

Chapter 1

Introduction

This third documentation product is a preliminary set of product specifications that reflect the design of the software. Basically, the purpose of this phase is to develop a detailed design for our software solution, including a definition of the relationship (interfaces) among units and detailed procedural flow.

Since the all of the software requirements probably hadn't been covered during the previous documentation products, the design phase usually an iterative process, enveloping changes in the development specification and corresponding changes in the design. Such changes if they are not major, are healthy sign that the user and developer are at least communicating. Perhaps the best management tool that may be employing during this phase is the structured walk through, which seeks to make the design process even more visible and hopefully leads to a more understandable product.

Our approach stresses hierarchical decomposition interface definition and modularity.

Chapter 2

Related Documents

- TOTEM, Requirements Definition Document for ARU, Totem Software Development Company, 1995 : This document is used for 'Design Environment' section of this document.
- TOTEM, Requirements Definition Document (Revised) for ARU, Totem Software Development Company, 1995 : This document is used for 'Design Environment' section of this document.
- TOTEM, Preliminary User's Manual of ARU, Totem Software Development Company, 1995 : This document is used for ' Structure Chart ' section of this document.
- Date, C. J., An Introduction Database Systems Volume 1, Fifth Edition., Addison-Wesley Company, 1990 : This book is used for 'Data Element Dictionary' section and Entity - Relationship Section' section of this document.
- Programmer's Guide of Borland's Delphi, Borland Corporation, 1995 : This book is used for 'Pseudo Code' section of this document.
- User's Manual of Borland's Delphi, Borland Corporation, 1995 : This book is used for 'Pseudo Code' section of this document.
- Reference Guide of Borland's Delphi, Borland Corporation, 1995 : This book is used for 'Pseudo Code' section of this document.
- Programmer's Guide of Borland's Paradox, Borland Corporation, 1995 : This book is used for 'File Dictionary' and 'Module Dictionary' sections of this document.
- User's Manual of Borland's Paradox, Borland Corporation, 1995 : This book is used for 'File Dictionary' and 'Module Dictionary' sections of this document.
- Reference Guide of Borland's Paradox, Borland Corporation, 1995 : This book is used for 'File Dictionary' and 'Module Dictionary' sections of this document.
- Weems Chip, Program Designing, D. C: HEATH AND COMPANY, 1990 : This book is used for 'Structure Chart' section of this document.

Chapter 3

Overview

3.1 About Our Design

The Automated Registration Utility system provides the user an easy-to-learn and easy-to-use environment. It will be easy for the users to keep all information necessary for the registration and retrieve this information as well. As the product development team, we have worked hard to make this software design as suitable to the user needs as possible. We hope that this product will help students at furthest levels. Any decisions about the Automated Registration Utility, can be forwarded to Totem Software Development Company. These will be taken into consideration in next versions.

3.2 Why have we chosen Borland's Delphi ?

Delphi represents a brand new way to develop applications for Windows. It combines the speed and ease of use of a visual development environment with the power, flexibility, and reusability of a fully object-oriented language, the world's fastest compiler, and leading-edge database technology.

Delphi is a component-based application development environment supporting rapid development of highly efficient Microsoft Windows-based applications with a minimum of coding. Many of the traditional requirements of programming for Windows are handled for the programmer within the Delphi class library, shielding him/her from complicated, or merely repetitive programming tasks.

Delphi provides design tools such as application and form templates, so the programmer can quickly create and test his/her application prototype. Then, by using Delphi's rich component set and intuitive code generation, he/she can turn his/her prototypes into robust applications that fit business needs.

Delphi's database tools enable the programmer to develop powerful desktop database and client/ server applications and reports. "Live" data can be viewed at design time, so the programmer knows immediately whether his/her query results are what he/she wants.

These summarize the reasons which made us chose Borland's Delphi as our software development kit.

Chapter 4

Design Environment

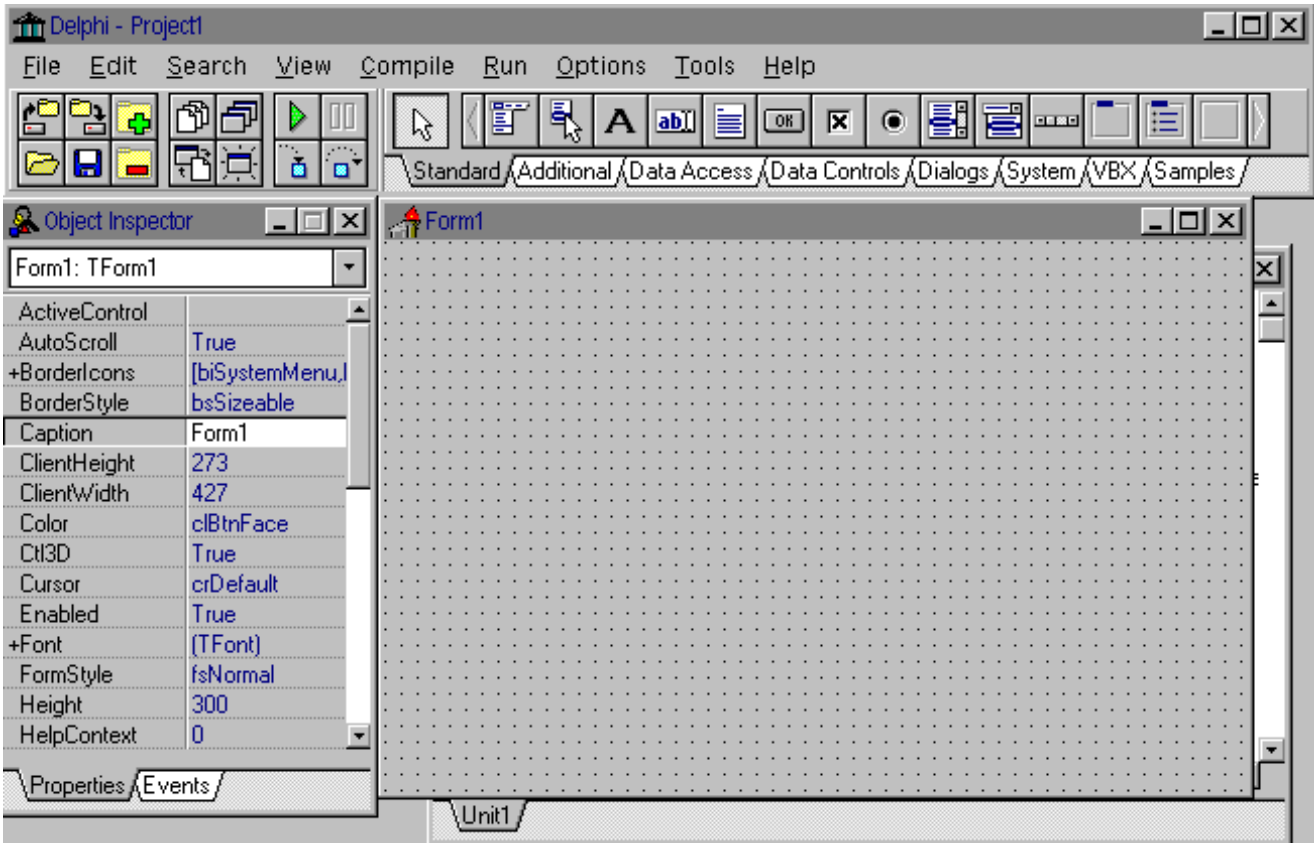
Automated Registration Utility is a fully database grounded program, so its database is created by the help of Paradox 5.1 for Windows. We have decided to use Paradox as our database management system because of two main reasons :

- We are already using a Borland product; Delphi. So, for the sake of compatibility and speed, we have used Paradox, as it is another Borland product.
- In the first steps of ARU design, we have selected Access to keep up with our database tables and queries. But, as soon as the real development begins, we notices that Borland's Delphi is not giving a direct support to a Microsoft product which is our Access. We may use the ODBC and such connections to achieve a database link. But it will be something much slower than a direct link support. As, of course, Delphi gives direct support to Paradox 4.x and Paradox 5.x versions, we have used Paradox 5.1 For Windows instead of Access.

Automated Registration Utility, is developed under a very new and strong combination of various systems. Basic information about these is given as follows :

4.1 Delphi

Issues concerning Delphi, are already explained in the Overview part. But what we want to show here about Delphi, is its IDE. It is a very powerful and complicated, integrated development environment that we can here give only a preview :



4.1.1 Forms

Forms are the focal point of nearly every application you develop in Delphi. You use the form like a canvas, placing and arranging components on it to design the parts of your user interface. Components are the building blocks of Delphi applications. They appear on the Component palette, displayed in the top right-hand part of the screen.

4.1.2 Component Palette

Components are the elements you use to build your Delphi applications. They include all the visible parts of an application, such as dialog boxes and buttons, as well as those that aren't visible while the application is running, such as system timers or Dynamic Data Exchange (DDE) servers.

4.1.3 Object Inspector

The Delphi Object Inspector enables you to easily customize the way a component appears and behaves in your application. The properties and events of the component that is selected in the form are displayed in the Object Inspector. You use the Properties page of the Object Inspector to customize components you've placed on a form (or the form itself), and the

Events page to generate and navigate among certain parts of program code, called event handlers. Event handlers are specialized procedures.

