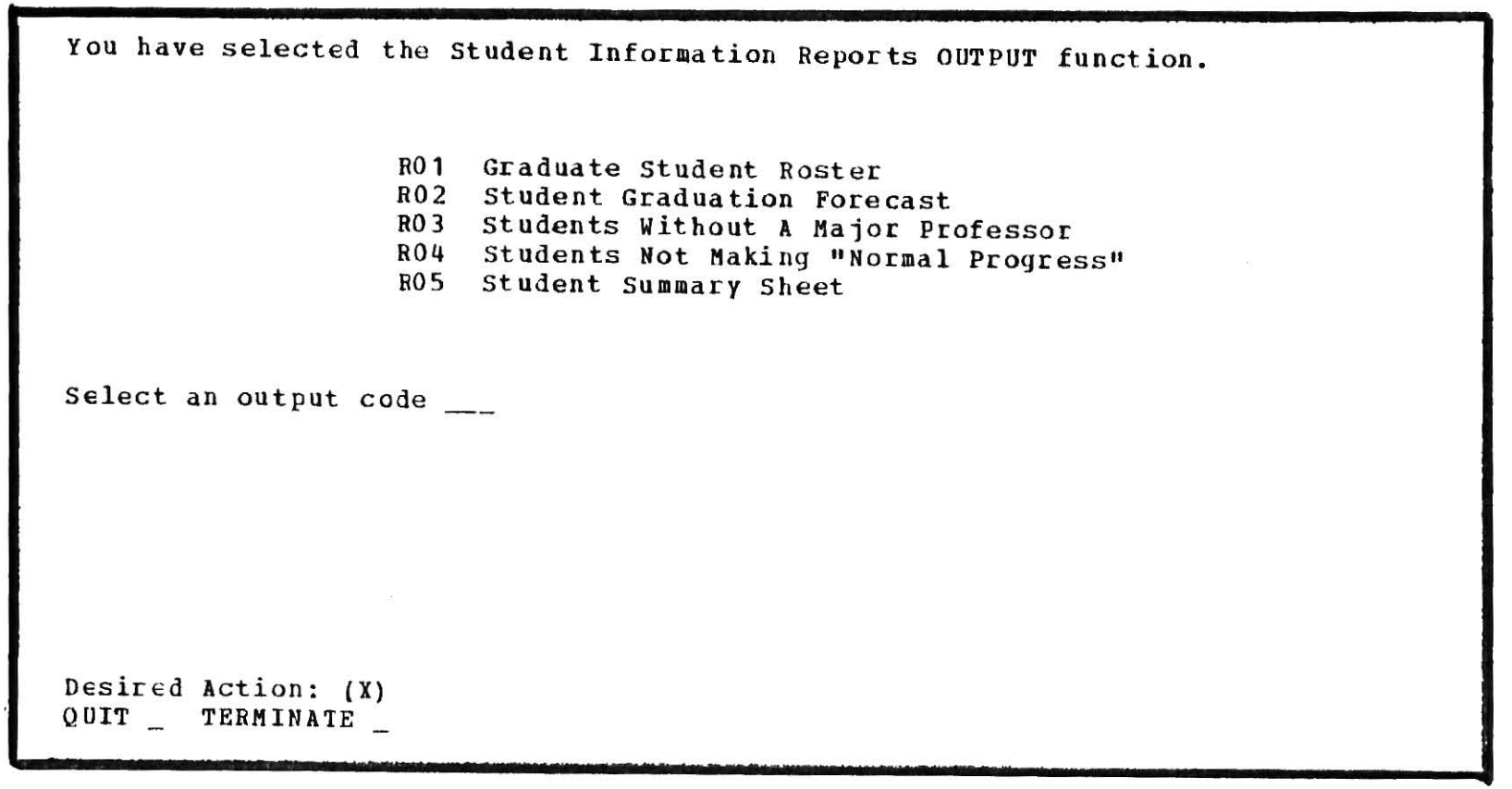


FIGURE 3.15: OUTPUT FUNCTION

processor. A Student Summary may be produced which is an outline of all data available on a given student. It is intended for faculty use in acedemic counseling. Once the specified output has been completely formatted on the screen, i.e., all needed data inserted, the PRINT action may be selected to produce a printed copy. In designing these screens, the Department Letters were designed for the read-write user; the read only user would almost always be concerned with the Student Information Reports. The same actions of PRINT, QUIT, and TERMINATE were provided as in other functions, however the SAVE action was not, as analysis indicated that it was not necessary to maintain letters/reports as a part of the data base. These are stored as UNIX files.