4.2 Paradox 5.1 For Windows

This is the latest version of the Borland's Paradox series and also the most sophisticated. We think that, it is better to give little information about this tool, because, it has various capabilities and none of them was essential for this project. In ARU development phase Paradox 5.1 is used only as an ordinary database management system. The tables are created by this tool and all the other manipulations are done by code under Delphi.

4.3 Windows 95

Windows 95 or version 4.0 is an advanced, 32-bit operating system that runs on 4MB machines and provides excellent response time to both 16 and 32-bit applications. It has the advantage of real multitasking which enables many programs to run at a time. This version has many features beyond the limit of version 3.1. In developing ARU, we have used Windows 95 Version 4.00.950 Release 3.

4.4 Computer System Used

In this project, we used a rather fast 486DX / 50 machine which has a hard disk of 420 MB and a quad speed CD-ROM drive. The amount of main memory, used in developing this project, is 8MB. With this configuration, we have encountered no serious problems in the design phase.

We decided to use this total system because of a basic reason. This total system consists of the latest versions of the development kits and a fast computer. As the product team, we thought that it would be easier and more reliable to develop a large project like Automated Registration Utility, in this system.

Chapter 5

Data Elements Dictionary

ARU uses a large database consisting of many tables and related queries. All these tables are created by using Borland's Paradox 5.1 For Windows which is a full database management system, so they support the relational database model at the furthest limits. The fields and information concerning them are given below table by table.

In the writing phase of this part we have used a special notation and title coordination. Let us give the meaning of the following words :

| | |
|-------------------|--------------------------------------|
| <i>Field Name</i> | : The name of the field in the table |
| <i>Type</i> | : The type of the field |
| <i>Length</i> | : The length of the field |
| <i>Comment</i> | : A brief comment about the field |

5.1 Database Tables

The tables of Automated Registration Utility are as follows :

5.1.1 Rules Table

Various rules concerning the registration process is kept in this table. This has two fields; an ID field and a string field containing the actual rule.

| Field Name | Type | Length | Comment |
|-------------|--------|--------|--|
| Rule ID | Short | | Unique ID number for a particular rule |
| Rule String | String | 100 | Rule information |

5.1.2 Departments Table

All of the departments of the university are kept here. This has two fields; an ID field and a string field containing the actual rule.

| Field Name | Type | Length | Comment |
|-----------------|--------|--------|--|
| Department | String | 4 | Unique ID string for a particular department |
| School | String | 35 | Name of the School that dept. belongs to |
| Department Name | String | 20 | Name of the department in full |

5.1.3 Advised Table

All of the advised courses advised by the advisor in a consultation are kept here. This has two fields; an course ID field and a student ID field.

| <u>Field Name</u> | <u>Type</u> | <u>Length</u> | <u>Comment</u> |
|-------------------|-------------|---------------|-------------------|
| Course ID | Short | | ID of the course |
| Student ID | Long | | ID of the student |

5.1.4 Courses Table

This table is the actual representation of the main course schedule of the semester in the database. Its records contains exactly the fields that are common to schedules.

| <u>Field Name</u> | <u>Type</u> | <u>Length</u> | <u>Comment</u> |
|-------------------|-------------|---------------|--|
| Course ID | Short | | Unique ID number for any particular course |
| Course Name | String | 40 | Name of the course in full |
| Instructor | String | 20 | Name of instructor |
| Credit | Short | | Credit value of the course |
| OpCode | Short | | Optic form code of the course |
| Department | String | 4 | Department code |
| Pre1 | Short | | First of the prerequisites in ID format |
| Pre2 | Short | | Second of the prerequisites |
| Pre3 | Short | | Third of the prerequisites |
| Section | Short | | Course section number |
| Days | String | 10 | Course days in M, T, W, TH, F format |
| Hours | String | 10 | Course hours |
| Rooms | String | 20 | Rooms in which that course is given |
| Quota | Short | | Maximum students to take that course |
| How Many In | Short | | Shows the number of stamps already taken for this course |
| Final | Date | | Date of the final exam |

5.1.5 Students Table

All the information about particular students are kept in student table. It contains not only student name, school ID numbers, class but also, semester number and department name.

| <u>Field Name</u> | <u>Type</u> | <u>Length</u> | <u>Comment</u> |
|-------------------|-------------|---------------|--|
| Student ID | Short | | Unique ID number for a particular student. |

| | | | |
|---------------|--------|----|---|
| Student Name | String | 20 | Name of the student |
| Class | Short | | Class of the student |
| Department | String | 4 | Department name of the student |
| Birth Place | String | 20 | Birth place of the student |
| Birth Date | Date | | Birth date of the student |
| Entrance Year | String | 11 | The year combination that shows the entrance of the student |
| Father Name | String | 10 | Father name of the student |
| PrLibrary | Logic | | Is there any problem with Library ? |
| PrInfirmary | Logic | | Is there any problem with Infirmary ? |
| PrMilitary | Logic | | Is there any problem with Military Office ? |
| Military | Logic | | Is military stamp taken ? |
| Infirmary | Logic | | Is infirmary stamp taken ? |
| Library | Logic | | Is library stamp taken ? |
| Course Stamp | Logic | | Had the student taken the course stamps ? |
| AllFinished | Logic | | Is the student already graduated ? |
| GPA | Number | | Shows the current GPA of the student |
| CC | Short | | Number of CC courses taken by student |
| HSS | Short | | Number of HSS courses taken by student |
| Finished | Logic | | Is registration finished ? |
| EdType | Char | | Shows the type of registration |
| Stay In Dorm | Logic | | Is student stays in dorm ? |
| VisaTaken | Logic | | Is Visa number taken ? |
| Visa | String | 16 | Student's Visa number |
| InsAttended | String | 20 | The last institute where the student graduated |
| Semester | Short | | Number of semester of the student |
| Tuition | Logic | | Is student paid the University Tuition ? |
| DormFee | Logic | | Is student paid the Dormitory Fee ? |
| RegFee | Logic | | Is student paid the Registration Fee ? |

5.1.6 Background Table

This table is more like a history of the university students. In this table, all the grades of the courses that were taken by each of the students is kept separately. Background table is the main source of information and is very important for solving the problem of prerequisites.

| Field Name | Type | Length | Comment |
|------------|--------|--------|---|
| Student ID | Short | | Unique ID number for a particular student |
| Course ID | Short | | Unique ID number for a particular course |
| Grade | String | 2 | The grade taken by the student in that course |
| Status 2 | String | 2 | This shows the course status, like NR for No Credit etc. |
| Semester | Short | | The semester in which the course had taken |
| Status | String | 2 | This also shows the course status, like NR for No Credit etc. |

5.1.7 Advisors Table

This table gives various information about the advisors of the university. Advisors table is mainly used to grant the users information request about the advisors at the main operations menu.

| <u>Field Name</u> | <u>Type</u> | <u>Length</u> | <u>Comment</u> |
|-------------------|-------------|---------------|---|
| Advisor ID | Short | | Unique ID number for a particular advisor |
| Advisor Name | String | 20 | Name of the advisor in full |
| Department | String | 4 | Department code |
| Semester | Short | | Semester number in which that advisor advises |

5.1.8 Calendar Table

This table implements the Academic Calendar of the university. Various dates are kept here and used in many places during the program progress.

| <u>Field Name</u> | <u>Type</u> | <u>Length</u> | <u>Comment</u> |
|-------------------|-------------|---------------|---|
| Event ID | Short | | Unique ID number for a particular event |
| Event | String | 20 | Name of the event in full |
| Event Date | Date | | Date of the event |

5.1.9 Passwords Table

This table keeps the passwords of the students. All passwords are saved encrypted and only decrypted when they are used. At no place, this decrypted version of any password is written. It is controlled and destroyed in code, only.

| <u>Field Name</u> | <u>Type</u> | <u>Length</u> | <u>Comment</u> |
|-------------------|-------------|---------------|---|
| Student ID | Short | | Unique ID number for a particular student |
| Password | String | 10 | Cryptic version of user password |

5.1.10 Last Table

This table lists the taken courses after the registration operation is finished. It is used by the registrar, just to grant or refuse these taken courses.

| <u>Field Name</u> | <u>Type</u> | <u>Length</u> | <u>Comment</u> |
|-------------------|-------------|---------------|----------------|
|-------------------|-------------|---------------|----------------|

| | | | |
|------------|-------|--|---|
| Student ID | Short | | Unique ID number for a particular student |
| Course ID | Short | | Unique ID number for a particular course |

5.1.11 Program Table

This table lists the courses offered only for this current semester. Also it keeps various information about the complementary nature of the courses.

| <u>Field Name</u> | <u>Type</u> | <u>Length</u> | <u>Comment</u> |
|-------------------|-------------|---------------|--|
| MainDept | String | 4 | Unique ID for a particular department |
| HSS | String | 3 | If 'yes' than this course is an HSS. |
| CC | String | 3 | If 'yes' than this course is an CC. |
| Course ID | Short | | Unique ID number for a particular course |

The information given in the upper part, is just presented for informative purposes. Any duplication or modification of the field structure, is strictly forbidden unless permission is taken from Totem Software Development Company.

5.2 Indices Created On Tables

In following lines, all the indices created on the tables of the ARU are explained in full detail. Explanation is given as a collection of tables whose headings are the same and consists of the following words. As the product development team, we decide to give the meaning of these heading words :

Index Name : That is the column of index names. In a relational database model every index should have a name. Notice that, this name is not the name of the DOS index file. The name of the file is given in another column.

Indexed Field : This column represents the name of the field or fields on which that particular index is created.

Index File : This is the column of the names of the actual DOS files which keeps the associated index. These files are normal binary files, created and used by the database management system.

Index Purpose : This column gives a little explanation about the purpose of the index, namely, why that particular index is created and used to accomplish what.

Following are the indices of the ARU database :

| Advisors Table | Index Name | Indexed Field | Index File | Index Purpose |
|----------------|--------------------|---------------|------------------------------|---|
| | Main Advisor Index | Advisor ID | ADVISORS.XG0 ADVISORS.YG0 | Reaching to any advisors in a request from any related table faster |
| | Adv Dept Index | Department | ADVISORS.XG1 ADVISORS.YG1 | Reaching to advisors of a particular department faster |

| Background Table | Index Name | Indexed Field | Index File | Index Purpose |
|------------------|----------------|---------------|--------------------------|---|
| | Back Stu Index | Student ID | BACKGR.XG0 BACKGR.YG0 | Reaching to any student-course records of a particular student faster |
| | Back Cou Index | Course ID | BACKGR.XG1 BACKGR.YG1 | Reaching to any student-course records of a particular course faster |

| Passwords Table | Index Name | Indexed Field | Index File | Index Purpose |
|-----------------|-----------------|---------------|------------------------------|--|
| | Main Pass Index | Student ID | PASSWORD.XG0 PASSWORD.YG0 | Reaching to any password record faster in a request from a related table |

| Rules Table | Index Name | Indexed Field | Index File | Index Purpose |
|-------------|-----------------|---------------|------------------------|--|
| | Main Rule Index | Rule ID | RULES.XG0 RULES.YG0 | Reaching to any rule record faster in a request from a related table |

| Calendar Table | Index Name | Indexed Field | Index File | Index Purpose |
|----------------|-----------------|---------------|------------------------------|---|
| | Main Cal Index | Event ID | CALENDAR.XG0 CALENDAR.YG0 | Reaching to any events faster in a request from a related table |
| | Cal Event Index | Event | CALENDAR.XG1 CALENDAR.YG1 | Reaching to any event information faster by the name of the event |
| | Cal Date Index | Event Date | CALENDAR.XG2 CALENDAR.YG2 | Reaching to any event of a particular date faster |

| Courses Table | Index Name | Indexed Field | Index File | Index Purpose |
|---------------|----------------|---------------|----------------------------|--|
| | Main Cou Index | Course ID | COURSES.XG0 COURSES.YG0 | Reaching to any courses faster in a request from a related table |
| | Cou Name Index | Course Name | COURSES.XG1 COURSES.YG1 | Reaching to any courses faster when a course name query is given |
| | Cou Inst Index | Instructor | COURSES.XG2 COURSES.YG2 | Reaching to any courses faster when a instructor name query is given |
| | Cou Dep Index | Department | COURSES.XG3 COURSES.YG3 | Reaching to any courses faster when given department code query |

| Students Table | Index Name | Indexed Field | Index File | Index Purpose |
|----------------|----------------|---------------|------------------------------|---|
| | Main Stu Index | Student ID | STUDENTS.XG0 STUDENTS.YG0 | Reaching to any student record faster in a request from a related table |
| | Stu Dep Index | Department | STUDENTS.XG1 STUDENTS.YG1 | Reaching to any students faster when given department code |

| | | | | |
|--|--|--|--|-------|
| | | | | query |
|--|--|--|--|-------|

| Instructors Table | Index Name | Indexed Field | Index File | Index Purpose |
|-------------------|-----------------------|---------------|------------------------|--|
| | Main Instructor Index | Name | INSTR.XG0 INSTR.YG0 | Reaching to any instructor record faster in a request from a related table |

| Departments Table | Index Name | Indexed Field | Index File | Index Purpose |
|-------------------|-----------------------|---------------|----------------------|--|
| | Main Department Index | Department | DEPT.XG0 DEPT.YG0 | Reaching to any department record faster in a request from a related table |

5.3 Estimation Of A Typical Database Size

A simple calculation on the tables gives the approximate disk space necessary to organize such a database. It goes as follows :

| | |
|--------------------------|---|
| Rules Table | $300 * 22 = 6600$ bytes for 300 rule records. |
| Courses Table | $500 * 128 = 64000$ bytes for 500 course records. |
| Students Table | $20000 * 35 = 700000$ bytes for 20000 students. |
| Background Table | $1000000 * 10 = 10000000$ bytes for 1000000 background records. |
| Advisors Table | $50 * 28 = 1400$ bytes for 50 advisor records |
| Calendar Table | $300 * 30 = 9000$ bytes for 300 event records |
| Passwords Table | $20000 * 12 = 240000$ bytes for 20000 passwords |
| Departments Table | $100 * 24 = 2400$ bytes for 100 departments |
| Last Table | $20000 * 4 * 5 = 400000$ bytes for 20000 students |
| Program Table | $200 * 12 = 2400$ bytes for 200 course offerings |
| Advised Table | $20000 * 4 * 5 = 400000$ bytes for 20000 students |

Total : $11825800 + \text{Additional Buffer (approx. 3 MB)} = 15 \text{ MB}$

So, for a full database of the explained kind at least 15 MB of hard disk space is necessary.

Chapter 6

File Dictionary

Automated Registration Utility, is a fully database oriented tool, so it is normal for it to use many database files. In addition to these database files, the ARU system uses a backup file to take the backup of total database. This backup file is nothing more than a collection of all database files mashed with ZIP data compression. In the following tables, the functionality of every table is explained one by one.

In the writing phase of this part, we have used a special title coordination. Let us give the meaning of the following words appearing in headings :

Name : The name of the file used in the ARU system
Functionality : The function of the file in the system
Comment : A brief explanation about the file

Now the files used in the system, will be explained according to these headings :

| | |
|----------------------|---|
| Name | ADVISORS.DB |
| Access Type | Indexed on Advisor ID and Department Fields |
| Functionality | Keeping various information about the advisors of the university. |
| Comment | This table is implemented for informative purposes, as well as practical purposes. Main necessity for Advisors Table is the process of listing advisors in the case of an information request |

| | |
|----------------------|---|
| Name | COURSES.DB |
| Access Type | Indexed on Course ID, Course Name, Instructor and Department |
| Functionality | Keeping various information about the courses given by the university. |
| Comment | This table is used in many places, for example, in giving course information, in advising new courses, etc. Practical purposes are more important in the implementation of Courses Table than informative purposes. |

| | |
|----------------------|--|
| Name | BACKGR.DB |
| Access Type | Indexed on Student ID and Course ID fields. |
| Functionality | Keeping the background of the whole university. |
| Comment | As this table contains all the grades taken in the history of university, it is mainly used in determining overloads, prerequisite satisfaction and so on. |

| | |
|----------------------|---|
| Name | STUDENTS.DB |
| Access Type | Indexed on Student ID and Department |
| Functionality | Keeping the information about the students of the university |
| Comment | This is one of the general purpose tables of the database. With the help of this table, information about students are used in many places like optic form development. |

| | |
|----------------------|--|
| Name | CALENDAR.DB |
| Access Type | Indexed on all of the fields |
| Functionality | Implementing the Academic Calendar of the university |
| Comment | This table is nothing more than a calendar keeping the important dates for the university. |

| | |
|----------------------|---|
| Name | PASSWORD.DB |
| Access Type | Indexed on Password ID field |
| Functionality | Keeping the system entrance passwords of the students |
| Comment | The passwords in this table are kept encrypted, so any unauthorized access to these are strictly forbidden. |

| | |
|----------------------|--|
| Name | RULES.DB |
| Access Type | Indexed on Rule ID field |
| Functionality | Keeping string versions of the various university rules. |
| Comment | In case of any limited operation, this table is used to explain why this university event is limited to the student using the system and not to some others. |

| | |
|----------------------|---|
| Name | ADVISED.DB |
| Access Type | Sequential |
| Functionality | Keeping information about the advised courses in a particular registration. |
| Comment | Temporary advised course information is readily obtained from here. |

| | |
|----------------------|---|
| Name | DEPT.DB |
| Access Type | Indexed on Department field |
| Functionality | Keeping full names of the various university departments. |
| Comment | Department full name is readily obtained from here. |

| | |
|----------------------|--|
| Name | LAST.DB |
| Access Type | Sequential |
| Functionality | Keeping the last courses taken after the process has finished. |
| Comment | This table is mainly bookkeeping for the registrar. |

| | |
|----------------------|--|
| Name | PROGRAM.DB |
| Access Type | Sequential |
| Functionality | Course offerings for the current semester are kept here. |
| Comment | This is the main course schedule. |

| | |
|----------------------|---|
| Name | ADVISORS.XG0 |
| Access Type | Sequential |
| Functionality | Index File on Advisors Table created by Paradox DBMS. |
| Comment | This index file is the first part of the twin index files created on one field which is Advisor ID. |