Chapter IV

IMPLEMENTATION

4.1 INTRODUCTION

Once the general design of the transaction processor had been completed, with screen formats and user functions developed, the environment in which the system was to execute was next analyzed. Specifically, implementation of the transaction processor using the INGRES data base management system running under the control of the Bell Laboratories' UNIX operating system was considered. The decision to use both these software packages was made by the department and was based on the fact that both UNIX and INGRES were already available on the department Perkin Elmer minicomputer. Actual implementation of the transaction processor utilizing a prototype approach was considered infeasible due to time and resource constraints. Additionally, research on the part of one of the Department Software Technicians, revealed that a major change release on the INGRES data base management system was due for user distribution in the January 1983 time frame. This meant that any implementation that was done prior to this date would more than likely have to be redone in terms of the new INGRES release. A logical analysis of the implementation,

however, was conducted to establish a framework upon which a future prototype system could expand. The capabilities and constraints of both the UNIX and INGRES software packages, their interaction and influence on system design, are outlined in the sections that follow.

4.2 USE OF UNIX

UNIX was developed at Bell Telephone Laboratories and has a simple command language that interfaces with a powerful, device independent file handling system. UNIX is used throughout the Bell System and in several hundred colleges, government agencies, and industrial installations, all of whom apply it to a wide range of applications. (15) Though UNIX posed no significant problems in the design of the transaction processor, several of its capabilities and limitations had to be taken into consideration. The first, and probably most important, is the file structure. The foundation of UNIX is a hierarchically structured file system, with multiple directories for different users and projects. There are no required access methods; the operating system handles the physical aspects of file management (device, track, sector, etc.) so that a file appears to a program as a sequence of bytes, upon which any desired structure may be imposed. Each user has a private directory which contains only the files that belong to him. These files are organized into a tree, with it possible for

any user to "walk" around this tree and find any file in the system, by starting at the root of the tree and walking along the proper set of branches. This freedom of access impacted system security by forcing a layered approach into the design, so as to prohibit unauthorized users from entering the data base and programs which access the data base. The only protection provided by UNIX is a minimal logon procedure (user ID and password) and a facility to protect files and directories with a read only control access. What this meant was that additional security for the GRASTURS system had to be provided in the application software and INGRES data base management system. Though not an overly complex problem, one that would not have had to be considered had a more sophisticated operating system been emplcyed.

Perhaps the greatest advantage UNIX provides is its established interface with the INGRES data base management system. INGRES was designed to run under UNIX and there are, therefore, several UNIX facilities which make data base creation, maintenance and use much easier. For example, UNIX allows for a single "INGRES Superuser" who is the only one that can authorize certain actions and procedures against a data base. This helps to insure data base integrity and adds to the security of the data base. Additionally, the ease with which the INGRES system may be entered is illustrated by the fcllowing UNIX command:

ingres "DATABASE NAME"

This command would be the only one required to be written into the data base interface program to allow the transaction processor access to the data base. Finally, through the use of the "sysmod" command, UNIX converts all relations in the data base to their best structure for use in INGRES. This command should be executed immediately after data base create in order to gain maximum access performance. Doing so will reduce the amount of time a user has to wait for the system to respond to his request for information.

One of the prime capabilities provided by UNIX which aided in the design of the system was the use of the text editor facility for creating files. As discussed in chapter two, a UNIX system file is built for each relation on a screen, facilitating the use of INGRES "Copy From" commands for adding data to the data base. This eliminates the need for using the more cumbersome "Append" command. The ease with which the text editor builds files is shown below:

ed -- ENTERS THE TEXT EDITOR

a -- APPENDS TEXT TO THE BUFFER

"DATA FROM THE SCREEN"

q -- QUIT THE TEXT EDITOR

Using these text editor files allows an entire relation to be entered into the data base at one time without the need for multiple INGRES data manipulation language commands. A facility that should prove extremely valuable once a prototype system is undertaken.