| | |
|----------------------|--|
| Name | ADVISORS.YG0 |
| Access Type | Sequential |
| Functionality | Index File on Advisors Table created by Paradox DBMS. |
| Comment | This index file is the second part of the twin index files created on one field which is Advisor ID. |

| | |
|----------------------|---|
| Name | ADVISORS.XG1 |
| Access Type | Sequential |
| Functionality | Index File on Advisors Table created by Paradox DBMS. |
| Comment | This index file is the first part of the twin index files created on one field which is Department. |

| | |
|----------------------|--|
| Name | ADVISORS.YG1 |
| Access Type | Sequential |
| Functionality | Index File on Advisors Table created by Paradox DBMS. |
| Comment | This index file is the second part of the twin index files created on one field which is Department. |

| | |
|----------------------|---|
| Name | BACKGR.XG0 |
| Access Type | Sequential |
| Functionality | Index File on Background Table created by Paradox DBMS. |
| Comment | This index file is the first part of the twin index files created on one field which is Student ID. |

| | |
|----------------------|--|
| Name | BACKGR.YG0 |
| Access Type | Sequential |
| Functionality | Index File on Background Table created by Paradox DBMS. |
| Comment | This index file is the second part of the twin index files created on one field which is Student ID. |

| | |
|----------------------|--|
| Name | BACKGR.XG1 |
| Access Type | Sequential |
| Functionality | Index File on Background Table created by Paradox DBMS. |
| Comment | This index file is the first part of the twin index files created on one field which is Course ID. |

| | |
|----------------------|---|
| Name | BACKGR.YG1 |
| Access Type | Sequential |
| Functionality | Index File on Background Table created by Paradox DBMS. |
| Comment | This index file is the second part of the twin index files created on one field which is Course ID. |

| | |
|----------------------|---|
| Name | CALENDAR.XG0 |
| Access Type | Sequential |
| Functionality | Index File on Calendar Table created by Paradox DBMS. |

| | |
|----------------|---|
| Comment | This index file is the first part of the twin index files created on one field which is Event ID. |
|----------------|---|

| | |
|----------------------|--|
| Name | CALENDAR.YG0 |
| Access Type | Sequential |
| Functionality | Index File on Calendar Table created by Paradox DBMS. |
| Comment | This index file is the second part of the twin index files created on one field which is Event ID. |

| | |
|----------------------|--|
| Name | CALENDAR.XG1 |
| Access Type | Sequential |
| Functionality | Index File on Calendar Table created by Paradox DBMS. |
| Comment | This index file is the first part of the twin index files created on one field which is Event. |

| | |
|----------------------|---|
| Name | CALENDAR.YG1 |
| Access Type | Sequential |
| Functionality | Index File on Calendar Table created by Paradox DBMS. |
| Comment | This index file is the second part of the twin index files created on one field which is Event. |

| | |
|----------------------|---|
| Name | CALENDAR.XG2 |
| Access Type | Sequential |
| Functionality | Index File on Calendar Table created by Paradox DBMS. |
| Comment | This index file is the first part of the twin index files created on one field which is Event Date. |

| | |
|----------------------|--|
| Name | CALENDAR.YG2 |
| Access Type | Sequential |
| Functionality | Index File on Calendar Table created by Paradox DBMS. |
| Comment | This index file is the second part of the twin index files created on one field which is Event Date. |

| | |
|--------------------|-------------|
| Name | COURSES.XG0 |
| Access Type | Sequential |

| | |
|------------------------------|--|
| Functionality Comment | Index File on Courses Table created by Paradox DBMS. This index file is the first part of the twin index files created on one field which is Course ID. |
|------------------------------|--|

| | |
|------------------------------|---|
| Name | COURSES.YG0 |
| Access Type | Sequential |
| Functionality Comment | Index File on Courses Table created by Paradox DBMS. This index file is the second part of the twin index files created on one field which is Course ID. |

| | |
|------------------------------|--|
| Name | COURSES.XG1 |
| Access Type | Sequential |
| Functionality Comment | Index File on Courses Table created by Paradox DBMS. This index file is the first part of the twin index files created on one field which is Course Name. |

| | |
|------------------------------|---|
| Name | COURSES.YG1 |
| Access Type | Sequential |
| Functionality Comment | Index File on Courses Table created by Paradox DBMS. This index file is the second part of the twin index files created on one field which is Course Name. |

| | |
|------------------------------|---|
| Name | COURSES.XG2 |
| Access Type | Sequential |
| Functionality Comment | Index File on Courses Table created by Paradox DBMS. This index file is the first part of the twin index files created on one field which is Instructor. |

| | |
|------------------------------|--|
| Name | COURSES.YG2 |
| Access Type | Sequential |
| Functionality Comment | Index File on Courses Table created by Paradox DBMS. This index file is the second part of the twin index files created on one field which is Instructor. |

| | |
|--------------------|-------------|
| Name | COURSES.XG3 |
| Access Type | Sequential |

| | |
|------------------------------|---|
| Functionality Comment | Index File on Courses Table created by Paradox DBMS. This index file is the first part of the twin index files created on one field which is Department. |
|------------------------------|---|

| | |
|------------------------------|--|
| Name | COURSES.YG3 |
| Access Type | Sequential |
| Functionality Comment | Index File on Courses Table created by Paradox DBMS. This index file is the second part of the twin index files created on one field which is Department. |

| | |
|------------------------------|--|
| Name | PASSWORD.XG0 |
| Access Type | Sequential |
| Functionality Comment | Index File on Passwords Table created by Paradox DBMS. This index file is the first part of the twin index files created on one field which is Password ID. |

| | |
|------------------------------|---|
| Name | PASSWORD.YG0 |
| Access Type | Sequential |
| Functionality Comment | Index File on Passwords Table created by Paradox DBMS. This index file is the second part of the twin index files created on one field which is Password ID. |

| | |
|------------------------------|--|
| Name | RULES.XG0 |
| Access Type | Sequential |
| Functionality Comment | Index File on Rules Table created by Paradox DBMS. This index file is the first part of the twin index files created on one field which is Rule ID. |

| | |
|------------------------------|---|
| Name | RULES.YG0 |
| Access Type | Sequential |
| Functionality Comment | Index File on Rules Table created by Paradox DBMS. This index file is the second part of the twin index files created on one field which is Rule ID. |

| | |
|-------------|--------------|
| Name | STUDENTS.XG0 |
|-------------|--------------|

| | |
|----------------------|---|
| Access Type | Sequential |
| Functionality | Index File on Students Table created by Paradox DBMS. |
| Comment | This index file is the first part of the twin index files created on one field which is Student ID. |

| | |
|----------------------|--|
| Name | STUDENTS.YG0 |
| Access Type | Sequential |
| Functionality | Index File on Students Table created by Paradox DBMS. |
| Comment | This index file is the second part of the twin index files created on one field which is Student ID. |

| | |
|----------------------|---|
| Name | STUDENTS.XG1 |
| Access Type | Sequential |
| Functionality | Index File on Students Table created by Paradox DBMS. |
| Comment | This index file is the first part of the twin index files created on one field which is Department. |

| | |
|----------------------|--|
| Name | STUDENTS.YG1 |
| Access Type | Sequential |
| Functionality | Index File on Students Table created by Paradox DBMS. |
| Comment | This index file is the second part of the twin index files created on one field which is Department. |

| | |
|----------------------|---|
| Name | DEPT.XG0 |
| Access Type | Sequential |
| Functionality | Index File on Departments Table created by Paradox DBMS. |
| Comment | This index file is the first part of the twin index files created on one field which is Department. |

| | |
|----------------------|--|
| Name | DEPT.YG0 |
| Access Type | Sequential |
| Functionality | Index File on Departments Table created by Paradox DBMS. |
| Comment | This index file is the second part of the twin index files created on one field which is Department. |

| | |
|----------------------|--|
| Name | ARUBACK.ZIP |
| Access Type | By decompressing it with PKUNZIP.EXE and then restoring to database. |
| Functionality | Keeping a backup of all of the database files. |
| Comment | This file is created every evening when closing the program and compared to the normal database files at the moment of the first running of the program. If any corruption is found, the latest backup is restored and so, information is kept safe. |

These definitions concerning Automated Registration Utility database files, construct the file dictionary of our software product. As you may notice, the files which are explained above are all database management system related files. ARU does not use any other file access except these and this concludes that ARU obeys the relational database model. Also, this proves the reliability of the data used in this program.