4.3 USE OF INGRES

As mentioned previously, INGRES is a relational model data base management system, designed to run under the UNIX operating system. The data manipulation language supported by the INGRES system is called QUEL (QUEry Language). It has several constraints which were considered but had no real impact on the design of the transaction processor. For example, a relation cannot have more than 49 domains and the tuple width cannot exceed 498 bytes. (9) Further, character strings can only be from 1 to 255 charactrers in length. (8) Though not impacting the design of the system, these constraints must be kept in mind should future implementation require an expansion of the data base.

As mentioned in the use of UNIX, the prime capability of INGRES, with regard to the transaction processor, is its ability to enter entire tuples into the data base and retrieve these same tuples without excessive data manipulation commands. This reduces the complexity of these commands and makes their inclusion in the data base interface program, where they are executed, a much simpler process. To assist in building the UNIX files INGRES allows delimiting characters to be specified for use with its "Copy" commands. This facility allows the formatted screen data to be moved into the UNIX files without losing its integrity with regard to domain sequence within the relation. Semicolons were used as delimiters between data

items which correspond to domains, and the "#" special character delimited the relations within the data base. These delimiters are removed when copying data into the data base.

An area that needs to be addressed during implementation, is how best to store the data base once it is created in memory so as to maximize efficiency and improve processing speed. INGRES can store a relation within a data base in three different internal structures. When a relation is first created it is created as a "Heap", as this is the most efficient structure to use to initially fill a relation using the "Copy" or "Append" commands. This storage structure, however, has two major disadvantages in that duplicate tuples within a relation are not removed and nothing is known about the location of the tuples. INGRES would have to read every tuple in the relation to find the one requested. It is, therefore, more desirable to restructure the data base into either the "Hash" or "Isam" structure once it has been created. Both structures eliminate duplicate tuples and speed access, as searches are conducted on key domains. The "Hash" structure is advantageous for locating tuples referenced in a qualification by an exact value. "Isam", on the other hand, is useful for both exact values and ranges of values. Since the "Isam" directory must be searched to locate tuples, it is never as efficient as "Hash". Also, as currently

designed, the data base should never have to provide a response to a query based on a range of values. This argues strongly in favor of the "Hash" storage structure as being the most appropriate for the system.

Perhaps the greatest disadvantage to using INGRES in the design of this system was the relative lack of experience on the part of the author and the department in general. Additionally, though the system documentation proved adequate, there was very little additional reference material to be found once something became unclear. This problem, it is hoped, will be remedied once the new INGRES change package is released to users, with accompanying documentation. Further, as INGRES is installed and becomes operational, the experience factor of depatment technicians will increase, facilitating greater use and future project implementation.

4.4 SYSTEM INTERACTION

The transaction processor's interaction with both the UNIX operating system and the INGRES data base management system takes place through a series of Pascal programs. Specifically, the data base interface program would contain Pascal code to execute the commands necessary to implement the UNIX text editor for building system files. This program would also contain the data manipulation language necessary for entering data into and retrieving data from the data

base. Finally, this Pascal program must contain the procedures necessary for logging users onto the INGRES system. This program is seen as being very complex with procedures and facilities needed for interaction with both UNIX and INGRES.

A second program of the transaction processing system which interacts directly with the UNIX system is the transaction processor. This program is responsible for displaying both users' menus and data on the terminal. This would require the program to use UNIX input/output facilities for data transfer and screen formatting. Further, as part of the system layered approach to security, a second UNIX users' file would be maintained by this program to insure that only authorized users entered the GRASTURS system.

The final program that interacts with UNIX is the screen formatter. This program maintains screen formats, as part of the system, on UNIX files. The procedures for accessing these formats from the files must be written as part of the Pascal code.

The problems of system interaction are not seen as being trivial. If any project is going to falter it is going to be where it interfaces and interacts with its surrounding environment or its subsystems of which it is made up. What complicates the task here is that we are dealing with three distinct entities. The UNIX operating system, the Pascal

coded transaction processing system and the INGRES data base management system. Insuring that all three work together to accomplish the assigned goal is seen as a prime task in implementing this system.