Chapter 7

Module Dictionary

In this part, as the ARU development team, we give all necessary information about the modules used in the software system. The dictionary is written in an accepted standard and in alphabetical order. This standard is explained below as follows :

| | |
|---------------------|--|
| <i>Name</i> | : Name of the module |
| <i>Parameters</i> | : Parameter list of the module and their functions |
| <i>Function</i> | : The function of the module in the system |
| <i>Sub-function</i> | : Names of the modules called by that module |
| <i>Callers</i> | : Names of the modules which call that module |

Now, let us give the module dictionary in full detail :

| | |
|--------------|---|
| Name | CopyrightScreen.Main |
| Parameters | None |
| Function | Shows the copyright agreement screen |
| Sub-function | CopyrightScreen.Show, CopyrightScreen.Hide, SecurityScreen.Show |
| Callers | Main Body |

| | |
|------------|--|
| Name | CoursesScreen.Main |
| Parameters | None |
| Function | Give information about the courses offered by the university |

| | |
|--------------|----------------------------------|
| Sub-function | DetailedCourseScreen.Show |
| Callers | MainOperationsScreen.ListCourses |

| | |
|--------------|--------------------------------|
| Name | LogoScreen.Create |
| Parameters | None |
| Function | Show the Totem ARU logo screen |
| Sub-function | LogoScreen.ShowModal |
| Callers | Main Body |

| | |
|--------------|--|
| Name | MainOperationsScreen.Create |
| Parameters | None |
| Function | Creates and loads the menu of the Main Operations Screen |
| Sub-function | LoadMenu |
| Callers | Main Body |
| Name | MainOperationsScreen.ListCourses |
| Parameters | Course Names, Department Names |
| Function | List all Courses vs. Departments |
| Sub-function | CoursesScreen.Show |
| Callers | MainOperationsScreen.Main |

| | |
|--------------|--------------------------------|
| Name | MainOperationsScreen.ListDates |
| Parameters | Registration Dates |
| Function | List the registration dates |
| Sub-function | RegDatesScreen.Show |
| Callers | MainOperationsScreen.Main |

| | |
|--------------|---------------------------------|
| Name | MainOperationsScreen.ListRules |
| Parameters | Registration Rules |
| Function | List all the registration rules |
| Sub-function | RulesScreen.Show |
| Callers | MainOperationsScreen.Main |

| | |
|--------------|---|
| Name | MainOperationsScreen.Main |
| Parameters | None |
| Function | Implement the main menu where all information requests are granted |
| Sub-function | MainOperationsScreen.ListCourses, MainOperationsScreen.ListAdvisors, MainOperationsScreen.ListDates, MainOperationsScreen.ListRules, MainOperationsScreen.ViewTranscript, MainOperationsScreen.PrintTranscript, MainOperationsScreen.Hide, RegistrationNotebook.Show, AboutScreen.ShowModal |
| Callers | Main Body |

| | |
|--------------|---|
| Name | MainOperationsScreen.PrintTranscript |
| Parameters | Transcript Style |
| Function | Take the print of the transcript from the printer |
| Sub-function | TranscriptStyleScreen.Show |
| Callers | MainOperationsScreen.Main |

| | |
|--------------|---------------------------------------|
| Name | MainOperationsScreen.ListAdvisors |
| Parameters | Advisor Names, Department Names |
| Function | List all the Advisors vs. Departments |
| Sub-function | AdvisorsScreen.Show |
| Callers | MainOperationsScreen.Main |

| | |
|--------------|-------------------------------------|
| Name | MainOperationsScreen.ViewTranscript |
| Parameters | Transcript Style |
| Function | View the transcript on the screen |
| Sub-function | TranscriptStyleScreen.Show |
| Callers | MainOperationsScreen.Main |

| | |
|--------------|--|
| Name | RegistrationNotebook.AdvisorMeeting.OpticForm |
| Parameters | None |
| Function | Shows the Optic Form Screen and enable student to make changes |
| Sub-function | RegistrationNotebook.Hide, OpticForm.Show |
| Callers | RegistrationNotebook.AdvisorMeeting |

| | |
|--------------|---|
| Name | RegistrationNotebook.AdvisorMeeting.SelectCCs |
| Parameters | None |
| Function | Selection of the complementary courses |
| Sub-function | CoursesScreen.Show, OpticForm.Add |
| Callers | RegistrationNotebook.AdvisorMeeting |

| | |
|--------------|---|
| Name | RegistrationNotebook.AdvisorMeeting.WhyNot |
| Parameters | None |
| Function | Raising a submission of a decision about new course to take |
| Sub-function | OpticForm.Add |
| Callers | RegistrationNotebook.AdvisorMeeting |

| | |
|------------|---------------------------|
| Name | RegistrationNotebook.Help |
| Parameters | Help Context |

| | |
|--------------|---------------------------------------|
| Function | Present the help screen about a topic |
| Sub-function | Winhelp |
| Callers | RegistrationNotebook.Main |

| | |
|--------------|---|
| Name | RegistrationNotebook.Payments.DormitoryFee |
| Parameters | None |
| Function | Simulates the paying process of dormitory fee |
| Sub-function | VisaScreen.Show |
| Callers | RegistrationNotebook.Payments |

| | |
|--------------|--|
| Name | RegistrationNotebook.Payments.RegistrationFee |
| Parameters | None |
| Function | Simulates the paying process of registration fee |
| Sub-function | VisaScreen.Show |
| Callers | RegistrationNotebook.Payments |

| | |
|--------------|---|
| Name | RegistrationNotebook.Payments.Tuition |
| Parameters | None |
| Function | Simulates the paying process of student tuition |
| Sub-function | VisaScreen.Show |
| Callers | RegistrationNotebook.Payments |

| | |
|--------------|--|
| Name | RegistrationNotebook.Stamps.CourseStamps |
| Parameters | Student Number, Course List |
| Function | Simulates the stamping process for the courses |
| Sub-function | None |
| Callers | RegistrationNotebook.Stamps |

| | |
|--------------|--|
| Name | RegistrationNotebook.Stamps.InfirmaryStamp |
| Parameters | Student Number |
| Function | Simulates the stamping process for the infirmary |
| Sub-function | None |
| Callers | RegistrationNotebook.Stamps |

| | |
|--------------|--|
| Name | RegistrationNotebook.Stamps.LibraryStamp |
| Parameters | Student Number |
| Function | Simulates the stamping process for the library |
| Sub-function | None |
| Callers | RegistrationNotebook.Stamps |

| | |
|------|---|
| Name | RegistrationNotebook.Stamps.MilitaryOfficeStamp |
|------|---|

| | |
|--------------|--|
| Parameters | Student Number |
| Function | Simulates the stamping process for the military office |
| Sub-function | None |
| Callers | RegistrationNotebook.Stamps |

| | |
|--------------|--|
| Name | SecurityScreen.Main |
| Parameters | Student Number, Password |
| Function | Authenticate the student with his/her password and school number |
| Sub-function | SecurityScreen.Hide, MainOperationsScreen.Show |
| Callers | CopyrightScreen.Main |

The information given in the upper part, is just presented for informative purposes. Any duplication or modification of the source codes, is strictly forbidden unless permission is taken from Totem Software Development Company.

Chapter 8

Pseudo Codes

Form : LogoScreen

```
Procedure LogoScreen.Create;  
Begin  
    LogoScreen.ShowModal; { shows a modal dialog box }  
    delay(3000);  
End
```

Form : CopyrightNotice

```
Procedure CopyrightScreen.Main;  
Begin  
    CopyrightScreen.Show; { shows a normal window }  
    if agree_selected then  
        begin  
            CopyrightScreen.Hide;  
            SecurityScreen.Show  
        end  
    else if disagree_selected then halt  
        else wait_event  
End
```

Form : SecurityScreen

```

Procedure SecurityScreen.Main;
Begin
    studentno := Input(Box1);
    password := Input(Box2);
    do { maximum 3 times }
    begin
        if password = get_database_pass(studentno)
        then begin
            ok = true;
            SecurityScreen.Hide;
            MainOperationsScreen.Show;
        end
    while trys < 3;
    halt;
End
    
```

Form : MainOperationsScreen

```

Procedure MainOperationsScreen.Create;
Begin
    LoadMenu(MainOperationsMenu)
End
    
```

```

Procedure MainOperationsScreen.Main;
Begin
    wait_menu_command;
    while not(cancel) do
    begin
        case courses_selected : ListCourses
        advisor_selected : ListAdvisors;
        registration_dates_selected : ListDates;
        registration_rules_selected : ListRules;
        transcript_view_selected : ViewTranscript;
        transcript_print_selected : PrintTranscript;
        continue_selected : begin
            MainOperationsScreen.Hide;
            RegistrationNotebook.Show;
        end;
        help_selected : winhelp('Contents');
            { call windows help }
        help_how_to : winhelp('How To');
        help_about : AboutScreen.ShowModal;
    endcase
    wait_menu_command;
    end
End
    
```

```
Procedure MainOperationsScreen.ListCourses;  
Begin  
    courses := getalldatabase(Courses);  
    departments := getalldatabase(Departments);  
    ListBox1.AddItem(courses); { add course names to list box }  
    ListBox2.AddItem(departments);  
    CoursesScreen.Show;  
End
```

```
Procedure MainOperationsScreen.ListAdvisors;  
Begin  
    advisors := getalldatabase(Advisors);  
    departments := getalldatabase(Departments);  
    ListBox1.AddItem(advisors); { add advisor names to list box }  
    ListBox2.AddItem(departments);  
    AdvisorsScreen.Show;  
End
```

```
Procedure MainOperationsScreen.ListRules;  
Begin  
    rules := getalldatabase(Rules);  
    ListBox1.AddItem(rules); { add rules to list box }  
    RulesScreen.Show;  
End
```

```
Procedure MainOperationsScreen.ListDates;  
Begin  
    dates := getalldatabase(Calendar);  
    ListBox1.AddItem(dates); { add dates names to list box }  
    RegDatesScreen.Show;  
End
```

```
Procedure MainOperationsScreen.ViewTranscript;  
Begin  
    transcript_style := TranscriptStyleScreen.Show;  
    while <history present> do  
    begin  
        history := getfromdatabase(Background,studentno);  
        print_on_screen(history)  
    end  
End
```

```
Procedure MainOperationsScreen.PrintTranscript;
```

```
Begin
  transcript_style := TranscriptStyleScreen.Show;
  while <history present> do
  begin
    history := getfromdatabase(Background,studentno);
    print_on_printer(history)
  end
End
```