Chapter V

CONCLUSIONS

5.1 SUMMARY

This report has outlined the design of a transaction processing system to be implemented as a component of the Graduate Student Record System (GRASTURS). As an integral part of GRASTURS, the transaction processor aids in addressing the problem of automating the administrative tracking of Computer Science graduate candidates from initial information request to completion of all requirements for graduation. Specifically, the report has concerned itself with the design of a user friendly interface for the loading, maintenance and use of student data, required for this administrative tracking process. The system interface with the other two components of GRASTURS, the Graduate Student Record Data Base (active and inactive) and the Output Processor, has also been discussed at a functional level.

The design approach for the transaction processor was one of modularity, in that functions within the system were defined and addressed as separate entities. This allowed greater flexibility in terms of design and overall system development. The operating environment, within which the

system was designed to function, included the Computer Science Department Perkin Elmer minicomputer running the UNIX operating system and the INGRES data base management software packages. The methods employed throughout the design, involved a great deal of interaction with the system users, as well as the developers of the GRASTURS data base and output processor functions. Where possible, a stepwise methodology was taken in that the design of each function was accomplished in increments, from beginning to end, before considering the next function.

In looking at the general design of the system, the report addresses the security of the system; the user, data base, and output processor interfaces; and an editor function.

A layered approach was adopted for system security, with two layers of user validation and one of data base authorization decided upon. Two types of users were also identified. Those requiring full system access and those only requiring a read only capability. The key design issue, with regard to the user interface, was one of "friendliness". This was true throughout the development process and included such considerations as menus, aids, and prompts; error handling; screen layouts; system instructions; and screen display intensities. The data base interface centered around the ability of the system to access student data through the INGRES data base management

system. The use of the INGRES "Copy From" command, as opposed to the more cumbersome "Append" command, was felt to be a more expedient method for adding data to the data base, while the "Retrieve All" command was used for accessing all the information on a student from the data base. With regard to the output processor interface, the design approach was not to have the transaction processor do any output processing. Control is passed to the output processor for formatting, and/or printing of departmental letters, student information reports or screen displays. Once output processing is complete, control is returned to the transaction processor. The final function addressed as part of the general design was that of an editor function. A simple suitability editor was designed, with edit criteria established to aid in data base integrity. The editor further insures that departmental letter output meets certain minimally established criteria before printing.

The report goes on to explain in detail the various user initiated functions for loading, maintaining and utilizing the student data in the GRASTURS system. This is done through a series of figures representing actual screen formats for the Add, Change, Purge, Delete, and Output functions. A thorough explanation is then provided so the reader gains an appreciation for how student information is processed through the system using each of the functions and its associated screens.

Finally, the report addresses the capabilities and constraints of both the UNIX operating system and the INGRES data base management system, their interaction and their influence on system design.

As the design of the transaction processing system progressed, it became obvious that there were some areas that were only partially addressed, and some that were not addressed at all, with regard to system capabilities. This was due primarily to time and resource constraints. The thrust of the project was to thoroughly design and analyze as many of these areas as possible within these constraints. Areas that were not totally addressed were not ignored, however, but were considered as system enhancements and are presented as future work in the following section.

5.2 FUTURE WORK

The first, and probably the most important, area of consideration for system enhancement is the upgrading from a single user to a multiple user environment. The ability of multiple users to access student data from the data base and process that data concurrently, is a key requirement for the system. This analysis should include not only multiple read only users but multiple read-write users simultaneously accessing and modifying student data within the system. Though not addressed in the current system, this enhancement remained a consideration throughout the design, with nothing done that would prohibit its future implementation.

The next consideration for system enhancement lies in the area of security. Specifically, the requirements of the Privacy Act of 1974 need to be addressed. The system, in the broadest sense, does provide for the controlled access and safegarding of sensitive student information, however, further research and analysis needs to be done to insure that the system totally meets the requirements of the act. As the report emphasizes, only a minimum level of security has been designed into the current GRASTURS system. To further enhance system security, analysis should also be conducted on a transaction audit trail. This audit trail should record dates, times and changes to the data by the various users, as well as unauthorized attempts at accessing the data base. The data base administrator could then use these audit trails to assist in detecting whether system security had been breached.

An enhancement to the editor function should also be considered at a future date should the integrity level of the data base so dictate. Currently, the editor is only a simple suitability edit which checks the syntax of data entering the data base with no regard to data content. If data integrity decreases to unacceptable levels, this function should be upgraded to include a more stringent suitability edit, with analysis conducted into the possibility of adding a compatability edit.