Form : CoursesScreen

```
Procedure CoursesScreen.Main;
Begin
  if course_on_click then DetailedCourseScreen.Show
  else wait_event
End
```

Form : RegistrationNotebook

```
Procedure RegistrationNotebook.Payments.Tuition;
Begin
  check := check_if_visa_number_entered_before;
  if not(check) then visanumber := VisaScreen.Show;
  check := check_if_tuition_payed_before(Students,studentno);
  if not(check) then Students(studentno).Tuition := true
End
```

```
Procedure RegistrationNotebook.Payments.RegistrationFee;
Begin
  check := check_if_visa_number_entered_before;
  if not(check) then visanumber := VisaScreen.Show;
  check := check_if_regfee_payed_before(Students,studentno);
  if not(check) then Students(studentno).RegFee := true
End
```

```
Procedure RegistrationNotebook.Payments.DormitoryFee;
Begin
  check := check_if_visa_number_entered_before;
  if not(check) then visanumber := VisaScreen.Show;
  check := check_if_dormfee_payed_before
```

```
                                (Students,studentno);  
    if not(check) then Students(studentno).DormFee := true  
End
```

```
Procedure RegistrationNotebook.AdvisorMeeting.SelectCCs;  
Begin  
    while <there is more CC to select > do  
    begin  
        selected := CoursesScreen.Show;  
        if suitable_course then OpticForm.Add(selected)  
            else error;  
    end  
End
```

```
Procedure RegistrationNotebook.AdvisorMeeting.WhyNot;  
Begin  
    course := takedecision; { take the why not question }  
    if suitable_course then OpticForm.Add(course)  
        else error  
End
```

```
Procedure RegistrationNotebook.AdvisorMeeting.OpticForm;  
Begin  
    RegistrationNotebook.Hide;  
    OpticForm.Show  
End
```

```
Procedure RegistrationNotebook.Help;  
Begin  
    winhelp('Registration Notebook')  
End
```

```
Procedure RegistrationNotebook.Stamps.CourseStamps  
Begin  
    while <there are courses that need stamps> do  
    begin  
        if suitable_student(Courses,Students,studentno,course)  
            then stamps_taken := true  
                else stamps_taken := false  
    end  
End
```

```
Procedure RegistrationNotebook.Stamps.LibraryStamp  
Begin  
    if no_books_on_student
```

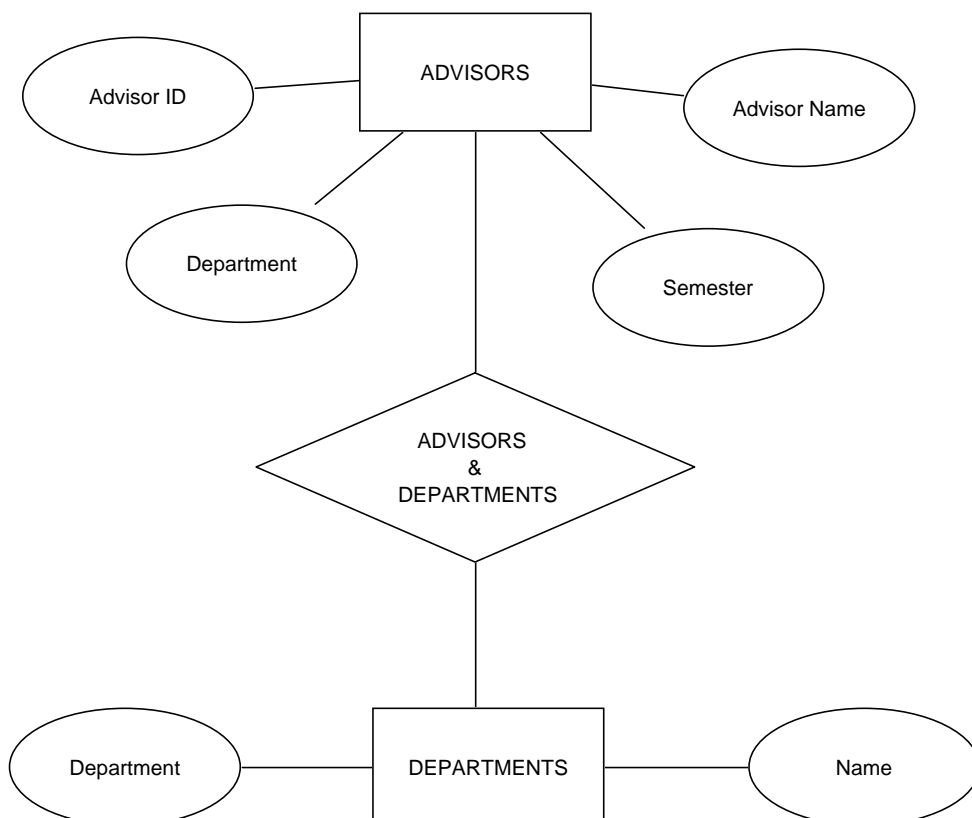
```
        then stamps_taken := true
        else stamps_taken := false
    End
    Procedure RegistrationNotebook.Stamps.MilitaryOfficeStamp
    Begin
        if no_military_obligation_on_student
            then stamps_taken := true
            else stamps_taken := false
        End
    End
```

```
    Procedure RegistrationNotebook.Stamps.InfirmaryStamp
    Begin
        if no_medical_problem_on_student
            then stamps_taken := true
            else stamps_taken := false
        End
    End
```