Still another area to be looked into as part of future enhancements is moving student information from the inactive back to the active data base. An occasion to use this facility would occur whenever purged students (i.e., inactive status) decided to re-enter the program. Being able to move all inactive student information back to the active data base, utilizing a few INGRES commands, would eliminate the need for re-entering the three screens of data required to be maintained on each student. This facility could be implemented as part of the currrent Purge function, with the user provided the option of moving information from either an active to inactive status or inactive to active status.

Finally, once a prototype system has been implemented and the user has had an opportunity to work with the current screen formats, an analysis should be conducted to determine the adequacy and ease of use of the formats. One area in particular that should be looked at, is the quantity of data items displayed on each screen. The question of how much to put on a screen became very difficult to address without seeing the actual screen and being able to evaluate its use in processing data. With the underlying concern of the entire system being one of user friendliness the analysis of this enhancement should be conducted as soon after implementation as practicable.

BIBLIOGRAPHY

1. Friedman, Lawrence, The New Look of Data Entry, Infosystems, vol 24, no 12, pp 46-50, December 1977.

2. Friedman, Lawrence, Data Entry: Managing the Installation, Infosystems, vol 25, no 12, pp 50-52, December 1978.

3. Orr, William, Data Entry Supersystems, Infosystems, vol 23, no 11, pp 42-46, November 1976.

4. Osborn, R.A., Bain, W.P., Lloyd, T., and Perring, P.H., SPG A Programming System for Commercial Transaction Processing, Computer Journal, vol 18, no 4, pp 290-297, November 1975.

5. Lelan, Gerrard, A Distributed System for Real-Time Transaction Processing, Computer, vol 14, no 2, pp 43-48, February 1981.

6. Heffler, Michael, Description of a Menu Creation and Interpretation system, Software-Practice and Experience, vol 12, no 3, pp 269-281, March 1982.

7. Dhiraj, S.K., and Aleta, G.M., The Display Text Editor TED: A Case Study in the Design and Implementation of Display-Oriented Interactive Human Interfaces, IEEE Transactions on Communications, vol 30, no 1, pp 111-119, January 1982.

8. Epstein, R., A Tutorial on INGRES, pp 1-33, December 1977.

9. Epstein, R., Creating and Maintaining a Database Using INGRES, pp 1-25, December 1977.

10. Schofield, D., Hillman, A.L., Rodgers, J.L., MM/1, A Man-Machine Interface, Software-Practice and Experience, vol 10, no 9, pp 751-763, September 1980.

11. Aderet, Avshalom, Menu-Driven and Command-Driven Word Processor Interfaces: Evaluation of Ease of Learning, Office Automation Conference Digest, pp 875-884, 1982.

12. Ireland, Patrick, A Model For Transaction Processing Systems, Unpublished PHD Proposal.

13. Kernighan, Brian, UNIX For Beginners-Second Edition, pp 1-14, September 1978.

14. Kernighan, Brian, A Tutorial Introduction to the UNIX Text Editor, pp 1-10, September 1978.

15. Cherlin, Edward, The UNIX Operating System: Portability A Plus, Mini-Micro Systems, vol 14, no 4, pp 153-159, April 1981.

DESIGN OF A TRANSACTION PROCESSING SYSTEM FOR THE GRADUATE STUDENT RECORD DATA BASE

By

DAVID K. SCHOTTEL
B.A., PARK COLLEGE, 1974

AN ABSTRACT OF A MASTER'S REPORT

Submitted in Partial Fulfillment of the

Requirements for the Degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

-1982-

ABSTRACT

A user friendly transaction processing system is a system that is easy to learn, easy to remember and easy to use and is one in which the mode of user computer processing lies between batch processing and interactive processing. This report discusses the design and implementation of such a system as part of the Graduate Student Record System for the Computer Science Department at Kansas State University. Through a series of menus and formatted screens the transaction processing system interfaces with the user to add, change, delete and display graduate student information. This report outlines the analysis of these functions as well as the problems encountered during their design. The interaction of the INGRES managed, relational Graduate Student Record Data Base, and the transaction processor is also discussed. Implementation issues are outlined, centering around the interaction of the UNIX operating system, the facilities of the INGRES data base management system and the Pascal coded transaction processing system. The report concludes by summarizing the system design issues and uses this as a basis for specifying future system enhancements.