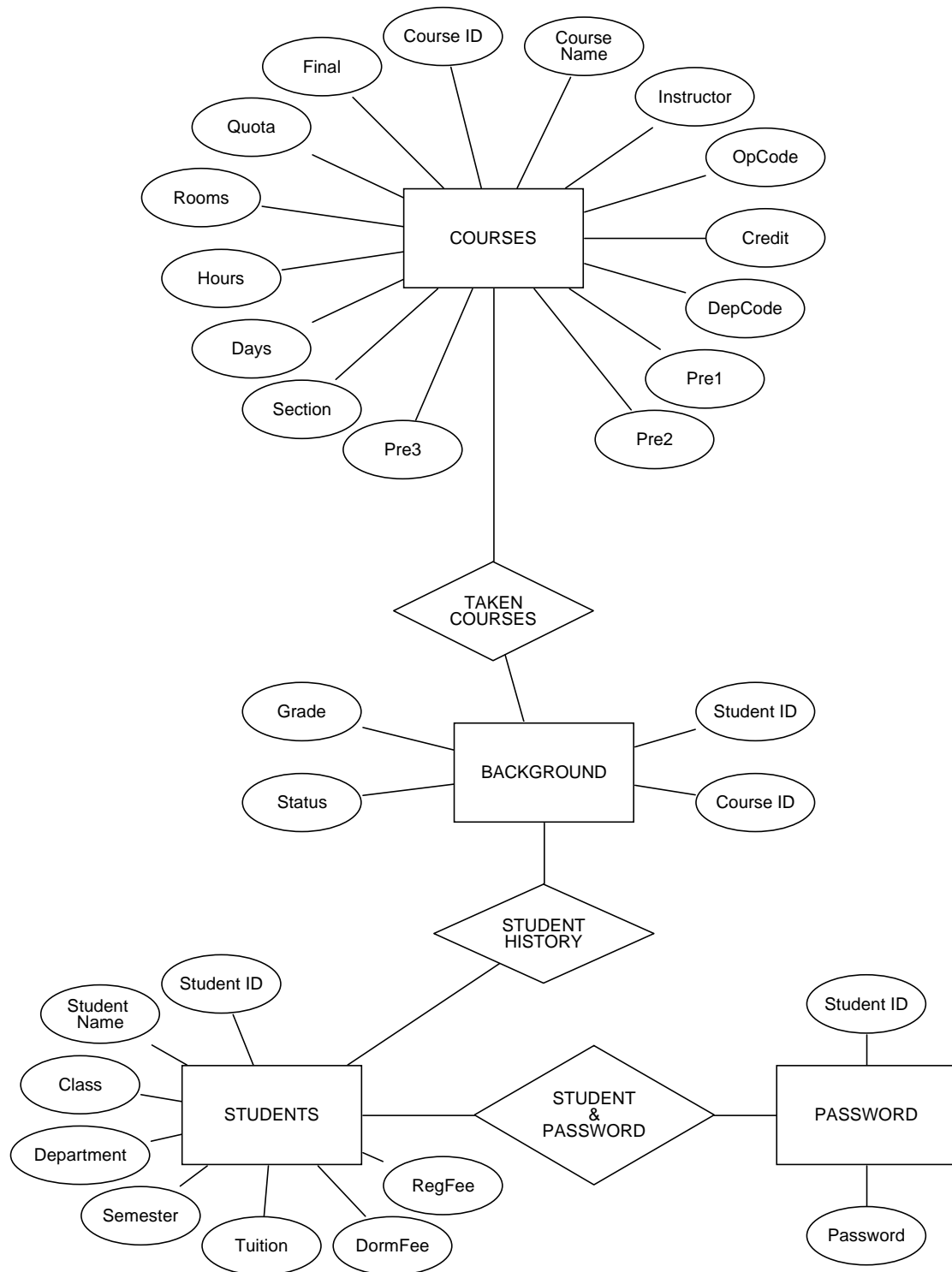
Chapter 9

Entity Relationship Diagrams

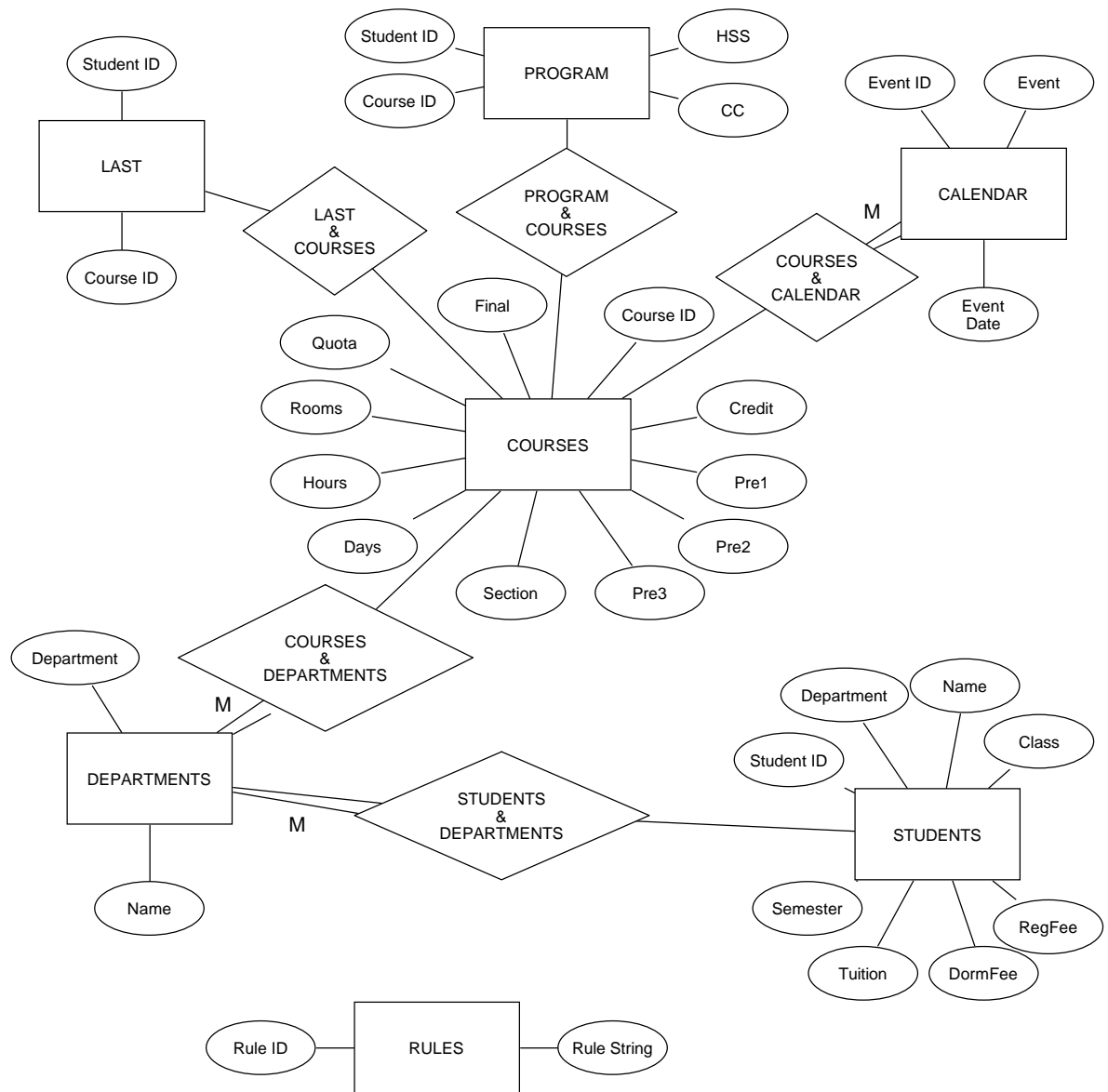
The entity relationship diagrams of the Automated Registration Utility database is fully given as following. The diagrams last for three pages. In this first page, relations between the Advisors and Departments tables are shown :



In this second page, the relations between the Courses, Students, Passwords and Background tables are given as follows :

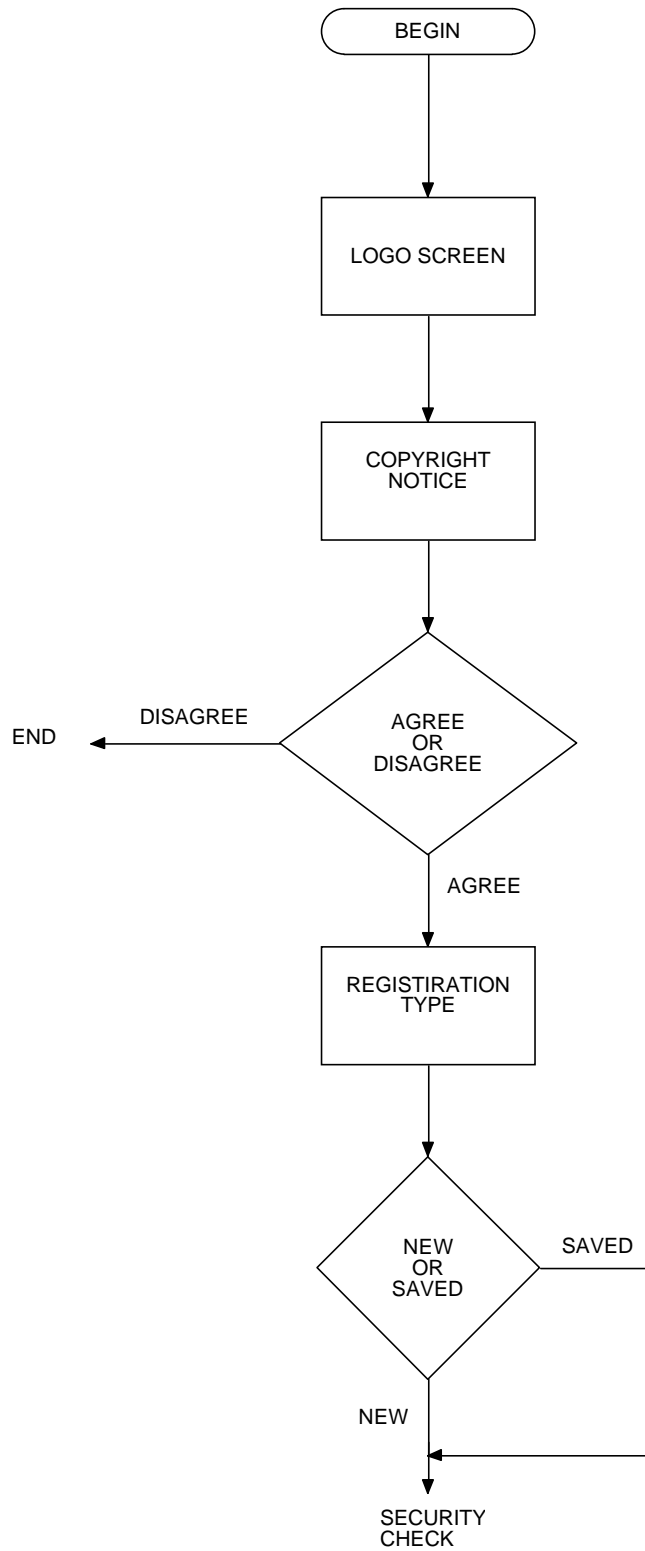


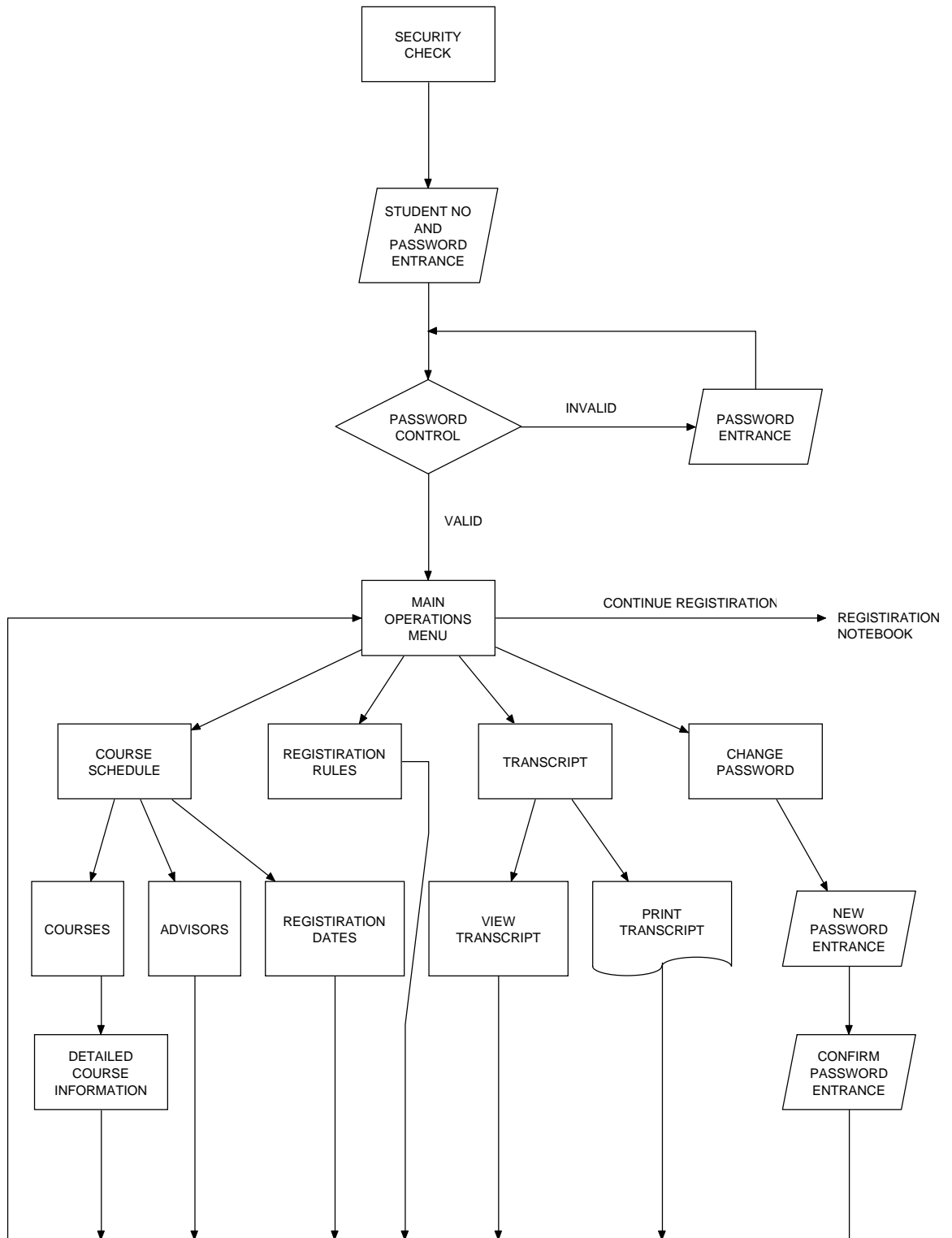
In this third page, the relations between the Last, Program, Courses, Calendar, Departments, Students and Rules tables are given as follows :

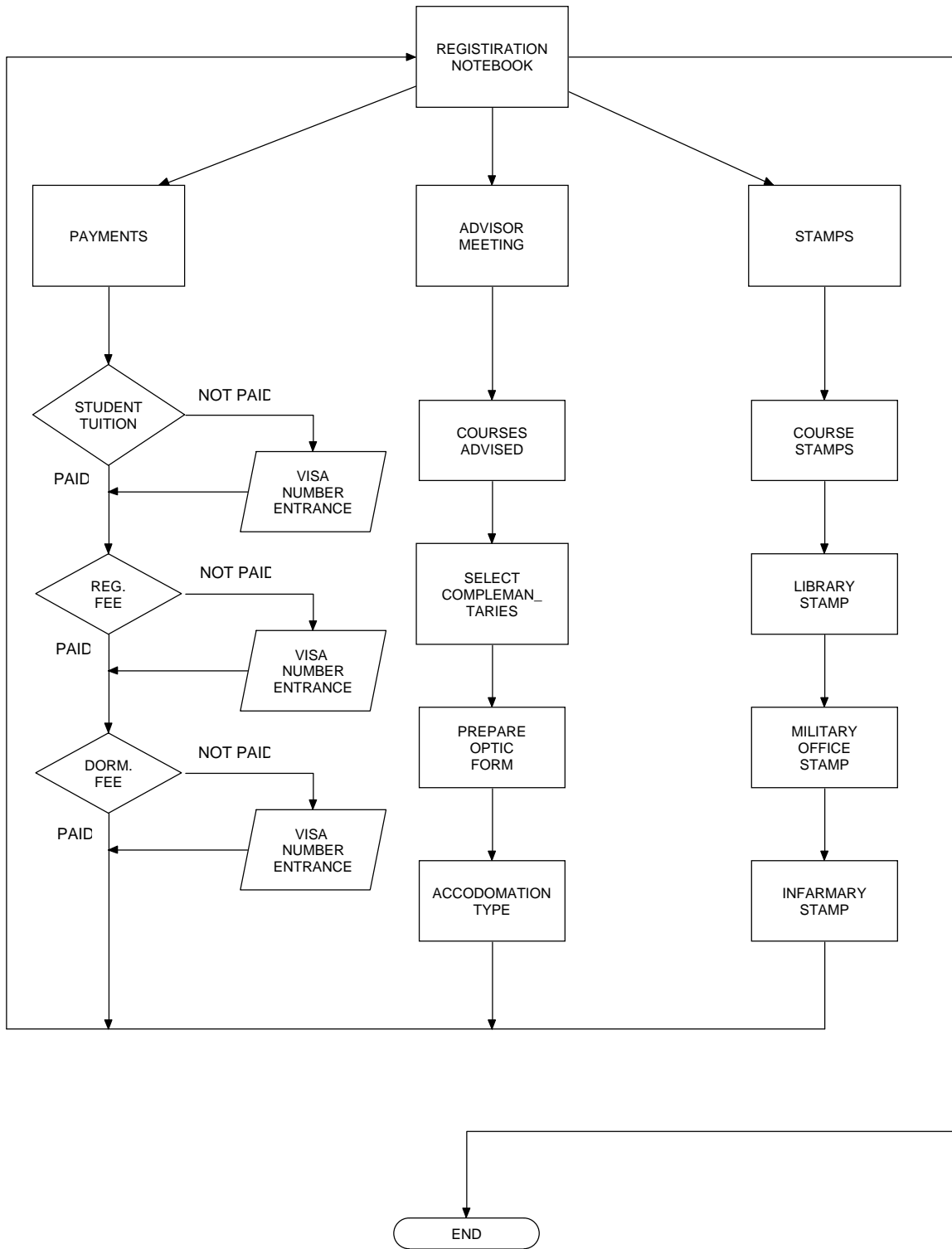


Chapter 10

Structure Chart







Data Flow Diagram

GLOSSARY

A

ARU : Automated Registration Utility.

B

Backup : Copying the data files from hard disk to an other storage device for system and data security.

D

Data : Information stored generated or processed by computers.

Database : Consist of some collection of persistent data that is used by the application systems of some given enterprise.

DOS : Disk Operating System which is a program that acts as an interface between the user and hardware

E

File : A collection of information that can be stored and retrieved from a disk

Form : Visual component of the program.

H

Hard Disk : A fixed data storage environment that provides to user high data access speed and high capacity.

Hardware : The equipment that makes up a computer system.

Help Manager : A utility of Microsoft Windows for getting help.

I

Index : The operation of creating files about data so that faster retrieving of data is achieved.

M

Menu : A collection of items from which you can select one.

Mouse : A pointing device which helps user to make the appropriate selection by moving along the menus.

P

PC : Personal Computer .

Printer : An output device connected to computer for getting printouts.

Program : A set of instructions written in computer language that tells the computer how to perform some tasks.

Q

Query : The form of question about various data in the database with conditions and linking the fields each other if necessary.

R

Record : Data storage unit which consists of some fields.

S

Software : The programs, routines or instructions that allow the computer to perform a task.

W

Windows : Visual operating system developed by MS.

Windows 95 : First 32 bits version of MS Windows.

Index

A

Academic Calendar · 10
Access · 4
Advisors Table · 9, 14
ARU · 2, 3, 4, 6, 11, 15, 22, 32

B

Background Table · 9, 14
binary file · 11
Borland · 2, 3

C

Calendar Table · 10, 14
Component Palette · 5
course schedule · 8
Courses Table · 8, 14

D

Data Elements Dictionary · 7
database · 3, 4
database management system · 4
DDE · 5
decrypt · 10
Delphi · 2, 3, 4
Departments Table · 7, 14
Design Environment · 4
DOS · 11

E

encrypt · 10
entity relationship diagrams · 32
Event · 6

F

File Dictionary · 14
Form · 5

H

hard disk · 6, 14

I

IDE · 4
Instructors Table · 7
integrated development environment · 4

M

Microsoft · 4
Module Dictionary · 22
multitasking · 6

O

Object Inspector · 5
object-oriented · 3

P

Paradox · 2, 4
password · 10
Passwords Table · 10, 14
Pseudo Codes · 26

R

relational database model · 7, 11
Rules Table · 7, 14

S

Structure Chart · 2, 35
Students Table · 8, 14

T

Totem · 2, 11, 26

W

Windows · 3, 6
Windows 95 · 6

Z

ZIP · 15

